Système de preuve modulo récurrence

Eric Deplagne

Département de formation doctorale en informatique
UFR STMIA

École doctorale IAEM Lorraine

Système de preuve modulo récurrence

THÈSE

présentée et soutenue publiquement le 15 novembre 2002

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1
(spécialité informatique)

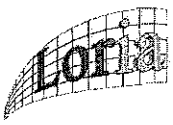
par

Eric Deplagne

Composition du jury

Président :	Jean-Yves Marion	Professeur INPL
Rapporteurs :	Thérèse Hardin	Professeur Université Paris 6
	Siva Anantharaman	Professeur Université d'Orléans
Examinateur :	Adam Cichon	Professeur UHP
Directeur de thèse :	Claude Kirchner	Directeur de Recherches INRIA
Rapporteur interne :	François Charpillet	Directeur de Recherches INRIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503



Mis en page avec la classe thloria.

Je dédie cette thèse à mon oncle Roger. Je ne t'ai connu que pendant deux ans, mais quelque chose en moi se souvient de toi.

Remerciements

Me voici donc face à l'exercice à la fois si traditionnel et si agréable des remerciements.

Je souhaite tout d'abord remercier mes parents, pour tous les sacrifices qu'ils ont fait pour moi. Bien qu'aucun mot ne puisse exprimer combien je leur suis reconnaissant, ma reconnaissance ne saurait être à la hauteur de ce qu'ils méritent.

Mes remerciements iront ensuite tout naturellement à Claude Kirchner, qui a su durant ces quatre années de thèse me guider en supportant mon entêtement comme mes périodes de découragement, notamment pendant la phase de rédaction de cette thèse. C'est à plusieurs reprises ma confiance en lui qui a compensé les failles de ma confiance en moi.

Je souhaite ensuite remercier les membres de mon jury :
- Siva Anantharam et Thérèse Hardin ont accepté la difficile tâche d'être les rapporteurs de cette thèse, je les en remercie très sincèrement.
- François Charpillet a accepté la même tâche en tant que rapporteur interne, et son éloignement par rapport au domaine traité lui a rendu d'autant plus difficile la lecture de ce document.
- Jean-Yves Marion a accepté de présider ce jury et je lui en suis très reconnaissant.
- Adam Cichon a accepté d'être examinateur et représentant de l'université Henri Poincaré au sein de ce jury, ce dont je le remercie vivement.

Je souhaite ensuite remercier Gilles Dowek et Thérèse Hardin pour l'intérêt qu'ils portent à ce travail et pour avoir été les complices de Claude Kirchner dans la conception de la déduction modulo. Utiliser et étendre ce formalisme a été pour moi très agréable.

Je souhaite ensuite remercier Jean-Michel Hufflen et Gilles Richard. Le premier en m'enseignant le λ-calcul puis le second en m'enseignant les substitutions explicites et en encadrant mon stage de DEA sur ce sujet on beaucoup influencé mon orientation scientifique.

Je souhaite ensuite remercier monsieur Couillault et monsieur Pellard, mes professeurs d'informatique en BTS.

Je souhaite ensuite étendre ces remerciements à tous mes enseignants.

Je souhaite également remercier ici tous les membres du projet PROTHEO, ainsi que ceux des projets et équipes CALLIGRAMME, POLKA, MODBIO, CASSIS, SPACES, ADAGE, MIRO, anciens et nouveaux noms d'équipes dont je connais les membres.

Je souhaite étendre ces remerciements à tous les membres du LORIA, grâce à qui le bon environnement matériel qu'offre le laboratoire se double d'un bon environnement humain.

Enfin, en ne nommant personne de peur d'en oublier trop, je souhaite remercier tous ceux qui, au LORIA et sur IRC ont contribué parfois à me faire perdre du temps, mais toujours à me soutenir humainement et moralement. Merci à tous.

Et j'étends encore ces remerciements à tous ceux qui, ne se reconnaissant pas dans ces catégories, ont compté pour moi durant ces quatre années. Je pense notamment à Laurent Ducerf qui reste mon meilleur ami depuis qu'on s'est connu en licence.

Table des matières

Chapitre 1

Introduction

Nécessité des preuves en informatique

Les systèmes informatiques, en tant que tels ou comme composante de systèmes automatiques, ont pris une grande importance dans de nombreux domaines. Dans certains domaines, ludiques ou portant peu de conséquences, tester et éprouver les logiciels peut paraître suffisant. Mais il en est d'autres où cela ne l'est de toute évidence pas.

En effet lorsque les enjeux sont importants, pour des raisons économiques (gestion des transactions boursières et bancaires, ...) et plus encore pour des raisons humaines (gestion d'un métro automatique, gestion des commandes d'un avion, gestion d'une centrale nucléaire, ...), il apparaît comme indispensable de pouvoir vérifier et prouver formellement la fiabilité du système ou au moins son respect de certaines propriétés par rapport à une spécification.

De plus avec l'essor d'internet, on a vu le besoin de sûreté des logiciels apparaître dans des applications en apparence moins complexes. En effet les protocoles qui assurent l'authentification des accès et la sécurisation des données ont un rôle clé notamment pour le développement du commerce électronique. Garantir leur fiabilité est nécessaire afin qu'ils ne deviennent pas le maillon faible du système dont ils sont déjà la partie accessible, et cela nécessite le recours à des preuves formelles souvent difficiles.

Intérêt de la preuve par récurrence

Lorsque l'ensemble des états dans lesquels peut être le système est fini, les méthodes dites de vérification de modèles peuvent convenir. Mais dans le cas contraire, trouver une abstraction finie satisfaisante est un problème difficile, notamment pour s'assurer que l'abstraction faite ne masque pas d'erreur. Un des moyens de s'affranchir de la contrainte de finitude est d'utiliser un raisonnement par récurrence.

Le raisonnement par récurrence est une méthode de preuve fondamentale en mathématiques, qui permet de prouver des propriétés dites inductives. Les propriétés inductives ne sont pas généralement valides (sur tous les modèles), mais elles le sont sur le domaine syntaxique (modèles de Herbrand), ce qui leur donne un grand intérêt car le domaine syntaxique est souvent celui qui nous intéresse pour les propriétés à prouver sur une spécification.

Par exemple, si on a une spécification classique des entiers, avec 0 et la fonction successeur s, et les règles $x + 0 \to x$ et $x + s(y) \to s(x + y)$ définissant la fonction d'addition $+$. Les propriétés classiques de l'addition, commutativité et associativité par exemple, sont des propriétés inductives. C'est le cas également des propriétés sur les protocoles, dont la preuve nécessite généralement la récurrence, sur la longueur des messages notamment.

Diversité des méthodes de preuve par récurrence

Depuis l'émergence de l'informatique, les preuves par récurrence ont été étudiées comme l'un des concepts fondamentaux à automatiser pour les nombreuses preuves du domaine mathématique utilisant la récurrence. Elles le sont encore plus aujourd'hui parce que les propriétés que l'on souhaite prouver au sujet d'un programme ou de sa spécification sont en général des propriétés inductives. C'est pourquoi les preuves par récurrence ont un rôle critique tant dans les assistants de preuve que dans les démonstrateurs automatiques.

Les preuves par récurrence peuvent être effectuées de façons diverses qu'il n'est pas évident de relier à un niveau autre que sémantique. Je classerai ici les méthodes en trois familles :

- La première famille est celle des méthodes utilisant explicitement le principe de récurrence noethérienne ou un principe de récurrence plus restreint mais directement relié, comme le principe de récurrence par les constructeurs dont le principe de récurrence de Peano est l'exemple type.
  Cette méthode est la plus évidente du point de vue de sa justification formelle, mais elle est difficile à automatiser dans toute sa généralité, notamment parce qu'elle s'exprime le plus naturellement dans un cadre d'ordre supérieur.
- La deuxième famille de méthodes utilise la réécriture. Les deux grandes méthodes dans cette famille sont la récurrence par consistance et la récurrence par réécriture.
  L'intérêt de ces méthodes est la simplicité du cadre de la réécriture. Leur difficulté est la démonstration que les preuves effectuées sont bien des preuves de propriétés inductives. Démontrer, au niveau des preuves, que les preuves effectuées à l'aide de ces méthodes sont des preuves par récurrence est l'objectif auquel je souhaite contribuer dans cette thèse.
- La troisième famille, qui n'est pas traitée ici, est appelée "descente infinie" [Wir00] et vise à simuler la façon informelle que peut avoir un mathématicien de décrire une preuve par récurrence.
  Cette méthode est particulièrement adaptée à des preuves par l'absurde, où l'on réfute la possibilité de trouver un contre-exemple minimal.

Diversité des systèmes effectuant des preuves par récurrence

Cette diversité au niveau des méthodes de preuve par récurrence s'est naturellement trouvée reflétée et même amplifiée au niveau des systèmes qui les implémentent.

Les méthodes de récurrence par consistance et de récurrence par réécriture sont par nature fortement automatiques. Elles sont implémentées par des systèmes comme Spike, RRL ou encore LP (the Larch prover).

Il en va bien sûr tout autrement de l'usage explicite du principe de récurrence noethérienne. On trouve deux catégories de systèmes implémentant ce type de méthodes :

- Une première famille comprenant notamment Coq, Isabelle, HOL et PVS est celle des assistants de preuve, dans lesquels chaque étape de la preuve est spécifiée par l'utilisateur ou par une "tactique".
- Une autre catégorie ou l'on trouve notamment NQTHM et son successeur ACL2, est celle où bien qu'utilisant le principe de récurrence de façon explicite, on détermine automatiquement les schémas de récurrence à employer afin de rendre les preuves automatiques.

Comparaison des systèmes et intérêt d'une coopération

Les assistants de preuve permettent de réaliser des preuves complexes, mais leur utilisation est rendue délicate et peu attrayante pour un non expert par la nécessité de spécifier chaque étape, même si les "tactiques" permettent de définir des séquences qu'on pourra alors ne plus répéter.

Les prouveurs automatiques ont l'avantage de ne pas requérir d'intervention de la part de l'utilisateur au cours de la preuve, mais ils ne peuvent pas réaliser certaines preuves possibles avec un assistant de preuve, et il faut parfois écrire la spécification de façon appropriée.

Ces caractéristiques font qu'on souhaiterait pouvoir faire coopérer différents systèmes, notamment un assistant de preuve et un prouveur automatique, afin d'effectuer automatiquement les preuves que le prouveur sait gérer tout en ayant la possibilité d'interragir au niveau de l'assistant de preuve pour terminer les preuves plus difficiles.

Intérêt du mode sceptique

Pour assurer un maximum de fiabilité, des systèmes comme Coq utilisent l'isomorphisme de Curry-Howard pour que le typage d'un terme de preuve par un noyau de taille restreinte permette de s'assurer qu'une preuve est correcte. Ainsi la fiabilité de l'ensemble du système est assurée par la seule fiabilité de ce noyau, puisque toute erreur dans le reste du système sera détectée par le noyau lors de la vérification de la preuve.

Dans une telle approche, chaque étape de preuve doit construire le terme de preuve permettant de la vérifier. Ainsi on ne peut se contenter pour une coopération entre cette approche et un prouveur automatique d'un mode confiant où la réponse est de la forme "oui" ou "non", mais il faut utiliser un mode sceptique où la réponse est un terme de preuve.

Un autre intérêt du terme de preuve est de pouvoir, toujours grâce à l'isomorphisme de Curry-Howard, et sous certaine conditions (preuve constructive, donc intuitionniste), en extraire un programme certifié qui sera donc assuré d'être conforme à sa spécification.

Cadre de coopération

Construire une coopération dans le mode sceptique nécessite un cadre unifié dans lequel comprendre la méthode employée par le prouveur en termes de celle que peut vérifier l'assistant de preuve. Construire un tel cadre et amorcer la compréhension de la méthode de preuve par récurrence par réécriture est l'objectif de cette thèse. Construire effectivement la coopération sceptique en sera une perspective naturelle.

Notre cadre doit permettre d'unifier le cadre d'ordre supérieur qui est celui des assistants de preuve, et le plus naturel de l'usage explicite du principe de récurrence noethérienne, avec le cadre de réécriture du premier ordre utilisé pour la récurrence par réécriture.

La déduction modulo [DHK98] est un formalisme construit pour permettre de distinguer clairement la partie *calcul* et la partie *déduction* d'un raisonnement. La partie calcul est alors placée (internalisée) dans une congruence modulo laquelle on va raisonner. La déduction modulo permet également de raisonner à l'ordre supérieur grâce à la formulation de la logique d'ordre supérieur de [DHK01], obtenue en plaçant dans la congruence un calcul de substitutions explicites ([ACCL91]). Une propriété importante de cette formulation est qu'elle est intentionnellement équivalente à la formulation classique, ce qui lui permet de préserver les propriétés algorithmiques d'un programme extrait.

Extension de la déduction modulo

Pour gérer les preuves par récurrence, j'ai étendu le cadre de la déduction modulo. En effet, les hypothèses de récurrence que l'on va vouloir internaliser sont par nature conditionnelles, ce qui n'est pas géré par le cadre initial de la déduction modulo. Gérer les règles conditionnelles m'a également amené à devoir prendre en compte le contexte dans lequel s'évalue la congruence, et notamment la condition.

D'autre part, l'ordre de récurrence ne peut pas être rendu compatible avec la congruence. Pour gérer cela j'ai introduit la notion de symbole protecteur. Un symbole protecteur empêche la congruence de fonctionner sur ses arguments. Ainsi bien que $s(x) + 0$ soit équivalent à $s(x)$, $s(x) + 0 > s(s(x))$ n'est pas équivalent à $s(x) > s(s(x))$, grâce au fait que $>$ est un symbole protecteur.

Traitement de la récurrence

On parvient dans cette extension de la déduction modulo à internaliser les hypothèses de récurrence, ce qui permet de placer les preuves effectuées par le prouveur dans la partie calcul, en conservant le cadre de l'assistant de preuve dans la partie déduction. On parvient ainsi à comprendre la méthode de récurrence par réécriture comme le résultat de l'internalisation des hypothèses de récurrence en déduction modulo.

En mettant en forme les preuves, on parvient également à expliquer l'usage d'hypothèses de récurrence provenant des autres buts, dans le cas d'une preuve simultanée de plusieurs buts. Ce point est un des

comportements étonnants mais particulièrement puissants de la méthode de preuve par récurrence par réécriture.

On donne aussi une preuve syntaxique d'une propriété permettant de ne pas vérifier explicitement la condition de récurrence, ce qui est un autre comportement étonnant de la méthode de preuve par récurrence par réécriture.

Plan de la thèse

- Dans le premier chapitre, on présente la logique du premier ordre. Après le rappel des définitions classiques de la logique du premier ordre, on donne des définitions permettant d'utiliser des prédicats d'égalité.
  Contrairement aux définitions classiques sur les prédicats d'égalité, on permet de contrôler l'action de ces prédicats par des symboles protecteurs. Ces symboles protecteurs ont pour rôle de bloquer l'égalité lorsque la sémantique de certaines opérations, comme la comparaison par un ordre '$>$' le nécessite.
- Le deuxième chapitre présente la déduction modulo [DHK98], dans une version étendue à l'utilisation de règles conditionnelles et à la présence de symboles protecteurs grâce aux définitions du premier chapitre.
  On présente d'abord la déduction modulo et ses propriétés indépendamment de la représentation choisie pour la congruence. On montre ensuite que la représentation à l'aide des systèmes de réécriture de classes conditionnels définis au chapitre précédent permet de reconstruire la contrepartie logique d'une congruence, ce qui est la propriété que l'on attend d'une bonne représentation.
- Le troisième chapitre présente le traitement que l'on peut faire de la logique du premier ordre elle-même avec les idées de la déduction modulo. Cette idée reprise de [Vir98] est ici corrigée afin d'obtenir les bonnes propriétés. Le système obtenu ne comporte plus dans sa partie déduction que les règles connues pour être responsables de l'indécidabilité de la logique du premier ordre, c'est-à-dire celles traitant les quantificateurs.
- Le quatrième chapitre présente la logique d'ordre supérieur et sa formulation dans le cadre de la déduction modulo [DHK01].
  On présente d'abord le λ-calcul et sa version typée. On présente ensuite la représentation de la logique d'ordre supérieur à l'aide du λ-calcul typé (HOL_λ). On présente ensuite les λ-calculs avec substitutions explicites et on détaille $\lambda\sigma$, dont on va se servir pour représenter la logique d'ordre supérieur dans le cadre de la déduction modulo.
- Le cinquième chapitre présente la méthode de preuve par récurrence noethérienne et la méthode de récurrence par réécriture.
  On présente les différentes phases de la méthode de preuve par récurrence noethérienne : mise en forme et généralisation, choix de la variable de récurrence, choix du schéma d'instanciation et choix de l'ordre de récurrence.
  On présente la définition des ensembles couvrants, puis la définition des ensembles test dans le cas général et dans le cas où l'on a des constructeurs libres.
  On présente ensuite un exemple de spécification Spike, ainsi qu'un exemple de preuve et un exemple de réfutation à l'aide du système Spike dans cette spécification.
  Enfin on présente rapidement la méthode de récurrence par consistance.
- Le sixième chapitre montre comment le fait de voir l'utilisation du principe de récurrence noethérienne dans un cadre de déduction modulo permet de comprendre la méthode de récurrence par réécriture. Après avoir montré comment obtenir et mettre en forme de puissantes hypothèses de récurrence, de façon à les internaliser lorsqu'elles sont équationnelles, on montre comment un ordre comme celui utilisé par Spike permet de ne pas vérifier explicitement la condition de récurrence. On termine par le résumé de la méthode et quelques exemples.

Première partie

Logique du premier ordre

Chapitre 2

Logique du premier ordre

Je donne dans ce chapitre les définitions de logique du premier ordre dont on va se servir par la suite.

Dans un premier temps, je rappelle les définitions classiques de la syntaxe et de la sémantique de la logique du premier ordre, ainsi que le calcul de séquents [GLT89, Gal86] qui permet de représenter les preuves.

Ensuite je donne des définitions permettant d'utiliser des prédicats d'égalité et ceci de deux façons dont on verra au chapitre 3 comment elles se correspondent. Contrairement aux définitions classiques de [Gal86] par exemple, on a ici la possibilité de restreindre (par des symboles protecteurs) les prédicats d'égalité.

La première façon d'utiliser les prédicats d'égalité consiste à définir la théorie qu'ils doivent satisfaire. La seconde est d'utiliser les systèmes de réécriture de classes conditionnels et les relations qu'ils définissent.

2.1 Logique du premier ordre

2.1.1 Syntaxe

2.1.1.1 Termes

Définition 2.1 Soit $\mathcal{F}$ un ensemble fini de symboles de fonctions avec leur arité.

$\mathcal{T}(\mathcal{F})$ désigne l'ensemble des *termes clos* construits sur $\mathcal{F}$ avec la règle :

- si f d'arité n est dans $\mathcal{F}$ et $t_1, \ldots, t_n$ sont dans $\mathcal{T}(\mathcal{F})$ alors $f(t_1, \ldots, t_n)$ est dans $\mathcal{T}(\mathcal{F})$.

Exemple 2.1 Soit $\mathcal{F} = \{a : 0, b : 0, c : 0, g : 1, f : 2, h : 3\}$. Des exemples de termes clos sont :

- a,
- $g(b)$,
- $f(a, g(b))$,
- $h(f(a, g(b)), g(a), a)$.

Définition 2.2 Soient $\mathcal{F}$ un ensemble fini de symboles de fonctions avec leur arité et $\mathcal{X}$ un ensemble infini dénombrable de symboles de variables.

$\mathcal{T}(\mathcal{F}, \mathcal{X})$ désigne l'ensemble des *termes* construits sur $\mathcal{F}$ et $\mathcal{X}$ avec les règles :

- si x est dans $\mathcal{X}$ alors x est dans $\mathcal{T}(\mathcal{F}, \mathcal{X})$,
- si f d'arité n est dans $\mathcal{F}$ et $t_1, \ldots, t_n$ sont dans $\mathcal{T}(\mathcal{F}, \mathcal{X})$ alors $f(t_1, \ldots, t_n)$ est dans $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Remarque 2.1 Tout terme clos dans $\mathcal{T}(\mathcal{F})$ est également un terme dans $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Exemple 2.2 Soient $\mathcal{F}$ comme précédemment et $\mathcal{X} = \{x, y, z, u, \ldots\}$. Des exemples de termes sont :

- x,
- $g(y)$,
- $f(a, g(y))$,
- $h(f(a, g(x)), g(b), y)$.

Remarque 2.2 Certains symboles de fonction sont parfois utilisés de façon infixée, notamment pour respecter l'usage. Par exemple on notera $x + y$, pour $+(x, y)$. Ce type de notation n'a pas d'influence à part sur la lisibilité.

2.1.1.2 Formules atomiques

Définition 2.3 Soit $\mathcal{P}$ un ensemble fini de symboles de prédicats avec leur arité.

On appelle $\mathcal{AP}(\mathcal{P}, \mathcal{F}, \mathcal{X})$ l'ensemble des *formules atomiques* construites sur l'ensemble des prédicats $\mathcal{P}$ et l'ensemble des termes $\mathcal{T}(\mathcal{F}, \mathcal{X})$ avec les règles :

- si P d'arité n est dans $\mathcal{P}$ et $t_1, \ldots, t_n$ sont dans $\mathcal{T}(\mathcal{F}, \mathcal{X})$ alors $P(t_1, \ldots, t_n)$ est dans $\mathcal{AP}(\mathcal{P}, \mathcal{F}, \mathcal{X})$.

Exemple 2.3 Soient $\mathcal{F}$ et $\mathcal{X}$ comme précédemment et $\mathcal{P} = \{P : 0, Q : 1, R : 2\}$. Des exemples de formules atomiques sont :

- P,
- $Q(a)$,
- $Q(x)$,
- $R(a, x)$.

2.1.1.3 Formules non atomiques

Définition 2.4 Soit $\mathcal{P}$ un ensemble fini de symboles de prédicats avec leur arité.

$\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$ est l'ensemble des *formules du premier ordre* construit sur les formules atomiques dans $\mathcal{AP}(\mathcal{P}, \mathcal{F}, \mathcal{X})$ avec les connecteurs $\wedge, \vee, \Rightarrow, \neg, \bot$ et les quantificateurs $\forall$ et $\exists$ avec les règles :

- si F est dans $\mathcal{AP}(\mathcal{P}, \mathcal{F}, \mathcal{X})$ alors F est dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$,
- $\bot$ est dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$,
- si F est dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$, alors $\neg F$ est dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$,
- si F_1 et F_2 sont dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$ alors $F_1 \wedge F_2$, $F_1 \vee F_2$ et $F_1 \Rightarrow F_2$ sont dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$,
- si x est dans $\mathcal{X}$ et F est dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$ alors $\exists x F$ et $\forall x F$ sont dans $\mathcal{Prop}(\mathcal{P}, \mathcal{F}, \mathcal{X})$.

Exemple 2.4 Soient $\mathcal{F}$, $\mathcal{X}$ et $\mathcal{P}$ comme précédemment. Des exemples de formules du premier ordre sont :

- $\bot$,
- $Q(a) \wedge R(b, x)$,
- $Q(b) \vee R(y, a)$,
- $P \Rightarrow Q(a)$,
- $\forall x\, Q(x)$,
- $\forall x\, (Q(x) \Rightarrow R(x, g(x)))$,
- $\exists x\, Q(x)$,
- $\exists x\, (Q(x) \wedge R(x, g(y)))$.

Remarque 2.3 Si on met toutes les parenthèses nécessaires à supprimer les ambiguïtés, les formules deviennent rapidement difficiles à lire. On utilisera donc les conventions suivantes :

- $\neg$ a une priorité plus faible que les connecteurs binaires, ainsi $\neg P \wedge Q(x)$ signifie $(\neg P) \wedge Q(x)$,
- les quantificateurs $\exists$ et $\forall$ ont une priorité plus grande que les connecteurs, ainsi $\forall x\, Q(x) \Rightarrow P$ signifie $\forall x\, (Q(x)) \Rightarrow P$, parfois aussi noté $(\forall x\, Q(x)) \Rightarrow P$ s'il y a un risque d'ambiguïté.

Remarque 2.4 Une variable située dans la portée d'un quantificateur est dite liée, sinon elle est dite libre. Ainsi dans $\forall x\, R(x, y)$, x est liée et y est libre.

Remarque 2.5 La liste de connecteurs donnée ici n'est pas minimale pour la logique classique, en effet n'importe lequel des connecteurs $\wedge$, $\Rightarrow$ ou $\vee$ suffit, accompagné de $\neg$, à définir les deux autres. De même chacun des quantificateurs $\forall$ et $\exists$, accompagné de $\neg$, permet de définir l'autre. On peut également soit définir $\neg A$ par $A \Rightarrow \bot$, soit se passer de $\bot$.

Cependant, il n'est pas possible de réduire le nombre de connecteurs dans d'autres logiques, notamment la logique intuitionniste, dans lesquelles les liens entre les connecteurs sont différents, et notamment le symbole $\bot$ indispensable à la définition de $\neg$.

Remarque 2.6 On pourrait considérer $\bot$ comme une formule atomique, mais cela rendrait le traitement des formules atomiques moins uniforme, notamment en ce qui concerne la sémantique.

Remarque 2.7 Il est facile en logique classique du premier ordre de définir des connecteurs supplémentaires, par exemple on utilise souvent :

$$F_1 \Leftrightarrow F_2 \triangleq (F_1 \Rightarrow F_2) \wedge (F_2 \Rightarrow F_1).$$

Remarque 2.8 Comme pour les symboles de fonction, certains symboles de prédicat sont parfois utilisés de façon infixée. Par exemple on notera $x \in y$, pour $\in(x, y)$.

Notation 2.1 Pour améliorer la lisibilité :
- $x_1, \ldots, x_n$ sera noté $\overline{x}$,
- $x_1\, O\, y_1 \wedge \cdots \wedge x_n\, O\, y_n$ sera noté $\overline{x}\, O\, \overline{y}$, où O est n'importe quel symbole (opérateur) binaire,
- on note $Q(\overline{t}, v_i, \overline{s})$ si v_i est le $i^{\text{ème}}$ argument de Q.

2.1.1.4 Positions

Définition 2.5 Pour repérer une partie d'un terme, ou un symbole particulier, on les désigne par leur *position*. On repère chaque position par la séquence des numéros d'arguments permettant d'atteindre cette position. La tête du terme est notée Λ.

Exemple 2.5 Ainsi dans le terme $h(f(a, g(b)))$, a est en position 1.1, $g(b)$ en position 1.2, et b en position 1.2.1.

Notation 2.2 On note $t|_p$ le sous-terme de t à la position p. On note $t[t']_p$ le fait que t a pour sous-terme à la position p le terme t'.

Ces définitions s'étendent naturellement aux formules.

Remarque 2.9 Les quantificateurs peuvent être considérés comme unaires, et indissociables de la variable qu'ils lient, ou binaires. On les considérera ici comme unaires, ce qui est le choix le plus fréquent.

2.1.1.5 Substitutions

Définition 2.6 Une *substitution* est une application de $\mathcal{X}$ vers $\mathcal{T}(\mathcal{F}, \mathcal{X})$.

Définition 2.7 Une substitution de $\mathcal{X}$ vers $\mathcal{T}(\mathcal{F})$ est appelée une *substitution close*.

Notation 2.3 On note $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n, \ldots\}$ la substitution donnant à x_1 la valeur t_1, ..., à x_n la valeur t_n, ... On simplifie la notation en ne mentionnant que les variables effectivement modifiées.

Une substitution s'étend naturellement aux termes, ainsi qu'aux formules sans quantificateurs. Pour les quantificateurs en revanche il faut faire attention à la liaison des variables, et aux captures de variables libres par un quantificateur.

Définition 2.8 Soit σ une substitution.
- $\sigma(\forall x\, F(x)) = \forall u\, \sigma(F(u))$,
- $\sigma(\exists x\, F(x)) = \exists u\, \sigma(F(u))$.

où $F(u) = \tau(F(x))$ avec $\tau = \{x \mapsto u\}$, et u est une variable n'apparaissant pas dans σ (u n'est aucun des x_i, ni n'apparaît dans un des t_i).

Exemple 2.6 Soient $\mathcal{F}$, $\mathcal{P}$ et $\mathcal{X}$ comme précédemment. Soit $\sigma = \{x \mapsto f(a, b), y \mapsto g(z)\}$.
- $\sigma(Q(f(x, y))) = Q(f(f(a, b), g(z)))$,
- $\sigma(\forall x\, Q(f(x, y))) = \forall u\, Q(f(u, g(z)))$,
- $\sigma(\forall z\, Q(f(y, z))) = \forall u\, Q(f(g(z), u))$.

2.1.2 Sémantique

2.1.2.1 Interprétations

Définition 2.9 On appelle *Bool* l'ensemble des booléens $\{faux, vrai\}$.

Définition 2.10 Une *interprétation* I sur un domaine D associe :
- à chaque symbole de fonction f d'arité n dans $\mathcal{F}$ une fonction de D^n dans D,
- à chaque symbole de prédicat P d'arité n dans $\mathcal{P}$ une fonction de D^n dans $Bool$.

2.1.2.2 Valuations

Définition 2.11 Une *valuation* v associe à chaque variable dans $\mathcal{X}$ une valeur dans D. Elle s'étend aux termes et aux formules atomiques grâce a l'interprétation, et aux formules non atomiques par les règles :
- $v(\bot) = faux$,
- $v(\neg F) = vrai$ si et seulement si $v(F) = faux$,
- $v(F_1 \wedge F_2) = vrai$ si et seulement si $v(F_1) = vrai$ et $v(F_2) = vrai$,
- $v(F_1 \vee F_2) = vrai$ si et seulement si $v(F_1) = vrai$ ou $v(F_2) = vrai$,
- $v(F_1 \Rightarrow F_2) = vrai$ si et seulement si $v(F_1) = faux$ ou $v(F_2) = vrai$,
- $v(\exists x F) = vrai$ si et seulement si il existe une valeur d dans D telle que $v(F\{d/x\}) = vrai$,
- $v(\forall x F) = vrai$ si et seulement si pour toute valeur d dans D, $v(F\{d/x\}) = vrai$.

2.1.2.3 Modèles

Définition 2.12 Un *modèle* d'une formule F est une interprétation telle que toute valuation rend F vraie.

Définition 2.13 Une formule F est
- *valide* si toute interprétation de F est un modèle de F,
- *satisfiable* s'il existe une interprétation de F qui est un modèle de F,
- *insatisfiable* s'il n'existe aucune interprétation de F qui soit un modèle de F.

Exemple 2.7 Soient $\mathcal{F}$, $\mathcal{P}$ et $\mathcal{X}$ comme précédemment :
- $\forall x\,(Q(x) \vee \neg Q(x))$ est valide,
- $\exists x\, Q(x)$ est satisfiable,
- $\bot$ est insatisfiable.

2.1.2.4 Conséquences

Définition 2.14 Une formule F est une *conséquence* (déductive) d'un ensemble de formules Γ si toute interprétation qui est modèle de toutes les formules de Γ est aussi un modèle de F.

Exemple 2.8 Soient $\mathcal{F}$ et $\mathcal{X}$ comme précédemment et $\mathcal{P} = \{P : 1, R : 1\}$:
- $\forall x\, R(x)$ est une conséquence de $\{\forall x\, P(x), \forall x\,(P(x) \Rightarrow R(x))\}$.

2.1.2.5 Interprétations de Herbrand

Définition 2.15 Une *interprétation de Herbrand* est une interprétation telle que :
- $D = \mathcal{T}(\mathcal{F})$,
- pour tout symbole de fonction f dans $\mathcal{F}$, $I(f) = f$.

2.1.2.6 Modèles de Herbrand

Définition 2.16 Un *modèle de Herbrand* d'une formule F est une interprétation de Herbrand qui est un modèle de F.

Définition 2.17 Une formule F est
- *inductivement valide* si toute interprétation de Herbrand de F est un modèle (de Herbrand) de F,
- *inductivement satisfiable* s'il existe une interprétation de Herbrand de F qui est modèle (de Herbrand) de F.

Exemple 2.9 Soient $\mathcal{F} = \{a : 0, b : 0\}$, $\mathcal{P} = \{R : 1\}$ et $\mathcal{X} = \{x\}$:
- $(R(a) \wedge R(b)) \Rightarrow \forall x\, R(x)$ est inductivement valide, mais n'est pas valide,
- $(\neg R(a) \wedge \neg R(b)) \Rightarrow \exists x\, R(x)$ est satisfiable, mais n'est pas inductivement satisfiable.

2.1.2.7 Conséquences inductives

Définition 2.18 Une formule F est une *conséquence inductive* d'un ensemble de formules Γ si toute interprétation de Herbrand qui est modèle de toutes les formules de Γ est aussi un modèle (de Herbrand) de F.

Exemple 2.10 Soient $\mathcal{F} = \{a : 0, b : 0\}$, $\mathcal{P} = \{R : 1\}$ et $\mathcal{X} = \{x\}$:
- $\forall x\, R(x)$ est une conséquence inductive de $\{R(a), R(b)\}$, mais n'en est pas une conséquence déductive.

$$\frac{}{P \vdash P}\text{axiome} \qquad \frac{\Gamma, P \vdash \Delta \quad \Gamma \vdash P, \Delta}{\Gamma \vdash \Delta}\text{coupure}$$

$$\frac{\Gamma, P, P \vdash \Delta}{\Gamma, P \vdash \Delta}\text{contraction-g} \qquad \frac{\Gamma \vdash P, P, \Delta}{\Gamma \vdash P, \Delta}\text{contraction-d}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, P \vdash \Delta}\text{affaiblissement-g} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash P, \Delta}\text{affaiblissement-d}$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta}\wedge\text{-g} \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta}\wedge\text{-d}$$

$$\frac{\Gamma, P \vdash \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, P \vee Q \vdash \Delta}\vee\text{-g} \qquad \frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash P \vee Q, \Delta}\vee\text{-d}$$

$$\frac{\Gamma \vdash P, \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, P \Rightarrow Q \vdash \Delta}\Rightarrow\text{-g} \qquad \frac{\Gamma, P \vdash Q, \Delta}{\Gamma \vdash P \Rightarrow Q, \Delta}\Rightarrow\text{-d}$$

$$\frac{\Gamma \vdash P, \Delta}{\Gamma, \neg P \vdash \Delta}\neg\text{-g} \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash \neg P, \Delta}\neg\text{-d}$$

$$\frac{}{\Gamma, \bot \vdash \Delta}\bot\text{-g}$$

$$\frac{\Gamma, Q\{t/x\} \vdash \Delta}{\Gamma, \forall x\, Q \vdash \Delta}(Q, x, t)\ \forall\text{-g} \qquad \frac{\Gamma \vdash Q\{y/x\}, \Delta}{\Gamma \vdash \forall x\, Q, \Delta}(Q, x, y)\ \forall\text{-d}$$

$$\frac{\Gamma, Q\{y/x\} \vdash \Delta}{\Gamma, \exists x\, Q \vdash \Delta}(Q, x, y)\ \exists\text{-g} \qquad \frac{\Gamma \vdash Q\{t/x\}, \Delta}{\Gamma \vdash \exists x\, Q, \Delta}(Q, x, t)\ \exists\text{-d}$$

Dans $\forall$-d et $\exists$-r, y doit être une nouvelle variable libre.
Dans $\forall$-r et $\exists$-d, t est un terme quelconque.

FIG. 2.1 – Le calcul des séquents

2.1.3 Calcul des séquents

Le calcul des séquents [GLT89, Gal86] (figure 2.1) permet de représenter les preuves. Du côté gauche du symbole $\vdash$ se trouve un ensemble Γ d'hypothèses. Du côté droit du symbole $\vdash$ se trouve un ensemble Δ de conclusions. Le jugement $\Gamma \vdash \Delta$ signifie "la conjonction des hypothèses dans Γ implique la disjonction des conclusions dans Δ".

2.2 Prédicats d'égalité

2.2.1 Symboles protecteurs

Les *symboles protecteurs* permettent d'empêcher les prédicats d'égalité d'agir sous certains symboles dont la sémantique et la définition ne sont pas compatibles avec une telle modification de leurs arguments.

Si on prend par exemple $\mathcal{F} = \{+, s, 0\}$ avec les égalités habituelles $x + 0 \asymp x$ et $x + s(y) \asymp s(x + y)$, on ne veut pas transformer $s(x) + 0 > s(s(x))$ en $s(x) > s(s(x))$ puisqu'un ordre rpo avec la précédence $+ > s > 0$ donne l'orientation $s(x) + 0 > s(s(x))$ et $s(s(x)) > s(x)$. En d'autres termes, les prédicats $\asymp$ et $>$ ne sont pas cohérents au sens de [JK86]. Pour ne pas avoir à les rendre cohérents, ce qui serait une limitation importante, on n'autorise pas le prédicat d'égalité $\asymp$ à agir sous le prédicat $>$ en faisant de ce prédicat un symbole protecteur.

On va tenir compte de ces symboles protecteurs de la même façon dans la théorie de l'égalité et dans les relations définies par un système de réécriture de classes conditionnel, afin d'obtenir la correspondance entre les deux au chapitre 3.

2.2.2 Théorie de l'égalité

On étend la définition de l'égalité pour tenir compte des symboles protecteurs :

Définition 2.19 Soit $\asymp$ un prédicat binaire, et soient $\mathcal{F}_{p,\asymp}$ et $\mathcal{P}_{p,\asymp}$ des sous-ensembles respectivement de $\mathcal{F}$ et de $\mathcal{P}$.

On note $\mathcal{F}_{\bar{p},\asymp} = \mathcal{F} \setminus \mathcal{F}_{p,\asymp}$ et $\mathcal{P}_{\bar{p},\asymp} = \mathcal{P} \setminus \mathcal{P}_{p,\asymp}$.

$\asymp$ est un prédicat d'égalité dont les symboles protecteurs sont les éléments de $\mathcal{F}$ et $\mathcal{P}$ s'il satisfait la théorie $Th_{\asymp}$ formée des axiomes suivants :

1. $\forall x\,(x \asymp x)$
2. $\forall x\ \forall y\,(x \asymp y \Rightarrow y \asymp x)$
3. $\forall x\ \forall y\ \forall z\,((x \asymp y \wedge y \asymp z) \Rightarrow x \asymp z)$
4. pour tout f d'arité n dans $\mathcal{F}$, pour tout $i \in [1 \ldots n]$,
   $\forall \overline{u}, x_i, \overline{v}, y_i\,(x_i \asymp y_i \Rightarrow (f(\overline{u}, x_i, \overline{v}) \asymp f(\overline{u}, y_i, \overline{v})))$
   si et seulement si f est dans $\mathcal{F}_{\bar{p},\asymp}$.
5. pour tout Q d'arité n dans $\mathcal{P}$, pour tout $i \in [1 \ldots n]$,
   $\forall \overline{u}, x_i, \overline{v}, y_i\,(x_i \asymp y_i \Rightarrow (Q(\overline{u}, x_i, \overline{v}) \Leftrightarrow Q(\overline{u}, y_i, \overline{v})))$
   si et seulement si Q est dans $\mathcal{P}_{\bar{p},\asymp}$.

Remarque 2.10 Les axiomes 1, 2 et 3 expriment respectivement la réflexivité, la symétrie et la transitivité.

Remarque 2.11 On peut définir $Th_{\asymp}$ de plusieurs façons équivalentes.

- Il est facile de démontrer, grâce à la transitivité et à la réflexivité que les axiomes 4 et 5 sont équivalents aux axiomes suivants :

  4'. pour tout f dans $\mathcal{F}$,
  $\forall \overline{x}, \overline{y}\,(\overline{x} \asymp \overline{y} \Rightarrow (f(\overline{x}) \asymp f(\overline{y})))$
  si et seulement si f est dans $\mathcal{F}_{\bar{p},\asymp}$.

5'. pour tout Q dans $\mathcal{P}$,
$\forall \overline{x}, \overline{y}\,(\overline{x} \asymp \overline{y} \Rightarrow (Q(\overline{x}) \Leftrightarrow Q(\overline{y})))$
si et seulement si Q est dans $\mathcal{P}_{\overline{p},\asymp}$.

- on peut aussi dans 5 et 5' utiliser un $\Rightarrow$ au lieu d'un $\Leftrightarrow$.

5''. pour tout Q dans $\mathcal{P}$,
$\forall \overline{x}, \overline{y}\,((\overline{x} \asymp \overline{y} \wedge Q(\overline{x})) \Rightarrow Q(\overline{y}))$
si et seulement si Q est dans $\mathcal{P}_{\overline{p},\asymp}$.

- la symétrie se déduit de 5'' et de la réflexivité, puis la transitivité se déduit de 5'' et de la symétrie, avec $\asymp$ pour Q dans les deux cas.

Remarque 2.12 Un prédicat d'égalité ne peut pas être un symbole protecteur pour lui-même. En effet, 5 appliqué à $\asymp$ se déduit de la symétrie et de la transitivité. On a donc nécessairement $\asymp \in \mathcal{P}_{\overline{p},\asymp}$, et on ne le mentionnera plus explicitement par la suite.

On peut noter que dans le cas des prédicats d'égalité, la notion de cohérence définie dans [PS81, JK86, Vir02] permet d'assurer la compatibilité de théories équationnelles. Néanmoins on ne souhaite pas ici se limiter à des théories compatibles. On peut également noter que la notion de "symbole gelé" introduite très récemment en logique de réécriture [BMM02] est similaire à notre notion de symbole protecteur.

Remarque 2.13 Dans les définitions précédentes, soit un symbole n'est pas protecteur, soit il l'est pour tous ses arguments. Une extension possible est de considérer des symboles protégeant seulement certains de leurs arguments.

Remarque 2.14 Il est important que les symboles d'égalité soient protecteurs les uns par rapport aux autres, en effet, si $\asymp_1$ n'est pas protecteur pour $\asymp_2$, et que $a \asymp_2 b$, on a $a \asymp_1 a$ par réflexivité de $\asymp_1$ et par 5 pour $\asymp_2$ avec $Q = \asymp_1$:

$$a \asymp_2 b \Rightarrow (a \asymp_1 a \Leftrightarrow a \asymp_1 b)$$

d'où $a \asymp_1 b$, par symétrie et transitivité de $\asymp_1$.

On obtient ainsi que $\asymp_1$ contient $\asymp_2$. Si réciproquement $\asymp_2$ n'est pas protecteur pour $\asymp_1$, alors les deux égalités se confondent. C'est ainsi que dans un cadre classique, sans symboles protecteurs, deux prédicats d'égalité sont toujours identiques.

Il faut donc rendre $\asymp_1$ protecteur pour $\asymp_2$ pour pouvoir rendre un symbole protecteur pour $\asymp_2$ et non protecteur pour $\asymp_1$.

La théorie de l'égalité sera considérée dans les deux contextes suivants :

1. l'identité, qui vérifie de plus
   - $\forall \overline{x}, \overline{y}\,((f(\overline{x}) \asymp f(\overline{y})) \Rightarrow \overline{x} \asymp \overline{y})$
     pour tout f dans $\mathcal{F}$.
   - $\forall \overline{x}, \overline{y}\,(\neg f(\overline{x}) \asymp g(\overline{y}))$
     pour tous f et g dans $\mathcal{F}$ tels que $f \neq g$.
2. les égalités impliquées par des axiomes de la forme $t \asymp t'$.

Définition 2.20 On définit le prédicat :=: comme un prédicat d'identité qui n'a de plus aucun symbole protecteur.

Ce prédicat dénote une égalité syntaxique sur les termes et sera utilisé pour les définitions de types abstraits. On note $Th_{:=:}$ sa théorie.

Définition 2.21 On définit le prédicat $\approx$ comme un prédicat d'égalité tel que :=: $\in \mathcal{P}_{p,\approx}$ (:=: est protecteur pour $\approx$), ce qui signifie que l'égalité définie par $\approx$ ne pourra pas transformer une égalité avec :=:. En effet, aucun symbole n'étant protecteur pour :=:, il est nécessaire comme expliqué dans la remarque 2.14 que :=: soit protecteur pour tout prédicat d'égalité pour lequel on souhaite avoir des symboles protecteurs.

On utilisera ce prédicat pour les théories équationnelles. On note $Th_{\approx}$ sa théorie.

Exemple 2.11 Soit $E = \{a \asymp b, f(x) \asymp g(x)\}$.

L'égalité générée par E est $\asymp_E$ et contient $a \asymp_E b$, $f(a) \asymp_E g(a)$, $f(b) \asymp_E g(b)$, $f(a) \asymp_E f(b)$, $g(a) \asymp_E g(b)$, $f(a) \asymp_E g(b)$, $f(b) \asymp_E g(a)$, ...

2.2.3 Systèmes de réécriture de classes conditionnels

2.2.3.1 Définitions

Définition 2.22 Une *règle de réécriture conditionnelle de termes* est une paire de termes l, r avec une proposition c, notée $l \to r$ si c.

Les variables de c et r doivent apparaître dans l.

c représente une condition c et peut être absent.

Exemple 2.12 Des règles de réécriture comme $x + 0 \to x$ et $l_1.x.l_2.y.l_3 \to l_1.y.l_2.x.l_3$ si $y > x$ font respectivement partie de la théorie des groupes et d'un algorithme de tri de listes.

Définition 2.23 Un *axiome conditionnel de termes* est une paire de termes f, g avec une proposition c, noté $f \asymp g$ si c.

Les variables de f et de g doivent être les mêmes, et les variables de c doivent apparaître dans f et g.

c représente une condition c et peut être absent.

Exemple 2.13 Des exemples d'axiomes équationnels de termes sont $x + y \asymp y + x$ et $x.(y.z) \asymp (x.y).z$ qui font respectivement partie de la théorie des groupes Abéliens et de la théorie des listes.

Définition 2.24 Une *règle de réécriture conditionnelle de propositions* est un triplet de propositions c, l, r où l est atomique et c, r sont arbitraires, noté $l \to r$ si c.

Les variables libres de c et r doivent apparaître dans l.

c représente une condition c et peut être absent.

Exemple 2.14 Un exemple de règle de réécriture de propositions est $x \in \mathcal{P}(y) \to \forall z\ (z \in x \Rightarrow z \in y)$ qui décrit l'ensemble des parties en théorie des ensembles.

Définition 2.25 Un *axiome conditionnel de propositions* est une paire de propositions atomiques l, r avec une proposition c, noté $l \asymp r$ si c.

Les variables de f et de g doivent être les mêmes, et les variables libres de c doivent apparaître dans f et g.

c représente une condition c et peut être absent.

Exemple 2.15 Un exemple d'axiome équationnel de propositions est la commutativité d'un symbole d'égalité $\sim$: $(x \sim y) \asymp (y \sim x)$

Définition 2.26 Un *système de réécriture de classes conditionnel* est une paire, notée $\mathcal{RE}$, composée de :
- $\mathcal{R}$: un ensemble de règles de réécriture conditionnelles de propositions ou de termes,
- $\mathcal{E}$: un ensemble d'axiomes équationnels de propositions ou de termes.

2.2.3.2 Théorie canonique associée à un système de réécriture de classes conditionnel

Définition 2.27 A un système de réécriture de classes conditionnel $\mathcal{RE}$, on associe la théorie notée $T_{\mathcal{RE}}$ telle que pour chaque règle de réécriture conditionnelle $l \to r$ si c ou axiome équationnel $l \asymp r$ si c, $T_{\mathcal{RE}}$ contient la proposition :
- $\forall \overline{x}(c \Rightarrow (l \Leftrightarrow r))$ si l et r sont des propositions,
- $\forall \overline{x}(c \Rightarrow (l \asymp r))$ si l et r sont des termes.

$T_{\mathcal{RE}}$ est la contrepartie logique de $\mathcal{RE}$, c'est-à-dire que la déduction en présence de $T_{\mathcal{RE}}$ est équivalente à la déduction modulo $\mathcal{RE}$ (voir lemme 3.2, section 3.4).

2.2.3.3 Relations définies par un système de réécriture de classes conditionnel

Définition 2.28 Soit t un terme, on définit $Prot(t, \omega, \asymp)$ comme étant vrai si et seulement si la position ω est sous un symbole protecteur (pour $\asymp$), c'est-à-dire si un symbole protecteur (pour $\asymp$) apparaît dans le terme t à une position préfixe de ω.

Définition 2.29 Soit P une proposition, on définit de la même façon $Prot(P, \omega, \asymp)$.

Définition 2.30 Etant donné un système de réécriture de classes conditionnel $\mathcal{RE}$, un ensemble de propositions Γ, des propositions P, P', Q, Q' et c, une occurrence ω non protégée pour $\asymp$ dans P (c'est-à-dire telle que $Prot(P, \omega, \asymp)$ est faux), et une substitution σ telle que $T_{\mathcal{RE}}, \Gamma \vdash \sigma(c)$, on définit les relations suivantes :

1. $P \longleftrightarrow^{\Gamma}_{\mathcal{E}} P'$, si $P' = P[\sigma(r)]_\omega$, pour un axiome équationnel $l \asymp r$ si c ou $r \asymp l$ si c dans $\mathcal{E}$ tel que $\sigma(l) = P_{|\omega}$. On dit que P est *$\mathcal{E}$-équivalent* à P' dans le contexte Γ.
2. $P =^{\Gamma}_{\mathcal{E}} P'$ est l'équivalence générée par $\mathcal{E}$ pour un contexte Γ, c'est-à-dire la clôture réflexive et transitive de $\longleftrightarrow^{\Gamma}_{\mathcal{E}}$.
3. $P \longrightarrow^{\Gamma}_{\mathcal{R}} P'$, si $P' = P[\sigma(r)]_\omega$, pour une règle de réécriture $l \to r$ si c dans $\mathcal{R}$ telle que $\sigma(l) = P_{|\omega}$. On dit que P se *$\mathcal{R}$-réécrit* en P' dans le contexte Γ.
4. $P \rightsquigarrow^{\Gamma}_{\mathcal{R}} P'$, si $P' = \sigma(P[r]_\omega)$, pour une règle de réécriture $l \to r$ si c dans $\mathcal{R}$ telle que $\sigma(l) = \sigma(P_{|\omega})$. On dit que P se *$\mathcal{R}$-surréduit* en P' dans le contexte Γ.
5. $Q \longrightarrow^{\Gamma}_{\mathcal{RE}} Q'$, si $Q =^{\Gamma}_{\mathcal{E}} P[\sigma(l)]_\omega$, $Q' =^{\Gamma}_{\mathcal{E}} P[\sigma(r)]_\omega$, pour une règle $l \to r$ si c dans $\mathcal{R}$. On dit que Q se *$\mathcal{RE}$-réécrit* en Q' dans le contexte Γ.
6. $P \longrightarrow^{\Gamma}_{\mathcal{R},\mathcal{E}} P'$, si $P' = P[\sigma(r)]_\omega$, pour une règle $l \to r$ si $c \in \mathcal{R}$ telle que $\sigma(l) =^{\Gamma}_{\mathcal{E}} P_{|\omega}$. On dit que P se *$\mathcal{R}, \mathcal{E}$-réécrit* en P' dans le contexte Γ. Ceci est la relation classique adaptée de Peterson et Stickel [PS81] où elle était définie uniquement pour des termes.
7. $=^{\Gamma}_{\mathcal{RE}}$ est la congruence générée par $\mathcal{R} \cup \mathcal{E}$ dans un contexte Γ, donc la clôture symétrique, réflexive et transitive de $\longleftrightarrow^{\Gamma}_{\mathcal{E}} \cup \longrightarrow^{\Gamma}_{\mathcal{R}}$.

Notation 2.4 Quand le contexte Γ est soit vide soit clair d'après le contexte on l'omet.

Notation 2.5 La clôture réflexive et transitive d'une relation $\longrightarrow$ est notée $\xrightarrow{*}$.

Notation 2.6 On note $\longleftrightarrow_{\mathcal{R}}$ la clôture symétrique de $\longrightarrow_{\mathcal{R}}$.

Notation 2.7 On note $\longleftrightarrow_{\mathcal{R}\cup\mathcal{E}} = \longleftrightarrow_{\mathcal{R}} \cup \longleftrightarrow_{\mathcal{E}}$.

La relation $=^{\Gamma}_{\mathcal{RE}}$ n'est pas décidable en général, notamment puisqu'on doit décider la validité de la condition pour appliquer une règle conditionnelle. Cependant plusieurs restrictions de la définition générale de réécriture conditionnelle, comme la réécriture non équationnelle et non conditionnelle avec un système noethérien, la rendent décidable. Des restrictions plus intéressantes comme les systèmes de réécriture décroissants sont par exemple décrits dans [DO90].

Chapitre 3

Déduction modulo

La déduction modulo [DHK98] permet, en raisonnant modulo une congruence, de distinguer la partie *calcul* d'un raisonnement de sa partie *déduction*. Cela permet de mettre en évidence les étapes importantes du raisonnement en "masquant" le calcul.

On présente d'abord la déduction modulo indépendamment de la façon de représenter la congruence. La congruence peut ici tenir compte du contexte dans lequel elle est appliquée, ce qui est une extension par rapport à [DHK98] et [Dow99].

On présente ensuite la représentation par système de réécriture de classes conditionnel qu'on utilisera pour la suite. Le caractère conditionnel des règles et des équations, ainsi que la prise en compte par les systèmes de réécriture de classes conditionnels de symboles protecteurs (voir chapitre 2) sont de nouvelles extensions au formalisme.

3.1 Introduction

Prouver par une *déduction* dans l'arithmétique de Peano que $2*2 = 4$ est relativement long et difficile, alors que le vérifier par un *calcul* est trivial. Comme le note Gilles Dowek dans sa thèse d'habilitation [Dow99], ce fait avait déjà été remarqué par Henri Poincaré dans [Poi02]. Il est la motivation principale de la déduction modulo.

La *déduction modulo* [DHK98] est une présentation de la logique du premier ordre, dans laquelle des termes mais aussi des propositions peuvent être rendus équivalents par une congruence. Les notions de terme et de proposition sont celles de la logique (multisortée) du premier ordre.

Quand on applique une règle de déduction on utilise un filtrage modulo cette congruence. Donc si on a par exemple $A \equiv B$ par la congruence, on est capable d'appliquer n'importe quelle règle utilisant A sur une entrée contenant en fait B. Par exemple si on a $2 * 2 \equiv 4$ alors une preuve de

$$\underline{A}, 4 = 4 \vdash 2 * 2 = 4, \underline{B}$$

est simplement

$$\frac{}{\underline{A}, 4 = 4 \vdash 2 * 2 = 4, \underline{B}}\ \text{axiome}$$

On considère une congruence C définie sur les propositions et capable de tenir compte de l'ensemble d'hypothèses Γ présent dans le contexte, on la note $=_C^\Gamma$. Les règles de déduction ont a prendre en compte cette congruence. Par exemple, la règle à droite pour la conjonction n'est pas formulée

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

mais

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash D, \Delta}\ \text{si } D =_C^\Gamma A \wedge B.$$

Les preuves en déduction modulo sont beaucoup plus courtes que leurs équivalents en déduction classique, puisque les calculs représentés par la congruence sont masqués. Cela permet de se focaliser sur les étapes importantes de la déduction.

3.2 Calcul des séquents modulo

Le *calcul des séquents modulo* (figure 3.1) étend le calcul classique [GLT89, Gal86] en permettant de travailler modulo une congruence C. En fait cette congruence est un peu plus générale ici que dans [DHK98], où elle ne tient pas compte du contexte Γ. Dans ces règles, Γ et Δ sont des multiensembles finis de propositions. Quand la congruence $=_C^\Gamma$ est simplement l'identité, ce calcul des séquents se réduit au calcul classique. Dans ce cas les séquents sont notés avec le symbole $\vdash$ habituel.

$$\frac{}{\Gamma, P \vdash_C Q}\text{axiome si } P =_C^\Gamma Q \qquad \frac{\Gamma, P \vdash_C \Delta \quad \Gamma \vdash_C Q, \Delta}{\Gamma \vdash_C \Delta}\text{cut si } P =_C^\Gamma Q$$

$$\frac{\Gamma, Q_1, Q_2 \vdash_C \Delta}{\Gamma, P \vdash_C \Delta}\text{contraction-g si } P =_C^\Gamma Q_1 =_C^\Gamma Q_2 \qquad \frac{\Gamma \vdash_C Q_1, Q_2, \Delta}{\Gamma \vdash_C P, \Delta}\text{contraction-d si } P =_C^\Gamma Q_1 =_C^\Gamma Q_2$$

$$\frac{\Gamma \vdash_C \Delta}{\Gamma, P \vdash_C \Delta}\text{affaiblissement-g} \qquad \frac{\Gamma \vdash_C \Delta}{\Gamma \vdash_C P, \Delta}\text{affaiblissement-d}$$

$$\frac{\Gamma, P, Q \vdash_C \Delta}{\Gamma, R \vdash_C \Delta}\wedge\text{-g si } R =_C^\Gamma (P \wedge Q) \qquad \frac{\Gamma \vdash_C P, \Delta \quad \Gamma \vdash_C Q, \Delta}{\Gamma \vdash_C R, \Delta}\wedge\text{-d si } R =_C^\Gamma (P \wedge Q)$$

$$\frac{\Gamma, P \vdash_C \Delta \quad \Gamma, Q \vdash_C \Delta}{\Gamma, R \vdash_C \Delta}\vee\text{-g si } R =_C^\Gamma (P \vee Q) \qquad \frac{\Gamma \vdash_C P, Q, \Delta}{\Gamma \vdash_C R, \Delta}\vee\text{-d si } R =_C^\Gamma (P \vee Q)$$

$$\frac{\Gamma \vdash_C P, \Delta \quad \Gamma, Q \vdash_C \Delta}{\Gamma, R \vdash_C \Delta}\Rightarrow\text{-g si } R =_C^\Gamma (P \Rightarrow Q) \qquad \frac{\Gamma, P \vdash_C Q, \Delta}{\Gamma \vdash_C R, \Delta}\Rightarrow\text{-d si } R =_C^\Gamma (P \Rightarrow Q)$$

$$\frac{\Gamma \vdash_C P, \Delta}{\Gamma, R \vdash_C \Delta}\neg\text{-g si } R =_C^\Gamma \neg P \qquad \frac{\Gamma, P \vdash_C \Delta}{\Gamma \vdash_C R, \Delta}\neg\text{-d si } R =_C^\Gamma \neg P$$

$$\frac{}{\Gamma, P \vdash_C \Delta}\bot\text{-g si } P =_C^\Gamma \bot$$

$$\frac{\Gamma, Q\{t/x\} \vdash_C \Delta}{\Gamma, P \vdash_C \Delta}(Q, x, t)\ \forall\text{-g si } P =_C^\Gamma \forall x\, Q \qquad \frac{\Gamma \vdash_C Q\{y/x\}, \Delta}{\Gamma \vdash_C P, \Delta}(Q, x, y)\ \forall\text{-d si } P =_C^\Gamma \forall x\, Q$$

$$\frac{\Gamma, Q\{y/x\} \vdash_C \Delta}{\Gamma, P \vdash_C \Delta}(Q, x, y)\ \exists\text{-g si } P =_C^\Gamma \exists x\, Q \qquad \frac{\Gamma \vdash_C Q\{t/x\}, \Delta}{\Gamma \vdash_C P, \Delta}(Q, x, t)\ \exists\text{-d si } P =_C^\Gamma \exists x\, Q$$

Dans $\forall$-d et $\exists$-r, y doit être une nouvelle variable libre.
Dans $\forall$-r et $\exists$-d, t est un terme quelconque.

FIG. 3.1 – Le calcul des séquents modulo

La décidabilité de la vérification de preuve pour le calcul des séquents modulo se réduit à la décidabilité de la relation $=_C^\Gamma$, puisqu'on peut vérifier pour chaque règle que les conditions d'application sont satisfaites et qu'on fournit les informations nécessaires dans les règles pour les quantificateurs. Dans le cas où $=_C^\Gamma$ n'est pas décidable, on pourra néanmoins utiliser des instances où il est "facile" de vérifier que les conditions d'application sont satisfaites.

3.3 Lien entre déduction modulo et déduction classique

On l'a dit, les preuves en déduction modulo sont plus courtes que leurs équivalents en déduction classique, mais on peut aller plus loin et les relier par une équivalence. Ainsi une preuve en déduction modulo n'est qu'une forme condensée d'une preuve en déduction classique qu'on est capable de reconstituer si nécessaire. Pour cela, on définit la notion de compatibilité entre un ensemble d'axiomes et une congruence.

Définition 3.1 Un ensemble d'axiomes $\mathcal{T}$ et une congruence C sont *compatibles sous* un ensemble d'axiomes $\mathcal{U}$ si :

A. pour toutes propositions P et Q et contexte Γ, $P =_C^{\mathcal{U},\Gamma} Q$ implique $\mathcal{T},\mathcal{U},\Gamma \vdash P \Leftrightarrow Q$ et,

B. pour toute proposition P dans $\mathcal{T}$, on a $\mathcal{U} \vdash_C P$.

Notation 3.1 Une congruence C compatible avec une théorie $\mathcal{T}$ sous une théorie $\mathcal{U}$ sera notée $C_{\mathcal{T}(\mathcal{U})}$.

Sous l'hypothèse de compatibilité, on accède ainsi à une modularisation puissante du processus de déduction qui permet de "déduire modulo" et qui est liée au "principe de Poincaré" de [BB97] :

Proposition 3.1 Si la théorie $\mathcal{T}$ et la congruence C sont compatibles sous une théorie $\mathcal{U}$ on a :

$$\mathcal{T},\mathcal{U},\Gamma \vdash \Delta \text{ si et seulement si } \mathcal{T},\mathcal{U},\Gamma \vdash_C \Delta.$$

Preuve: La preuve est similaire à celle présentée dans [DHK98].

- La partie "seulement si" est une simple récurrence sur la structure de la dérivation de $\mathcal{T},\mathcal{U},\Gamma \vdash \Delta$, explicitant les témoins en utilisant le filtrage du premier ordre.
- La partie "si" est une récurrence sur la structure de la dérivation de $\mathcal{T},\mathcal{U},\Gamma \vdash_C \Delta$.
  Notons d'abord qu'en utilisant la règle de contraction toute preuve de ce jugement peut être transformée en une autre où les propositions de $\mathcal{T}$ apparaissent en partie gauche de chaque séquent.
  Par exemple, détaillons le cas où la dernière règle appliquée est $\wedge$-d.
  Dans ce cas, la preuve est de la forme :

$$\dfrac{\dfrac{\pi}{\mathcal{T},\mathcal{U},\Gamma \vdash_C P,\Delta} \quad \dfrac{\rho}{\mathcal{T},\mathcal{U},\Gamma \vdash_C Q,\Delta}}{\mathcal{T},\mathcal{U},\Gamma \vdash_C R,\Delta}\ \wedge\text{-d} \quad \text{où } R =_C^{\mathcal{T},\mathcal{U},\Gamma} P \wedge Q$$

  Par hypothèse de récurrence on a des preuves π' et ρ' de $\mathcal{T},\mathcal{U},\Gamma \vdash P,\Delta$ et $\mathcal{T},\mathcal{U},\Gamma \vdash Q,\Delta$.
  On construit la preuve :

$$\dfrac{\dfrac{\pi'}{\mathcal{T},\mathcal{U},\Gamma \vdash P,\Delta} \quad \dfrac{\rho'}{\mathcal{T},\mathcal{U},\Gamma \vdash Q,\Delta}}{\mathcal{T},\mathcal{U},\Gamma \vdash P \wedge Q,\Delta}\ \wedge\text{-d}$$

  On a $R =_C^{\mathcal{T},\mathcal{U},\Gamma} P \wedge Q$, donc puisque $\mathcal{T}$ et C sont compatibles (sous $\mathcal{U}$), $\mathcal{T},\mathcal{U},\Gamma \vdash (P \wedge Q) \Rightarrow R$.
  Le *modus ponens* est une règle dérivée du calcul de séquents puisque :

$$\dfrac{\dfrac{\dfrac{}{\mathcal{U},\Gamma,B \vdash B,\delta}\ \text{axiome} \quad \dfrac{\dfrac{\cdots}{\mathcal{U},\Gamma \vdash A,\delta}}{\mathcal{U},\Gamma \vdash A,B,\delta}\ \text{affaiblissement-d}}{\mathcal{U},\Gamma,A \Rightarrow B \vdash B,\delta}\ \Rightarrow\text{-g} \quad \dfrac{\cdots}{\mathcal{U},\Gamma \vdash A \Rightarrow B,\delta}}{\mathcal{U},\Gamma \vdash B,\delta}\ \begin{matrix}\text{affaiblissement-d}\\ \text{coupure}\end{matrix}$$

  Donc on peut construire une preuve de $\mathcal{T},\mathcal{U},\Gamma \vdash R,\Delta$. □

Corollaire 3.1 Si la théorie $\mathcal{T}$ et la congruence C sont compatibles sous une théorie $\mathcal{U}$ on a :

$$\mathcal{T},\mathcal{U},\Gamma \vdash \Delta \text{ si et seulement si } \mathcal{T},\Gamma \vdash_C \Delta.$$

Preuve: En partant de la proposition précédente, il nous reste à montrer que

$$\mathcal{T}, \mathcal{U}, \Gamma \vdash_C \Delta \text{ si et seulement si } \mathcal{U}, \Gamma \vdash_C \Delta.$$

Or puisque $\mathcal{T}$ et C sont compatibles sous $\mathcal{U}$, on a $\mathcal{U} \vdash_C P$ pour toute proposition P dans $\mathcal{T}$, ce qui nous permet de conclure.

□

Sous l'hypothèse de compatibilité, la proposition précédente permet d'internaliser dans la congruence une partie du processus de déduction. Quand la congruence est décidable, vérifier l'équivalence de deux termes ou de deux propositions est juste un calcul. Cela signifie que la déduction modulo permet de tracer une séparation entre la déduction (en général indécidable) et le calcul (une partie décidable du processus de raisonnement). Un des principaux avantages de cette approche est qu'en passant d'une preuve dans le calcul des séquents standard à une preuve dans le calculs des séquents modulo, la partie calcul de la preuve devient invisible puisqu'elle est enfermée dans la congruence.

Appliquer la direction : si $\mathcal{T}, \Gamma \vdash \Delta$ alors $\Gamma \vdash_C \Delta$ est appelé *pousser* ou *internaliser* (parfois aussi *intégrer*) la théorie $\mathcal{T}$ dans la définition du séquent et permet de faire disparaître la théorie au niveau de la déduction pour la faire passer au niveau du calcul. La direction opposée est appelée *retirer* la théorie $\mathcal{T}$ de la définition du séquent ou *externaliser* la théorie $\mathcal{T}$ et permet de faire réapparaître la théorie au niveau de la déduction. Cette opération peut même être effectuée de façon incrémentale puisqu'elle a la bonne propriété d'être modulaire comme démontré par les résultats suivants :

Définition 3.2 Soient C_1 et C_2 deux congruences.

La congruence $C_1 \cup C_2$ est définie récursivement par $a =_{C_1 \cup C_2} b$ si :

- soit $a =_{C_1} b$,
- soit $a =_{C_2} b$,
- soit il existe c tel que $a =_{C_1} c =_{C_1 \cup C_2} b$,
- soit il existe c tel que $a =_{C_2} c =_{C_1 \cup C_2} b$.

Lemme 3.1 Soient $\mathcal{T}_1$ et $\mathcal{T}_2$ deux ensembles d'axiomes et $C_{\mathcal{T}_1(\mathcal{U}_1)}$ et $C_{\mathcal{T}_2(\mathcal{U}_2)}$ des congruences compatibles respectivement avec $\mathcal{T}_1$ et $\mathcal{T}_2$ sous les ensembles d'axiomes $\mathcal{U}_1$ et $\mathcal{U}_2$.

La congruence $C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2)}$ est compatible avec $\mathcal{T}_1 \cup \mathcal{T}_2$ sous $\mathcal{U}_1 \cup \mathcal{U}_2$.

Preuve: On doit prouver que

A. pour toutes propositions P et Q et contextes Γ,
$P =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2)}} Q$ implique $\mathcal{T}_1, \mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash P \Leftrightarrow Q$.
On procède suivant les cas de la définition 3.2.
   - Si $P =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_1(\mathcal{U}_1)}} Q$, par la compatibilité de $\mathcal{T}_1$ et $C_{\mathcal{T}_1(\mathcal{U}_1)}$ sous $\mathcal{U}_1$, on obtient que pour tout Γ', $P =^{\mathcal{U}_1,\Gamma'}_{C_{\mathcal{T}_1(\mathcal{U}_1)}} Q$ implique $\mathcal{T}_1, \mathcal{U}_1, \Gamma' \vdash P \Leftrightarrow Q$.
     Donc en particulier si $\Gamma' = \mathcal{U}_2, \Gamma$ on obtient que $P =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_1(\mathcal{U}_1)}} Q$ implique $\mathcal{T}_1, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash P \Leftrightarrow Q$.
     La monotonie de la logique nous permet d'ajouter $\mathcal{T}_2$ et donc de conclure.
   - Si $P =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_2(\mathcal{U}_2)}} Q$, par la compatibilité de $\mathcal{T}_2$ et $C_{\mathcal{T}_2(\mathcal{U}_2)}$ sous $\mathcal{U}_2$, on obtient que pour tout Γ', $P =^{\mathcal{U}_2,\Gamma'}_{C_{\mathcal{T}_2(\mathcal{U}_2)}} Q$ implique $\mathcal{T}_2, \mathcal{U}_2, \Gamma' \vdash P \Leftrightarrow Q$.
     Donc en particulier si $\Gamma' = \mathcal{U}_1, \Gamma$ on obtient que $P =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_2(\mathcal{U}_2)}} Q$ implique $\mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash P \Leftrightarrow Q$.
     La monotonie de la logique nous permet d'ajouter $\mathcal{T}_1$ et donc de conclure.
   - si $P =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_1(\mathcal{U}_1)}} R =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2)}} Q$, on a $\mathcal{T}_1, \mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash P \Leftrightarrow R$ et par hypothèse de récurrence $\mathcal{T}_1, \mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash R \Leftrightarrow Q$ d'où on déduit facilement $\mathcal{T}_1, \mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash P \Leftrightarrow Q$.
   - si $P =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_2(\mathcal{U}_2)}} R =^{\mathcal{U}_1,\mathcal{U}_2,\Gamma}_{C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2)}} Q$, on a $\mathcal{T}_1, \mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash P \Leftrightarrow R$ et par hypothèse de récurrence $\mathcal{T}_1, \mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash R \Leftrightarrow Q$ d'où on déduit facilement $\mathcal{T}_1, \mathcal{T}_2, \mathcal{U}_1, \mathcal{U}_2, \Gamma \vdash P \Leftrightarrow Q$.

B. Pour chaque proposition P dans $\mathcal{T}_1 \cup \mathcal{T}_2$ on a $\mathcal{U}_1, \mathcal{U}_2 \vdash_{C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2)}} P$.
Deux cas sont possibles :
   - soit P est dans $\mathcal{T}_1$ et on a $\mathcal{U}_1 \vdash_{C_{\mathcal{T}_1(\mathcal{U}_1)}} P$, donc $\mathcal{U}_1, \mathcal{U}_2 \vdash_{C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2)}} P$,
   - soit P est dans $\mathcal{T}_2$ et on a $\mathcal{U}_2 \vdash_{C_{\mathcal{T}_2(\mathcal{U}_2)}} P$, donc $\mathcal{U}_1, \mathcal{U}_2 \vdash_{C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2)}} P$.

□

Par le lemme précédent et le corollaire 3.1, on obtient immédiatement la modularité :

Corollaire 3.2 Soient $\mathcal{T}_1$ et $\mathcal{T}_2$ deux ensembles d'axiomes et $C_{\mathcal{T}_1(\mathcal{U}_1)}$ et $C_{\mathcal{T}_2(\mathcal{U}_2)}$ des congruences compatibles respectivement avec $\mathcal{T}_1$ et $\mathcal{T}_2$ sous les ensembles d'axiomes $\mathcal{U}_1$ et $\mathcal{U}_2$. Soit $C_{\mathcal{T}_1\cup\mathcal{T}_2(\mathcal{U}_1\cup\mathcal{U}_2)}$ une congruence compatible avec $\mathcal{T}_1 \cup \mathcal{T}_2$ sous $\mathcal{U}_1 \cup \mathcal{U}_2$.

Pour tout Γ,

$$\begin{array}{c} \mathcal{U}_1,\mathcal{U}_2,\Gamma \vdash_{C_{\mathcal{T}_1\cup\mathcal{T}_2(\mathcal{U}_1\cup\mathcal{U}_2)}} \Delta \\ \Updownarrow \\ \mathcal{T}_2,\mathcal{U}_1,\mathcal{U}_2,\Gamma \vdash_{C_{\mathcal{T}_1(\mathcal{U}_1)}} \Delta \Leftrightarrow \mathcal{T}_1,\mathcal{T}_2,\mathcal{U}_1,\mathcal{U}_2,\Gamma \vdash \Delta \Leftrightarrow \mathcal{T}_1,\mathcal{U}_1,\mathcal{U}_2,\Gamma \vdash_{C_{\mathcal{T}_2(\mathcal{U}_2)}} \Delta \\ \Updownarrow \\ \mathcal{U}_1,\mathcal{U}_2,\Gamma \vdash_{C_{\mathcal{T}_1(\mathcal{U}_1)}\cup C_{\mathcal{T}_2(\mathcal{U}_2)}} \Delta \end{array}$$

Remarque 3.1 La modularité permet de choisir d'internaliser ou d'externaliser $\mathcal{T}_1$ ou $\mathcal{T}_2$ ou les deux, ou encore l'un puis l'autre de la façon qui convient le mieux. Elle permet aussi de prendre simplement $C_{\mathcal{T}_1(\mathcal{U}_1)} \cup C_{\mathcal{T}_2(\mathcal{U}_2}$ pour $C_{\mathcal{T}_1\cup\mathcal{T}_2(\mathcal{U}_1\cup\mathcal{U}_2)}$. Elle sera donc très utile dans le traitement de la récurrence.

Remarque 3.2 Bien que trouver une congruence correspondant à un ensemble d'axiomes soit une opération modulaire, il n'en sera pas forcément de même au niveau des représentations. En effet, l'union des représentations des congruences ne sera pas forcément une représentation de l'union des congruences.

Si, dans la représentation choisie, pour toute congruence C il existe un ensemble d'axiomes $\mathcal{T}_{C(\mathcal{U})}$ compatible avec C sous l'ensemble d'axiomes $\mathcal{U}$, alors l'hypothèse de compatibilité n'est pas une restriction et la déduction modulo n'est pas une extension propre de la logique du premier ordre [DHK98]. En effet pour toute congruence $=_C^{\mathcal{U},\Gamma}$, on peut trouver une théorie $\mathcal{T}$ telle que $\mathcal{U},\Gamma \vdash P$ est prouvable modulo $=_C^{\mathcal{U},\Gamma}$ si et seulement si $\mathcal{T},\mathcal{U},\Gamma \vdash P$ est prouvable en logique du premier ordre classique. Bien sûr, les propositions prouvables sont les mêmes, mais les preuves sont très différentes, en fait beaucoup plus courtes en déduction modulo.

3.4 Congruence représentée par un système de réécriture de classes conditionnel

On montre ici que pour tout système de réécriture de classes conditionnel $\mathcal{RE}$, il existe un ensemble d'axiomes $\mathcal{T}$ tel que $\mathcal{T}$ et $\mathcal{RE}$ sont compatibles sous la théorie de l'égalité :

Lemme 3.2 Pour tout système de réécriture de classes conditionnel $\mathcal{RE}$, la théorie $\mathcal{T}_{\mathcal{RE}}$ de la définition 2.27 est compatible avec $\mathcal{RE}$ sous $Th_\asymp$.

Preuve: On doit prouver que :

A. pour toutes propositions P et Q et tous contextes Γ tels que $P =_{\mathcal{RE}}^{Th_\asymp,\Gamma} Q$, on a $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P \Leftrightarrow Q$.

On procède par récurrence sur la longueur n de la derivation $P \overset{n}{\longleftrightarrow}_{\mathcal{R}\cup\mathcal{E}}^{Th_\asymp,\Gamma} Q$.

- si $n = 0$ alors $P = Q$ et la propriété est trivialement vérifiée,
- sinon $P \longleftrightarrow_{\mathcal{R}\cup\mathcal{E}}^{Th_\asymp,\Gamma} P' \overset{n-1}{\longleftrightarrow}_{\mathcal{R}\cup\mathcal{E}}^{Th_\asymp,\Gamma} Q$.

  Il y a six possibilités, selon que l'on applique un axiome, une règle dans le sens direct ou une règle dans le sens symétrique, et selon que la règle ou l'axiome porte sur des termes ou des propositions.
  - Si la règle ou l'axiome porte sur des termes.
    On a dans $\mathcal{T}_{\mathcal{RE}}$ un axiome $\forall\overline{x}\,(c(\overline{x}) \Rightarrow (l(\overline{x}) \asymp r(\overline{x})))$. On va donc instancier cet axiome suivant σ et l'appliquer en utilisant les axiomes de l'égalité.
    Pour chaque cas, on détaille la construction de la preuve dans le calcul des séquents.

* Si on applique un axiome $f \asymp g$ si c ou $g \asymp f$ si c, on a

$$P = P[\sigma(f)]_\omega \approx_{\mathcal{E}}^{Th_\asymp,\Gamma} P[\sigma(g)]_\omega = P'$$

pour une occurrence ω dans P telle que $Prot(P,\omega,\asymp)$ est faux et une substitution σ telle que $T_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash \sigma(c)$.
Par hypothèse de récurrence on a $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P' \Leftrightarrow Q$.
On construit maintenant une preuve de $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P \Leftrightarrow Q$:

$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par contraction et instantiation]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(c(\overline{x}))\Rightarrow(\sigma(f(\overline{x})) \asymp \sigma(g(\overline{x}))) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par implication à gauche et affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash \sigma(c(\overline{x}))$ et $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(f(\overline{x})) \asymp \sigma(g(\overline{x})) \vdash P \Leftrightarrow Q$

$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash \sigma(c(\overline{x}))$ est vrai par hypothèse.
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(f(\overline{x})) \asymp \sigma(g(\overline{x})) \vdash P \Leftrightarrow Q$ est prouvé ainsi :

$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(f(\overline{x})) \asymp \sigma(g(\overline{x})) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [en utilisant les axiomes de l'égalité]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(f(\overline{x})) \asymp \sigma(g(\overline{x})) \vdash P' \Leftrightarrow Q$
$\Leftarrow$ [par affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P' \Leftrightarrow Q$

ce qui est exactement l'hypothèse de récurrence.

* Si on applique une règle $l \to r$ si c dans le sens direct, on a

$$P = P[\sigma(l)]_\omega \to_{\mathcal{R}}^{Th_\asymp,\Gamma} P[\sigma(r)]_\omega = P'$$

pour une occurrence ω dans P telle que $Prot(P,\omega,\asymp)$ est faux et une substitution σ telle que $T_{\mathcal{RE}}, \Gamma \vdash \sigma(c)$.
Par hypothèse de récurrence on a $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P' \Leftrightarrow Q$.
On construit maintenant une preuve de $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P \Leftrightarrow Q$:

$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par contraction et instantiation]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(c(\overline{x}))\Rightarrow(\sigma(l(\overline{x})) \asymp \sigma(r(\overline{x}))) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par implication à gauche et affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash \sigma(c(\overline{x}))$ et $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$

$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash \sigma(c(\overline{x}))$ est vrai par hypothèse.
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$ est prouvé ainsi :

$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [en utilisant les axiomes de l'égalité]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P' \Leftrightarrow Q$
$\Leftarrow$ [par affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P' \Leftrightarrow Q$

ce qui est exactement l'hypothèse de récurrence.

* Si on applique une règle $l \to r$ si c dans le sens symétrique, on a

$$P = P[\sigma(r)]_\omega \leftarrow_{\mathcal{R}}^{Th_\asymp,\Gamma} P[\sigma(l)]_\omega = P'$$

pour une occurrence ω dans P' telle que $Prot(P,\omega,\asymp)$ est faux et une substitution σ telle que $T_{\mathcal{RE}}, \Gamma \vdash \sigma(c)$.
Par hypothèse de récurrence on a $\mathcal{T}_{\mathcal{RE}}, Th_\asymp, \Gamma \vdash P' \Leftrightarrow Q$.

On construit maintenant une preuve de $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P \Leftrightarrow Q$:

$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par contraction et instantiation]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(c(\overline{x})) \Rightarrow (\sigma(l(\overline{x})) \asymp \sigma(r(\overline{x}))) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par implication à gauche et affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash \sigma(c(\overline{x}))$ et $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$

$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash \sigma(c(\overline{x}))$ est vrai par hypothèse.
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$ est prouvé ainsi :

$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [en utilisant les axiomes de l'égalité]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(l(\overline{x})) \asymp \sigma(r(\overline{x})) \vdash P' \Leftrightarrow Q$
$\Leftarrow$ [par affaiblissment]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P' \Leftrightarrow Q$

ce qui est exactement l'hypothèse de récurrence.

- Si la règle ou l'axiome porte sur des propositions.
  On a dans $\mathcal{T}_{\mathcal{RE}}$ un axiome $\forall \overline{x}\,(c(\overline{x}) \Rightarrow (l(\overline{x}) \Leftrightarrow r(\overline{x})))$. On va donc instancier cet axiome suivant σ et appliquer l'équivalence obtenue.
  Pour chaque cas, on détaille la construction de la preuve dans le calcul des séquents.
  * Si on applique un axiome $f \asymp g$ si c ou $g \asymp f$ si c, on a

$$P = P[\sigma(f)]_\omega \approx_{\mathcal{E}}^{Th_{\asymp},\Gamma} P[\sigma(g)]_\omega = P'$$

pour une occurrence ω dans P telle que $Prot(P, \omega, \asymp)$ est faux et une substitution σ telle que $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash \sigma(c)$.
Par hypothèse de récurrence on a $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P' \Leftrightarrow Q$.
On construit maintenant une preuve de $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P \Leftrightarrow Q$:

$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par contraction et instantiation]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(c(\overline{x})) \Rightarrow (\sigma(f(\overline{x})) \Leftrightarrow \sigma(g(\overline{x}))) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par implication à gauche et affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash \sigma(c(\overline{x}))$ et $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(f(\overline{x})) \Leftrightarrow \sigma(g(\overline{x})) \vdash P \Leftrightarrow Q$

$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash \sigma(c(\overline{x}))$ est vrai par hypothèse.
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(f(\overline{x})) \Leftrightarrow \sigma(g(\overline{x})) \vdash P \Leftrightarrow Q$ est prouvé ainsi :

$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(f(\overline{x})) \Leftrightarrow \sigma(g(\overline{x})) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [grâce à l'équivalence]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma, \sigma(f(\overline{x})) \Leftrightarrow \sigma(g(\overline{x})) \vdash P' \Leftrightarrow Q$
$\Leftarrow$ [par affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P' \Leftrightarrow Q$

ce qui est exactement l'hypothèse de récurrence.

  *
  * Si on applique une règle $l \to r$ si c dans le sens direct, on a

$$P = P[\sigma(l)]_\omega \to_{\mathcal{R}}^{Th_{\asymp},\Gamma} P[\sigma(r)]_\omega = P'$$

pour une occurrence ω dans P telle que $Prot(P, \omega, \asymp)$ est faux et une substitution σ telle que $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash \sigma(c)$.
Par hypothèse de récurrence on a $\mathcal{T}_{\mathcal{RE}}, Th_{\asymp}, \Gamma \vdash P' \Leftrightarrow Q$.

On construit maintenant une preuve de $\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash P \Leftrightarrow Q$:

$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par contraction et instantiation]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(c(\overline{x})) \Rightarrow (\sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x}))) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par implication à gauche et affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash \sigma(c(\overline{x}))$ et $\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$

$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash \sigma(c(\overline{x}))$ est vrai par hypothèse.
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$ est prouvé ainsi :

$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [grâce à l'équivalence]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P' \Leftrightarrow Q$
$\Leftarrow$ [par affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash P' \Leftrightarrow Q$

ce qui est exactement l'hypothèse de récurrence.

* Si on applique une règle $l \to r$ si c dans le sens symétrique, on a

$$P = P[\sigma(r)]_\omega \leftarrow_{\mathcal{R}}^{Th_{\prec}, \Gamma} P[\sigma(l)]_\omega = P'$$

pour une occurrence ω dans P' telle que $Prot(P, \omega, \prec)$ est faux et une substitution σ telle que $\mathcal{T}_{\mathcal{RE}}, \Gamma \vdash \sigma(c)$.
Par hypothèse de récurrence on a $\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash P' \Leftrightarrow Q$.
On construit maintenant une preuve de $\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash P \Leftrightarrow Q$:

$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par contraction et instantiation]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(c(\overline{x})) \Rightarrow (\sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x}))) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [par implication à gauche et affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash \sigma(c(\overline{x}))$ and $\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$

$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash \sigma(c(\overline{x}))$ est vrai par hypothèse.
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$ est prouvé ainsi :

$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P \Leftrightarrow Q$
$\Leftarrow$ [grâce à l'équivalence]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma, \sigma(l(\overline{x})) \Leftrightarrow \sigma(r(\overline{x})) \vdash P' \Leftrightarrow Q$
$\Leftarrow$ [par affaiblissement]
$\mathcal{T}_{\mathcal{RE}}, Th_{\prec}, \Gamma \vdash P' \Leftrightarrow Q$

ce qui est exactement l'hypothèse de récurrence.

B. Pour toute proposition P dans $\mathcal{T}_{\mathcal{RE}}$, on a $Th_{\prec} \vdash_{\mathcal{RE}} P$.
Il y a deux sortes de propositions dans $\mathcal{T}_{\mathcal{RE}}$.
- Si P est de la forme, $\forall \overline{x}(c \Rightarrow (p \Leftrightarrow q))$, on a à prouver qu'on a $Th_{\prec} \vdash_{\mathcal{RE}} \forall \overline{x}(c \Rightarrow (p \Leftrightarrow q))$. Pour cela on va appliquer l'axiome ou la règle correspondant dans $\mathcal{RE}$, ce qui s'écrit ainsi dans le calcul des séquents modulo :

$Th_{\prec} \vdash_{\mathcal{RE}} \forall \overline{x}(c \Rightarrow (p \Leftrightarrow q))$
$\Leftarrow$ [par affaiblissement]
$\vdash_{\mathcal{RE}} \forall \overline{x}(c \Rightarrow (p \Leftrightarrow q))$
$\Leftarrow$ [par libération et implication à droite]
$c(\overline{y}) \vdash_{\mathcal{RE}} p(\overline{y}) \Leftrightarrow q(\overline{y})$
$\Leftarrow$ [par et à droite et deux fois implication à droite]
$c(\overline{y}), p(\overline{y}) \vdash_{\mathcal{RE}} q(\overline{y})$ et $c(\overline{y}), q(\overline{y}) \vdash_{\mathcal{RE}} p(\overline{y})$

Qui sont vrais par application de Axiome modulo.

- Si P est de la forme $\forall \overline{x}(c \Rightarrow (g \asymp d))$, on a à prouver qu'on a $Th_{\asymp} \vdash_{\mathcal{RE}} \forall \overline{x}(c \Rightarrow (g \asymp d))$. Pour cela on va à nouveau appliquer l'axiome ou la règle correspondant dans $\mathcal{RE}$, cette fois-ci à l'aide de $Th_{\asymp}$, ce qui s'écrit ainsi dans le calcul des séquents modulo :

  $Th_{\asymp} \vdash_{\mathcal{RE}} \forall \overline{x}(c \Rightarrow (g \asymp d))$
  $\Leftarrow$ [par libération et implication à droite]
  $Th_{\asymp}, c(\overline{y}) \vdash_{\mathcal{RE}} g(\overline{y}) \asymp d(\overline{y})$
  $\Leftarrow$ [par affaiblissement]
  $\forall x\,(x \asymp x), c(\overline{y}) \vdash_{\mathcal{RE}} g(\overline{y}) \asymp d(\overline{y})$
  $\Leftarrow$ [par instantiation]
  $g(\overline{y}) \asymp g(\overline{y}), c(\overline{y}) \vdash_{\mathcal{RE}} g(\overline{y}) \asymp d(\overline{y})$

  Ce qui est vrai par application de Axiome modulo.

□

Remarque 3.3 Bien sûr il est tentant d'organiser les choses de façon à obtenir des systèmes de réécriture de classes $\mathcal{RE}$ qui soient confluents et noethériens. Cela facilite en particulier la décision de l'égalité modulo $\mathcal{RE}$. Mais on doit remarquer qu'à ce point du formalisme, aucune restriction à propos de la confluence ou de la terminaison de $\mathcal{RE}$ n'est imposée, ce qui sera utile par la suite puisqu'on aura un système faiblement terminant pour la représentation de la logique d'ordre supérieur et de la réécriture ordonnée dans la récurrence par réécriture. De plus la confluence n'est pas non plus exigée pour la récurrence par réécriture (la convergence close de la définition n'est nécessaire que pour obtenir la complétude réfutationnelle).

Remarque 3.4 Le souhait d'avoir des systèmes confluents et noethériens est une limitation à la modularité. En effet, l'union de deux systèmes confluents et noethériens n'est pas nécessairement un système confluent et noethérien pour l'union des congruences. Dans le cas des systèmes de réécriture habituels, la confluence est modulaire si les systèmes sont disjoints [Toy87]. Pour la terminaison, des conditions suffisantes existent également :

- aucun des deux systèmes ne contient de règle effondrante [Rus87],
- aucun des deux systèmes ne contient de règle dupliquante [Rus87],
- l'un des deux systèmes ne contient ni règle effondrante, ni règle dupliquante [Mid89].

Ces trois conditions, comme le montre Ohlebusch dans [Ohl93], sont des corollaires de la condition suivante : si les systèmes sont disjoints et terminants et que leur union est non terminante, alors l'un des systèmes a une règle effondrante et l'autre a une règle dupliquante.

Chapitre 4

La logique du premier ordre comme déduction modulo

Le calcul des séquents du premier ordre est généralement considéré comme ne contenant aucun **calcul** mais seulement de la **déduction** pure. Mais, comme l'a remarqué Patrick Viry [Vir98], cela n'est pas complètement vrai si on y regarde avec attention, en utilisant un environnement de *déduction modulo.*

Les éléments de calcul que l'on va mettre dans la congruence sont d'abord des comportements implicites du calcul, puis des conséquences bien connues qu'on ne souhaite plus prouver. Suivant Patrick Viry, on transforme le calcul des séquents en système de réécriture pour terminer avec un calcul sous forme de *théorie de réécriture orientée* [Vir95] pleinement dans l'esprit de la déduction modulo [DHK98].

Le système auquel aboutit Patrick Viry dans [Vir98] n'a pas les propriétés souhaitées, on le modifie donc afin de les obtenir. Ce travail a été publié à la session étudiante de ESSLLI 2000 [Dep00].

4.1 Rendre le calcul des séquents explicite

On donne ici (figure 4.1) une définition du calcul des séquents classique qui est moins générale que celle de [GLT89] dans le sens qu'elle ne peut pas être étendue facilement, à l'intuitionnisme par exemple. Elle est néanmoins suffisante pour la logique classique et servira mieux nos objectifs pour des raisons qui seront précisées section 4.5.

La congruence sera représentée par un système de réécriture, constituant la partie ER de la théorie de réécriture orientée, confluent et terminant modulo l'associativité-commutativité de certains opérateurs, conservée dans la partie E de la théorie de réécriture orientée. On va modifier le calcul en identifiant autant d'éléments de calcul que possible, afin d'obtenir la congruence la plus large possible tout en préservant la confluence. Il faudra bien sûr être également attentif à préserver la prouvabilité.

Les règles du système de réécriture orientée sont notés avec le symbole $\longrightarrow$ pour la partie R, avec le symbole $\xrightarrow{\approx}$ pour la partie ER et avec le symbole $\approx$ pour la partie E. Le symbole $\xrightarrow{\approx}$ représente le fait que bien qu'elles soient orientées d'un point de vue opérationnel, les règles dans ER conservent leur sémantique de congruence.

4.1.1 Ensembles de formules

Dans la figure 4.1, les $\underline{A}, \underline{B}, \ldots$ sont des ensembles de formules, donc l'operateur ',' :
- est commutatif : ceci est implémenté dans les règles $\mathcal{L}X$ et $\mathcal{R}X$.
- est associatif : ceci est laissé implicite. En effet on n'a même aucune parenthèse dans $\mathcal{L}X$, $\mathcal{R}X$, $\mathcal{L}C$ et $\mathcal{R}C$.
- est idempotent : ceci est implémenté dans les règles $\mathcal{L}C$ et $\mathcal{R}C$.
- admet un élément neutre à droite et à gauche, l'ensemble vide de formules : ceci est implicite dans le membre droit ou gauche vide dans des expressions comme $\vdash \underline{B}$ ou $\underline{A} \vdash$.

$$\frac{}{\underline{A}, C \vdash C, \underline{B}}\ \text{Axiom} \qquad \frac{\underline{A} \vdash C, \underline{B} \quad \underline{A}, C \vdash \underline{B}}{\underline{A} \vdash \underline{B}}\ \text{Cut}$$

$$\frac{\underline{A}, D, C, \underline{A'} \vdash \underline{B}}{\underline{A}, C, D, \underline{A'} \vdash \underline{B}}\ \mathcal{L}X \qquad \frac{\underline{A} \vdash \underline{B}, D, C, \underline{B'}}{\underline{A} \vdash \underline{B}, C, D, \underline{B'}}\ \mathcal{R}X$$

$$\frac{\underline{A}, C, C \vdash \underline{B}}{\underline{A}, C \vdash \underline{B}}\ \mathcal{L}C \qquad \frac{\underline{A} \vdash C, C, \underline{B}}{\underline{A} \vdash C, \underline{B}}\ \mathcal{R}C$$

$$\frac{\underline{A} \vdash C, \underline{B}}{\underline{A}, \neg C \vdash \underline{B}}\ \mathcal{L}\neg \qquad \frac{\underline{A}, C \vdash \underline{B}}{\underline{A} \vdash \neg C, \underline{B}}\ \mathcal{R}\neg$$

$$\frac{\underline{A}, C, D \vdash \underline{B}}{\underline{A}, C \wedge D \vdash \underline{B}}\ \mathcal{L}\wedge \qquad \frac{\underline{A} \vdash C, \underline{B} \quad \underline{A} \vdash D, \underline{B}}{\underline{A} \vdash C \wedge D, \underline{B}}\ \mathcal{R}\wedge$$

$$\frac{\underline{A}, C \vdash \underline{B} \quad \underline{A}, D \vdash \underline{B}}{\underline{A}, C \vee D \vdash \underline{B}}\ \mathcal{L}\vee \qquad \frac{\underline{A} \vdash C, D, \underline{B}}{\underline{A} \vdash C \vee D, \underline{B}}\ \mathcal{R}\vee$$

$$\frac{\underline{A} \vdash C, \underline{B} \quad \underline{A}, D \vdash \underline{B}}{\underline{A}, C \Rightarrow D \vdash \underline{B}}\ \mathcal{L}\Rightarrow \qquad \frac{\underline{A}, C \vdash D, \underline{B}}{\underline{A} \vdash C \Rightarrow D, \underline{B}}\ \mathcal{R}\Rightarrow$$

$$\frac{\underline{A}, C\{a/x\}, \forall x.C \vdash \underline{B}}{\underline{A}, \forall x.C \vdash \underline{B}}\ \mathcal{L}\forall \qquad \frac{\underline{A} \vdash \forall x.C, C\{y/x\}, \underline{B}}{\underline{A} \vdash \forall x.C, \underline{B}}\ \mathcal{R}\forall$$

$$\frac{\underline{A}, C\{y/x\}, \exists x.C \vdash \underline{B}}{\underline{A}, \exists x.C \vdash \underline{B}}\ \mathcal{L}\exists \qquad \frac{\underline{A} \vdash \exists x.C, C\{a/x\}, \underline{B}}{\underline{A} \vdash \exists x.C, \underline{B}}\ \mathcal{R}\exists$$

Dans $\mathcal{R}\forall$ et $\mathcal{L}\exists$, y doit être une nouvelle variable libre.
Dans $\mathcal{L}\forall$ et $\mathcal{R}\exists$, a est un terme quelconque.

FIG. 4.1 – Le calcul des séquents classique

Donc on peut ajouter le symbole '∇' pour l'ensemble de formules vide et les axiomes :

$$\begin{array}{rcl} \underline{A}, (\underline{B}, \underline{C}) & \approx & (\underline{A}, \underline{B}), \underline{C} \\ \underline{A}, \underline{B} & \approx & \underline{B}, \underline{A} \\ \underline{A}, \nabla & \approx & \underline{A} \\ \underline{A}, \underline{A} & \approx & \underline{A} \end{array}$$

A ce stade on peut éliminer $\mathcal{L}C$, $\mathcal{R}C$, $\mathcal{L}X$ et $\mathcal{R}X$ du système de déduction puisqu'ils sont remplacés par la congruence.

4.1.2 Ensembles de séquents

Les règles à deux prémisses ont un opérateur muet qui signifie qu'on manipule en fait des ensembles de séquents. Donc cet opérateur que l'on dénote '$\bullet$' comme dans

$$\frac{\underline{A} \vdash C, \underline{B} \quad \bullet \quad \underline{A}, D \vdash \underline{B}}{\underline{A}, C \Rightarrow D \vdash \underline{B}}\ \mathcal{L}\Rightarrow$$

a les mêmes propriétés que ','. Son élément neutre sera noté '$\diamond$' et n'est rien d'autre que la prémisse vide de la règle Axiom

$$\frac{\diamond}{\underline{A}, C \vdash C, \underline{B}}$$

On ajoute donc à la congruence les axiomes suivants pour '•', où Γ, Δ et Θ sont des ensembles de séquents :

$$\begin{array}{rcl} \underline{\Gamma} \bullet (\underline{\Delta} \bullet \underline{\Theta}) & \approx & (\underline{\Gamma} \bullet \underline{\Delta}) \bullet \underline{\Theta} \\ \underline{\Gamma} \bullet \underline{\Delta} & \approx & \underline{\Delta} \bullet \underline{\Theta} \\ \underline{\Gamma} \bullet \Diamond & \approx & \underline{\Gamma} \\ \underline{\Gamma} \bullet \underline{\Gamma} & \approx & \underline{\Gamma} \end{array}$$

On utilise ici une congruence permettant d'identifier non seulement des formules, mais aussi des ensembles de formules (section précédente) et même ici des ensembles de séquents. La correction de cette congruence dépend grandement des propriétés du calcul des séquents du premier ordre. On doit signaler aussi que cela change l'objet de preuve. L'arbre de preuve habituellement utilisé avec le calcul des séquents devient une suite d'ensembles de séquents. Par exemple :

$$\dfrac{\dfrac{\underline{A} \vdash C, \underline{B} \qquad \dfrac{\underline{A}, D \vdash \underline{B} \quad \underline{A}, E \vdash \underline{B}}{\underline{A}, D \vee E \vdash \underline{B}}\,\mathcal{L}\vee}{\underline{A}, C \Rightarrow (D \vee E) \vdash \underline{B}}\,\mathcal{L}\Rightarrow$$

devient

$$\dfrac{\dfrac{\underline{A} \vdash C, \underline{B} \bullet \underline{A}, D \vdash \underline{B} \bullet \underline{A}, E \vdash \underline{B}}{\underline{A} \vdash C, \underline{B} \bullet \underline{A}, D \vee E \vdash \underline{B}}\,\mathcal{L}\vee}{\underline{A}, C \Rightarrow (D \vee E) \vdash \underline{B}}\,\mathcal{L}\Rightarrow$$

Ce changement est plus important qu'il n'y paraît, parce qu'il permet de rassembler des éléments de la preuve qui seraient très éloignés dans l'arbre de preuve et d'utiliser cela pour simplifier la preuve. Notamment, le fait d'avoir l'axiome $\Gamma \bullet \Gamma \approx \Gamma$ introduit un partage des séquents qui interviennent plusieurs fois dans la preuve, faisant ainsi penser plus à un graphe orienté acyclique qu'à un arbre.

4.1.3 Substitutions explicites

Regardons les règles traitant les quantificateurs. Le $C\{a/x\}$ est un méchanisme de substitution qui est laissé complètement implicite. Le rendre explicite n'est pas évident puisque, à cause des quantificateurs il faut éviter les captures. On va donc avoir besoin d'un calcul de substitutions explicites travaillant avec les quantificateurs au lieu de l'abstraction du λ-calcul.

Ce calcul aura un meilleur comportement que le calcul correspondant pour λ puisque ses lieurs – les quantificateurs – sont des formules et ne peuvent pas être réintroduits par la normalisation d'une substitution. Mais la nécessité de ne pas capturer les variables existe et on a la même solution : les indices de De Bruijn [dB72]. Ce calcul doit gérer la signature de nos termes et formules pour trouver où les substitutions doivent avoir lieu.

On utilise $\lambda\sigma$ [ACCL91] comme base pour notre calcul. Rappelons la signification des notations :

- Les substitutions deviennent des objets syntaxiques de façon a pouvoir être explicitement appliquées à un terme ou à une formule en utilisant l'opérateur '[]',
- Une substitution est essentiellement une liste de termes à substituer aux indices,
- Le constructeur de liste est noté '$\cdot$',
- La substitution identité est notée '*id*' et correspond à la liste infinie $1 \cdot 2 \cdot 3 \ldots$,
- La substitution décalage est notée '$\uparrow$' et correspond à la liste infinie $2 \cdot 3 \cdot 4 \ldots$, elle est utilisée pour gérer les lieurs et de telle façon que 2 soit noté $1[\uparrow]$, 3 soit noté $1[\uparrow][\uparrow]$, ...,
- Le dernier opérateur est '$\circ$' qui dénote la composition de deux substitutions de telle façon que $1[\uparrow][\uparrow]$ soit équivalent à $1[\uparrow \circ \uparrow]$,
- Une substitution $a \cdot b \cdot (\uparrow \circ \uparrow)$ signifie que l'indice 1 doit être remplacé par a et l'indice 2 doit être remplacé par b, les autres indices restant inchangés grâce à la substitution $\uparrow \circ \uparrow$ (qui correspond à $3 \cdot 4 \ldots$).

On commence par les axiomes qui gèrent la signature, pour tout f et P dans la signature :

$$\begin{array}{rcl} f(a_1, \ldots, a_n)[s] & \approx & f(a_1[s], \ldots, a_n[s]) \\ P(a_1, \ldots, a_n)[s] & \approx & P(a_1[s], \ldots, a_n[s]) \end{array}$$

La traversée des quantificateurs, en tenant compte de leur pouvoir liant et en évitant les captures, est gérée par les axiomes :

$$\begin{array}{rcl}(\forall A)[s] & \approx & \forall(A[1\cdot(s\circ\uparrow)])\\ (\exists A)[s] & \approx & \exists(A[1\cdot(s\circ\uparrow)])\end{array}$$

Les axiomes pour les connecteurs sont évidents :

$$\begin{array}{rcl}(\neg A)[s] & \approx & \neg(A[s])\\ (A\wedge B)[s] & \approx & A[s]\wedge B[s]\\ (A\vee B)[s] & \approx & A[s]\vee B[s]\\ (A\Rightarrow B)[s] & \approx & A[s]\Rightarrow B[s]\end{array}$$

On ajoute aussi les axiomes de σ :

$$\begin{array}{rcl}a[id] & \approx & a\\ (a[s])[t] & \approx & a[s\circ t]\\ 1[a\cdot s] & \approx & a\\ id\circ s & \approx & s\\ \uparrow\circ(a\cdot s) & \approx & s\\ (s_1\circ s_2)\circ s_3 & \approx & s_1\circ(s_2\circ s_3)\\ (a\cdot s)\circ t & \approx & a[t]\cdot(s\circ t)\\ s\circ id & \approx & s\\ 1\cdot\uparrow & \approx & id\\ 1[s]\cdot(\uparrow\circ s) & \approx & s\end{array}$$

Et on termine par des axiomes Id et $Clos$ dédiés au cas d'une substitution appliquée à une proposition :

$$\begin{array}{rcl}A[id] & \approx & A\\ (A[s])[t] & \approx & A[s\circ t]\end{array}$$

4.2 Simplification des preuves

Maintenant qu'on a rendu explicites les calculs implicites du calcul des séquents classique, on va s'intéresser à une autre source de calcul. Il y a des propriétés qui sont des conséquences de la logique et qu'on ne veut plus prouver, parce qu'elles sont bien connues. On peut en intégrer certaines dans le calcul, mais on verra qu'on doit être prudent à ce sujet pour préserver les bonnes propriétés du système résultant. On ajoute donc dans la congruence les axiomes :

$$\begin{array}{rcl}A\Rightarrow B & \approx & \neg A\vee B\\ \exists A & \approx & \neg\forall\neg A\end{array}$$

ce qui permet d'abandonner les règles $\mathcal{L}\Rightarrow$, $\mathcal{R}\Rightarrow$, $\mathcal{L}\exists$ et $\mathcal{R}\exists$.

Par ailleurs les équivalences standard :

$$\begin{array}{rcl}\neg\neg A & \approx & A\\ A\wedge A & \approx & A\\ A\vee A & \approx & A\end{array}$$

permettent de simplifier encore les preuves. Cependant on verra section 4.3 qu'il faut ajouter une règle supplémentaire pour préserver la confluence.

On peut aussi, en identifiant des séquents et des ensembles de séquents, déplacer directement certaines règles dans la congruence, y compris la règle Axiom :

$$\begin{array}{rcl}\underline{A},\neg C\vdash\underline{B} & \approx & \underline{A}\vdash C,\underline{B}\\ \underline{A}\vdash\neg C,\underline{B} & \approx & \underline{A},C\vdash\underline{B}\\ \underline{A},C\wedge D\vdash\underline{B} & \approx & \underline{A},C,D\vdash\underline{B}\\ \underline{A}\vdash C\vee D,\underline{B} & \approx & \underline{A}\vdash C,D,\underline{B}\\ \underline{A}\vdash C\wedge D,\underline{B} & \approx & \underline{A}\vdash C,\underline{B}\ \bullet\ \underline{A}\vdash D,\underline{B}\\ \underline{A},C\vee D\vdash\underline{B} & \approx & \underline{A},C\vdash\underline{B}\ \bullet\ \underline{A},D\vdash\underline{B}\\ \underline{A},C\vdash C,\underline{B} & \approx & \Diamond\end{array}$$

4.3 Orientation des équations

On veut maintenant orienter les axiomes pour être opérationnels et pour prouver que la congruence est en fait décidable. La congruence est décidable si on peut orienter les axiomes en un système de réécriture de classes convergent modulo un ensemble d'axiomes pour lequel on sait déterminer une telle propriété. En pratique cela signifie que seuls des axiomes d'associativité-commutativité peuvent rester (figure 4.2). Ceci répond aussi aux critères d'opérationnalité et assure de pouvoir implémenter avec un système utilisant la réécriture comme par exemple ELAN[1] [BKK+98]. Les règles dans ER peuvent correspondre à des règles non nommées en ELAN, les règles dans R étant des règles nommées.

Faire ceci montre qu'on doit être prudent quant aux propriétés qu'on internalise, parce qu'il peut y avoir des problèmes de confluence. Avec les axiomes des sections 4.1 et 4.2, des expressions comme

$$\underline{A} \vdash C \wedge D, C \wedge D, \underline{B}$$

permettent de choisir de décomposer le '$\wedge$' ou d'utiliser l'idempotence de ','. Ce choix ne converge pas :

$$\begin{array}{l}
\underline{A} \vdash C \wedge D, C \wedge D, \underline{B} \longrightarrow \underline{A} \vdash C \wedge D, \underline{B} \longrightarrow \underline{A} \vdash C, \underline{B} \bullet \underline{A} \vdash D, \underline{B} \\
\downarrow \\
\underline{A} \vdash C \wedge D, C, \underline{B} \bullet \underline{A} \vdash C \wedge D, D, \underline{B} \\
\downarrow \\
\underline{A} \vdash C, C, \underline{B} \bullet \underline{A} \vdash D, C, \underline{B} \bullet \underline{A} \vdash C \wedge D, D, \underline{B} \\
\downarrow \\
\underline{A} \vdash C, C, \underline{B} \bullet \underline{A} \vdash D, C, \underline{B} \bullet \underline{A} \vdash C, D, \underline{B} \bullet \underline{A} \vdash D, D, \underline{B} \\
\downarrow \\
\underline{A} \vdash C, \underline{B} \bullet \underline{A} \vdash D, C, \underline{B} \bullet \underline{A} \vdash C, D, \underline{B} \bullet \underline{A} \vdash D, D, \underline{B} \\
\downarrow \\
\underline{A} \vdash C, \underline{B} \bullet \underline{A} \vdash D, C, \underline{B} \bullet \underline{A} \vdash C, D, \underline{B} \bullet \underline{A} \vdash D, \underline{B}
\end{array}$$

ce qui est résolu en ajoutant un axiome de subsomption qui manquait dans [Vir98] :

$$\underline{A} \vdash \underline{B} \bullet \underline{A}, \underline{A'} \vdash \underline{B'}, \underline{B} \approx \underline{A} \vdash \underline{B}$$

Il faut noter que cet axiome est facile à exprimer dans notre formalisme mais ne peut pas être appliqué dans son orientation utile si on garde la représentation habituelle des preuves sous forme d'arbres, en effet cela revient à réunir deux branches de l'arbre, créant ainsi un graphe orienté acyclique.

L'axiome de subsomption ajouté, l'orientation peut être effectuée et la complétion en utilisant CiME [CM96] ajoute des règles de réécriture (voir figure 4.4) pour gérer le cas où un ensemble de formules est vide comme dans

$$\neg C \vdash \underline{B} \to \bigtriangledown \vdash C, \underline{B} \text{ et } C \vdash C, \underline{B} \to \Diamond$$

Ceci assure que ER est localement confluent modulo E. La terminaison du système, conjecturée dans [Dep00] a été prouvée par Xavier Urbain à l'aide de CiME2, qui exploite les techniques de preuve incrémentale de terminaison développées dans sa thèse [Urb01].

$$\mathsf{E} = \left\{ \begin{array}{lrcl}
Ac & \underline{A}, (\underline{B}, \underline{C}) & \approx & (\underline{A}, \underline{B}), \underline{C} \\
Cc & \underline{A}, \underline{B} & \approx & \underline{B}, \underline{A} \\
 & & & \\
A\bullet & \underline{\Gamma} \bullet (\underline{\Delta} \bullet \underline{\Theta}) & \approx & (\underline{\Gamma} \bullet \underline{\Delta}) \bullet \underline{\Theta} \\
C\bullet & \underline{\Gamma} \bullet \underline{\Delta} & \approx & \underline{\Delta} \bullet \underline{\Gamma}
\end{array} \right.$$

FIG. 4.2 – Les axiomes restants (associativité-commutativité)

[1] http ://elan.loria.fr/

4.4 Déconnecter déduction et congruence

Maintenant on veut être en mesure de se concentrer sur l'application des règles de déduction, qui doit être contrôlée de façon non déterministe, et faire uniquement de la normalisation avec la congruence. Pour faire cela, il faut éviter que la congruence puisse interférer avec la déduction. Pour assure cela, on utilise les techniques de [Vir95], donc on transforme les règles de déduction restantes pour former une théorie de réécriture orientée. Ces règles doivent être *cohérentes* avec ER modulo E, ce qui est obtenu par *complétion de cohérence* en ajoutant des règles (voir figure 4.3) pour gérer le cas où un ensemble de formules est vide comme :

$$\forall C \vdash \underline{B} \quad \to \quad C[a \cdot id], \forall C \vdash \underline{B}$$

issu de :

$$\underline{A}, \forall C \vdash \underline{B} \quad \to \quad \underline{A}, C[a \cdot id], \forall C \vdash \underline{B}$$

dans le cas ou $\underline{A}$ est ∇.

On ajoute aussi la règle $\Diamond \to \Box$. Cette règle permet de vérifier que la preuve a été finie par la congruence, par exemple dans le cas propositionnel où cette règle est la seule à rester, traduisant la décidabilité du calcul des propositions. La règle $\underline{\Gamma} \bullet \Box \longrightarrow \underline{\Gamma}$ dans R est seulement nécessaire pour préserver la cohérence , mais ne sera jamais utilisée avec une stratégie utilisant la congruence en tant que normalisation. Contrairement à ce qui est dit dans [Dep00], elle doit rester dans R car elle génère la règle $\Diamond \to \Box$ comme paire critique avec la règle $\underline{\Gamma} \bullet \Diamond \xrightarrow{\approx} \underline{\Gamma}$ dans ER.

$$\mathsf{R} = \begin{cases} \mathcal{L}\forall\text{-1} & \underline{A}, \forall C \vdash \underline{B} & \longrightarrow & \underline{A}, C[a \cdot id], \forall C \vdash \underline{B} \\ \mathcal{L}\forall\text{-2} & \forall C \vdash \underline{B} & \longrightarrow & C[a \cdot id], \forall C \vdash \underline{B} \\ \mathcal{R}\forall\text{-1} & \underline{A} \vdash \forall C, \underline{B} & \longrightarrow & \underline{A} \vdash \forall C, C[n \cdot id], \underline{B} \\ \mathcal{R}\forall\text{-2} & \underline{A} \vdash \forall C & \longrightarrow & \underline{A} \vdash \forall C, C[n \cdot id] \\ \text{Cut} & \underline{A} \vdash \underline{B} & \longrightarrow \underline{A} \vdash C, \underline{B} \bullet \underline{A}, C \vdash \underline{B} & \\ U\bullet\text{-2} & \underline{\Gamma} \bullet \Box & \longrightarrow & \underline{\Gamma} \\ \text{congruent} & \Diamond & \longrightarrow & \Box \end{cases}$$

In $\mathcal{R}\forall$-1 and $\mathcal{R}\forall$-2, n must be a fresh free index.
In $\mathcal{L}\forall$-1 and $\mathcal{L}\forall$-2, a is any term.

FIG. 4.3 – Les règles de déduction

On obtient que R est *fortement cohérent* avec ER modulo E, ce qui assure que n'importe quelle stratégie peut être appliquée pour l'utilisation de la congruence par rapport aux règles de R. On peut remarquer que la congruence peut augmenter beaucoup la taille de l'objet.

Théorème 4.1 *Un séquent $\underline{A} \vdash \underline{B}$ a une preuve dans* R *modulo* ER ∪ E *si et seulement si il en a une dans le calcul des séquents classique.*

Une preuve de $\underline{A} \vdash \underline{B}$ dans R modulo ER ∪ E est une dérivation de $\underline{A} \vdash \underline{B}$ jusqu'à $\Box$. Les résultats en sections 4.3 et 4.4 prouvent qu'utiliser la stratégie que nous avons décrite est équivalent à utiliser la relation R/ER ∪ E donc il rester à prouver que $\underline{A} \vdash \underline{B} \xrightarrow{\mathsf{R/ERUE}} \Box$ si et seulement si $\underline{A} \vdash \underline{B}$ a une preuve dans le calcul des séquents classique. Notons que la relation R/ER ∪ E considère ER comme un ensemble d'équations plutôt que comme un ensemble de règles.

La partie "si" est facile, en simulant chaque règle du calcul des séquents par des règles de R ou des axiomes de ER ∪ E et en remarquant qu'une preuve se termine toujours par une utilisation de la règle *congruent*.

La partie "seulement si" doit assurer que toutes les équations introduites dans les sections 4.1, 4.2 et 4.3 identifient des objets dont la prouvabilité est la même dans le calcul des séquents classique. Ceci est évident pour les équations introduites en section 4.1. C'est encore facile pour les équations introduites en section 4.2 et 4.3 en se rappelant qu'on veut préserver les preuves et non leur structure.

$\mathsf{ER}_{\mathsf{Seq}} \approx$

Uc	$\underline{A}, \triangledown$	$\xrightarrow{\approx}$	$\underline{A}$
Ic	$\underline{A}, \underline{A}$	$\xrightarrow{\approx}$	$\underline{A}$
$U\bullet$-1	$\underline{\Gamma} \bullet \diamond$	$\xrightarrow{\approx}$	$\underline{\Gamma}$
$I\bullet$	$\underline{\Gamma} \bullet \underline{\Gamma}$	$\xrightarrow{\approx}$	$\underline{\Gamma}$
$f\neg$	$\neg\neg A$	$\xrightarrow{\approx}$	A
$fI\wedge$	$A \wedge A$	$\xrightarrow{\approx}$	A
$fI\vee$	$A \vee A$	$\xrightarrow{\approx}$	A
$f\Rightarrow$	$A \Rightarrow B$	$\xrightarrow{\approx}$	$\neg A \vee B$
$f\exists$	$\exists A$	$\xrightarrow{\approx}$	$\neg\forall\neg A$
$s\neg l$-1	$\underline{A}, \neg C \vdash \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash C, \underline{B}$
$s\neg l$-2	$\neg C \vdash \underline{B}$	$\xrightarrow{\approx}$	$\triangledown \vdash C, \underline{B}$
$s\neg r$-1	$\underline{A} \vdash \neg C, \underline{B}$	$\xrightarrow{\approx}$	$\underline{A}, C \vdash \underline{B}$
$s\neg r$-2	$\underline{A} \vdash \neg C$	$\xrightarrow{\approx}$	$\underline{A}, C \vdash \triangledown$
$\mathcal{L}\wedge$-1	$\underline{A}, C \wedge D \vdash \underline{B}$	$\xrightarrow{\approx}$	$\underline{A}, C, D \vdash \underline{B}$
$\mathcal{L}\wedge$-2	$C \wedge D \vdash \underline{B}$	$\xrightarrow{\approx}$	$C, D \vdash \underline{B}$
$\mathcal{R}\vee$-1	$\underline{A} \vdash C \vee D, \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash C, D, \underline{B}$
$\mathcal{R}\vee$-2	$\underline{A} \vdash C \vee D$	$\xrightarrow{\approx}$	$\underline{A} \vdash C, D$
$\mathcal{R}\wedge$-1	$\underline{A} \vdash C \wedge D, \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash C, \underline{B} \bullet \underline{A} \vdash D, \underline{B}$
$\mathcal{R}\wedge$-2	$\underline{A} \vdash C \wedge D$	$\xrightarrow{\approx}$	$\underline{A} \vdash C \bullet \underline{A} \vdash D$
$\mathcal{L}\vee$-1	$\underline{A}, C \vee D \vdash \underline{B}$	$\xrightarrow{\approx}$	$\underline{A}, C \vdash \underline{B} \bullet \underline{A}, D \vdash \underline{B}$
$\mathcal{L}\vee$-2	$C \vee D \vdash \underline{B}$	$\xrightarrow{\approx}$	$C \vdash \underline{B} \bullet D \vdash \underline{B}$
Ax-1	$\underline{A}, C \vdash C, \underline{B}$	$\xrightarrow{\approx}$	$\diamond$
Ax-2	$\underline{A}, C \vdash C$	$\xrightarrow{\approx}$	$\diamond$
Ax-3	$C \vdash C, \underline{B}$	$\xrightarrow{\approx}$	$\diamond$
Ax-4	$C \vdash C$	$\xrightarrow{\approx}$	$\diamond$
Sub-1	$\underline{A} \vdash \underline{B} \bullet \underline{A}, \underline{A'} \vdash \underline{B'}, \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash \underline{B}$
Sub-2	$\underline{A} \vdash \underline{B} \bullet \underline{A}, \underline{A'} \vdash \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash \underline{B}$
Sub-3	$\underline{A} \vdash \underline{B} \bullet \underline{A} \vdash \underline{B'}, \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash \underline{B}$
Sub-4	$\underline{A} \vdash \triangledown \bullet \underline{A}, \underline{A'} \vdash \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash \triangledown$
Sub-5	$\triangledown \vdash \underline{B} \bullet \underline{A} \vdash \underline{B'}, \underline{B}$	$\xrightarrow{\approx}$	$\triangledown \vdash \underline{B}$
Sub-6	$\underline{A} \vdash \triangledown \bullet \underline{A} \vdash \underline{B}$	$\xrightarrow{\approx}$	$\underline{A} \vdash \triangledown$
Sub-7	$\triangledown \vdash \underline{B} \bullet \underline{A} \vdash \underline{B}$	$\xrightarrow{\approx}$	$\triangledown \vdash \underline{B}$
Sub-8	$\triangledown \vdash \triangledown \bullet \underline{A} \vdash \underline{B}$	$\xrightarrow{\approx}$	$\triangledown \vdash \triangledown$

$\mathsf{ER} = \mathsf{ER}_{\forall\exists\sigma} \cup \mathsf{ER}_{\mathsf{Seq}}$

FIG. 4.4 – La congruence (orientée)

$$
\mathsf{ER}_{\forall\exists\sigma} = \left\{
\begin{array}{lrcl}
App_f & f(a_1,\ldots,a_n)[s] & \stackrel{\approx}{\longrightarrow} & f(a_1[s],\ldots,a_n[s]) \\
App_P & P(a_1,\ldots,a_n)[s] & \stackrel{\approx}{\longrightarrow} & P(a_1[s],\ldots,a_n[s]) \\
App_\neg & (\neg A)[s] & \stackrel{\approx}{\longrightarrow} & \neg(A[s]) \\
App_\wedge & (A \wedge B)[s] & \stackrel{\approx}{\longrightarrow} & A[s] \wedge B[s] \\
App_\vee & (A \vee B)[s] & \stackrel{\approx}{\longrightarrow} & A[s] \vee B[s] \\
App_\Rightarrow & (A \Rightarrow B)[s] & \stackrel{\approx}{\longrightarrow} & A[s] \Rightarrow B[s] \\
Abs_\forall & (\forall A)[s] & \stackrel{\approx}{\longrightarrow} & \forall(A[1 \cdot (s \circ \uparrow)]) \\
Abs_\exists & (\exists A)[s] & \stackrel{\approx}{\longrightarrow} & \exists(A[1 \cdot (s \circ \uparrow)]) \\
Id_t & a[id] & \stackrel{\approx}{\longrightarrow} & a \\
Id_f & A[id] & \stackrel{\approx}{\longrightarrow} & A \\
Clos_t & (a[s])[t] & \stackrel{\approx}{\longrightarrow} & a[s \circ t] \\
Clos_f & (A[s])[t] & \stackrel{\approx}{\longrightarrow} & A[s \circ t] \\
VarCons & 1[a \cdot s] & \stackrel{\approx}{\longrightarrow} & a \\
IdL & id \circ s & \stackrel{\approx}{\longrightarrow} & s \\
ShiftCons & \uparrow \circ (a \cdot s) & \stackrel{\approx}{\longrightarrow} & s \\
AssEnv & (s_1 \circ s_2) \circ s_3 & \stackrel{\approx}{\longrightarrow} & s_1 \circ (s_2 \circ s_3) \\
MapEnv & (a \cdot s) \circ t & \stackrel{\approx}{\longrightarrow} & a[t] \cdot (s \circ t) \\
IdR & s \circ id & \stackrel{\approx}{\longrightarrow} & s \\
VarShift & 1 \cdot \uparrow & \stackrel{\approx}{\longrightarrow} & id \\
SCons & 1[s] \cdot (\uparrow \circ s) & \stackrel{\approx}{\longrightarrow} & s
\end{array}
\right.
$$

FIG. 4.5 – Le calcul de substitutions pour les quantificateurs

4.5 Choix du calcul

Dans cette section, on va expliquer les choix faits pour le calcul de la figure 4.1. On va aussi voir comment il est possible d'aller plus loin.

- Pour préserver la confluence, on n'a pas de règle d'affaiblissement. En effet ces règles génèrent des paires critiques qu'il est impossible de refermer. L'absence de règle d'affaiblissement est compensée par la forme de la règle Axiom de la figure 4.1, qui effectue en une seule fois tous les affaiblissements.
- La présence de contractions implicites dans les règles concernant les quantificateurs est nécessaire pour obtenir la cohérence avec la règle issue de la contraction. Je remercie ici Jürgen Stuber de m'avoir fait remarquer ce problème.
  Intuitivement, le problème est qu'il peut être nécessaire d'utiliser plusieurs instances d'une quantification, ce qui est rendu impossible par la contraction orientée dans le sens terminant d'une réduction. Formellement :

$$
\begin{array}{l}
\underline{A}, \forall C, \forall C \vdash \underline{B} \longrightarrow \underline{A}, C[a.id], \forall C \vdash \underline{B} \longrightarrow \underline{A}, C[a.id], C[b.id] \vdash \underline{B} \\
\downarrow \\
\underline{A}, \forall C \vdash \underline{B} \\
\downarrow \\
\underline{A}, C[a.id] \vdash \underline{B}
\end{array}
$$

- La forme choisie pour les règles $\mathcal{R}\vee$ et $\mathcal{L}\wedge$ permet d'éviter d'introduire des affaiblissements cachés. Elle n'est cependant pas compatible avec la logique intuitionniste. Pour gérer la logique in-

tuitionniste, il faudrait donc introduire un opérateur supplémentaire représentant la disjonction d'ensembles de formules.

- On n'a pas non plus la formule toujours fausse $\bot$. On pourrait l'introduire, ainsi que la règle correspondante et la définition $\neg A \approx A \Rightarrow \bot$, ce qui est nécessaire pour la logique intuitionniste. En revanche, poser $A \wedge \neg A \approx \bot$ comme dans [Vir98] provoque l'apparition de la règle de coupure en tant que paire critique, et on ne parvient pas ensuite à trouver un système confluent.

4.6 Conclusion

On a montré que le calcul des séquents du premier ordre peut être vu comme un calcul modulo. Premièrement en rendant explicites ses éléments calculatoires implicites, ensuite en ajoutant à la congruence plusieurs conséquences du calcul.

Les seules règles de déduction qui restent ensuite sont différentes versions de $\mathcal{R}\forall$ et $\mathcal{L}\forall$. On peut voir que ce sont les règles qui gèrent les quantificateurs et en effet il est bien connu que l'indécidabilité de la logique du premier ordre réside là. On a donc obtenu une distinction claire entre la partie déductive indécidable et la partie calcul décidable placée dans la congruence.

Ceci rend la recherche de preuves plus simple puisque la partie calcul peut être gérée par simple normalisation de l'ensemble de séquents que l'on veut prouver et donc on n'a qu'à chercher l'option intelligente lorsqu'on utilise une règle de R. Cette idée pourrait être implémentée en ELAN [BKK+98] en utilisant des règles non nommées pour ER et des règles nommées et une stratégie pour R. Il resterait cependant à gérer les témoins a dans les règles $\mathcal{L}\forall$ et l'introduction de lemmes par la règle Cut. Cette gestion peut se faire soit par des interactions avec l'utisateur soit avec des méthodes d'unification ou d'autres méthodes de recherche automatique de preuves.

Deuxième partie

Preuves par récurrence

Chapitre 5

Logique d'ordre supérieur

La logique d'ordre supérieur permet de quantifier non seulement sur les termes, mais aussi sur les fonctions et même les prédicats et donc les formules. Cette possibilité met au même plan termes, fonctions et prédicats, le formalisme le plus évident pour exprimer la logique d'ordre supérieur est donc le λ-calcul, ce qui donne le système HOL_λ.

Cependant, puisque notre premier but est de formaliser les preuves par récurrence dans le contexte de la déduction modulo, nous aurons à exprimer l'axiome de récurrence, qui est une formule d'ordre deux. Pour le faire dans une présentation de premier ordre de la logique d'ordre supérieur, on utilise le formalisme introduit dans [DHK01].

5.1 lambda-calcul

On rappelle ici les définitions classiques du λ-calcul que l'on peut retrouver dans [Bar84] ou [Kri93] par exemple.

5.1.1 Syntaxe du λ-calcul

Définition 5.1 Soit $\mathcal{X}$ un ensemble infini dénombrable de symboles de variables, et $\mathcal{F}$ un ensemble fini de symboles de constantes.

Les termes du λ-calcul sont construits comme suit :
- si x est dans $\mathcal{X}$ alors x est un λ-terme,
- si f est dans $\mathcal{F}$ alors f est un λ-terme,
- si t est un λ-terme et x est dans $\mathcal{X}$ alors $\lambda x.t$ est un λ-terme,
- si t et t' sont des λ-termes alors $(t\ t')$ est un λ-terme.

Exemple 5.1 Soient $\mathcal{X} = \{x, y, z, \ldots\}$ et $\mathcal{F} = \{a, f\}$. Des exemples de λ-termes sont :
- x,
- a,
- $\lambda x.a$,
- $\lambda x.(f\ x)$,
- $(x\ a)$,
- $(f\ y)$,
- $(x\ y)$,
- $(f\ f)$,
- $(f\ a)$,
- $(\lambda x.(f\ x)\ a)$,
- $(\lambda x.(f\ x)\ y)$,
- $(\lambda x.(x\ x)\ \lambda x.(x\ x))$,
- $((\lambda x.(f\ x)\ y)\ (\lambda z.(f\ z)\ x))$.

5.1.2 Occurrences libres et liées

Définition 5.2 Une occurrence de variable est dite *liée* si elle est dans la portée d'un λ, *libre* sinon. Formellement, soit t un λ-terme et x une variable :
- si t est la variable x, alors
  - l'occurrence de x dans t est libre,
  - x n'a pas d'occurrence liée dans t.
- si t est une variable y différente de x, alors
  - x n'a pas d'occurrence libre dans t,
  - x n'a pas d'occurrence liée dans t.
- si t est de la forme $(t_1\ t_2)$ alors
  - les occurrences libres de x dans t sont ses occurrences libres dans t_1 et t_2,
  - les occurrences liées de x dans t sont ses occurrences liées dans t_1 et t_2.
- si t est de la forme $\lambda y.t'$ avec y différent de x, alors
  - les occurrences libres de x dans t sont ses occurrences libres dans t',
  - les occurrences liées de x dans t sont ses occurrences liées dans t'.
- si t est de la forme $\lambda x.t'$, alors
  - x n'a pas d'occurrence libre dans t,
  - les occurences liées de x dans t sont ses occurences libres et liées dans t'.

Exemple 5.2 Soit t le λ-terme $((\lambda x.(f\ x)\ y)\ (\lambda z.(f\ z)\ x))$.
- x a une occurrence libre et une occurrence liée dans t,
- y a une occurrence libre et pas d'occurrence liée dans t,
- z a une occurrence liée et pas d'occurrence libre dans t.

Remarque 5.1 Une variable peut avoir des occurrences libres et des occurrences liées dans un même terme. De même elle peut avoir des occurrences liées par des λ différents. Ces deux situations sont source de confusion, puisque les différentes occurrences de la même variable syntaxique représentent alors des objets différents. C'est pourquoi Barendregt a posé comme condition d'hygiène d'éviter cela ([Bar84]).

5.1.3 Substitutions

Définition 5.3 Une *substitution* est une application de $\mathcal{X}$ vers les λ-termes.

Notation 5.1 On note $\{x_1/t_1, \ldots, x_n/t_n, \ldots\}$ la substitution donnant à x_1 la valeur t_1, ..., à x_n la valeur t_n, ... On simplifie la notation en ne mentionnant que les variables effectivement modifiées.

L'extension d'une substitution aux λ-termes n'est pas triviale. En effet il faut gérer le risque qu'un λ capture une variable libre.

Notation 5.2 On note $t\{x_1/t_1, \ldots, x_n/t_n, \ldots\}$ l'application de la substitution $\{x_1/t_1, \ldots, x_n/t_n, \ldots\}$ au terme t.

Définition 5.4 On définit la *substitution d'ordre supérieur* par :
- $x\{x_1/t_1, \ldots, x_n/t_n, \ldots\} = t_i$, si $x = x_i$,
- $y\{x_1/t_1, \ldots, x_n/t_n, \ldots\} = y$ si pour tout i, $y \neq x_i$,
- $c\{x_1/t_1, \ldots, x_n/t_n, \ldots\} = c$, si c est une constante,
- $(c\ d)\{x_1/t_1, \ldots, x_n/t_n, \ldots\} = (c\{x_1/t_1, \ldots, x_n/t_n, \ldots\}\ d\{x_1/t_1, \ldots, x_n/t_n, \ldots\})$,
- $\lambda y.c\{x_1/t_1, \ldots, x_n/t_n, \ldots\} = \lambda z.(c\{y/z\}\{x_1/t_1, \ldots, x_n/t_n, \ldots\})$, où z n'est aucun des x_i et n'a d'occurrence libre ni dans c, ni dans aucun des t_i.

Exemple 5.3
- $\lambda x.(xy)\{x/y\} = \lambda x.(x\ y)$
- $\lambda x.(xy)\{y/x\} = \lambda z.(z\ x)$

Remarque 5.2 La substitution d'ordre supérieur n'est en fait pas plus complexe que la substitution sur les formules du premier ordre avec quantificateurs. Les quantificateurs sont en effet également des lieurs. La différence vient du fait que le λ est un symbole de fonction et donc la difficulté apparaît dès les λ-termes, alors qu'au premier ordre seules les formules sont concernées.

5.1.4 Axiomes de conversion

Deux λ-termes syntaxiquement différents peuvent avoir le même "sens", il sera donc nécessaire de pouvoir déterminer si deux termes ont ou non le même sens. Pour cela on associe un *axiome de conversion* à chaque façon d'obtenir deux termes équivalents.

5.1.4.1 Axiome α

La première est la possibilité de renommer une variable liée par un λ, ce qui est notamment nécessaire lorsqu'une variable existe libre et liée ou liée par deux λ différents. Il faut remarquer que cette possibilité existe aussi en logique du premier ordre, au niveau des formules, puisque les quantificateurs sont des lieurs. L'axiome associé est l'axiome α. Cet axiome n'est pas orientable, mais ce problème est résolu dans les implémentations grâce aux indices de De Bruijn ([dB72]).

Définition 5.5 On définit l'*axiome* α par :

$$\lambda x.t =_\alpha \lambda y.t\{x/y\} \text{ si } y \text{ n'a pas d'occurrence libre dans } t$$

Exemple 5.4 Soient $\mathcal{X}$ et $\mathcal{F}$ comme précédemment.
$\lambda x.(f\ x) =_\alpha \lambda z.(f\ z)$.

Un terme avec indices de De Bruijn représente la classe des termes équivalents par l'axiome α. Lorsqu'on fait un raisonnement à la main et qu'on veut éviter de devoir utiliser l'axiome alpha, mais qu'on ne veut pas utiliser les indices de De Bruijn, notamment parce qu'ils ne sont pas faciles à lire, on utilise les conventions de Barendregt [Bar84], qui reviennent simplement à ne pas utiliser la même variable pour deux objets différents.

5.1.4.2 Axiome β

La deuxième vient de la forme $(t\ t')$ représentant l'application d'une fonction à un argument, il faut pouvoir évaluer le résultat de cette application. L'axiome permettant cela est l'axiome β, qui est l'axiome essentiel du λ-calcul et est toujours donné sous forme de règle :

Définition 5.6 On définit l'*axiome* β par

$$(\lambda x.t\ t') \rightarrow_\beta t\{x/t'\}$$

Exemple 5.5 Soient $\mathcal{X}$ et $\mathcal{F}$ comme précédemment.
$(\lambda x.(f\ x)\ a) \rightarrow_\beta \lambda x.(f\ x)\{x/a\}$.

5.1.4.3 Axiome η

La troisième et dernière est que l'on peut avoir une fonction qui ne se sert pas de son argument, l'axiome correspondant est l'axiome η, qu'on trouve utilisée en tant que réduction, mais aussi en tant qu'expansion dans le λ-calcul typé, où le typage assure la terminaison d'un tel usage.

Définition 5.7 On définit l'*axiome* η par

$$\lambda x.t =_\eta t \text{ si } x \text{ n'a pas d'occurrence libre dans } t.$$

Exemple 5.6 Soient $\mathcal{X}$ et $\mathcal{F}$ comme précédemment.
$\lambda x.a =_\eta a$.

5.1.5 λ-calcul typé

Définition 5.8 Soit $\mathcal{T}$ un ensemble de types de base.
Les *types simples* sont définis par
- si $T \in \mathcal{T}$ alors T est un type simple.
- si T et U sont des types simples alors $T \to U$ est un type simple.

Exemple 5.7 Soit $\mathcal{T} = \{\iota, o\}$.
Des exemples de types simples sont :
- $\iota \to o$,
- $o \to o$,
- $(i \to o) \to o$.

Notation 5.3 Soient T, U et V des types simples.
On note $T \to U \to V$ pour $T \to (U \to V)$.

Définition 5.9 Seuls sont considérés comme valides dans le λ-calcul typé les *λ-termes typés* auxquels on peut associer un type à partir de ceux attribués aux symboles de variables et de constantes de la façon suivante :
- si t est un terme de type U et x est une variable de type T alors $\lambda x.t$ est un terme de type $T \to U$.
- si t est un terme de type $T \to U$ et t' est un terme de type T alors $(t\ t')$ est un terme de type U.

Exemple 5.8 Soient $\mathcal{T} = \{T_a, T_f\}$, $\mathcal{X} = \{x, y, z, \ldots\}$ et $\mathcal{F} = \{a, f\}$, avec x de type T_a, y de type T_a, z de type T_a, a de type T_a et f de type $T_a \to T_f$.
- $\lambda x.a$ est de type $T_a \to T_a$,
- $\lambda x.(f\ x)$ est de type $T_a \to T_f$,
- $(x\ a)$ n'est pas un λ-terme typé,
- $(f\ y)$ est de type T_f,
- $(x\ y)$ n'est pas un λ-terme typé,
- $(f\ f)$ n'est pas un λ-terme typé,
- $(f\ a)$ est de type T_f,
- $(\lambda x.(f\ x)\ a)$ est de type T_f,
- $(\lambda x.(f\ x)\ y)$ est de type T_f,
- $(\lambda x.(x\ x)\ \lambda x.(x\ x))$ n'est pas un λ-terme typé,
- $((\lambda x.(f\ x)\ y)\ (\lambda z.(f\ z)\ x))$ n'est pas un λ-terme typé.

Proposition 5.1 La réduction $\beta\eta$ (avec η orienté dans le sens d'une réduction) est confluente et terminante dans le λ-calcul typé. La forme normale (unique) d'un terme a est notée $a\downarrow$.

5.2 HOL_λ

On considère un λ-calcul typé ayant comme types de base ι et o, et parmi ses constantes $\dot{\Rightarrow}$, $\dot{\wedge}$ et $\dot{\vee}$, de type $o \to o \to o$, $\dot{\neg}$ de type $o \to o$, $\dot{\bot}$ de type o, $\dot{\forall}_T$ et $\dot{\exists}_T$ de type $(T \to o) \to o$ (on utilise une notation avec un point pour les constantes pour les différentier des connecteurs et quantificateurs de la logique du premier ordre).

Les propositions sont par définition les termes of type o.

Les règles de déduction ([DHK01]) sont données en figure 5.1 où toutes les propositions sont supposées en forme normale. Une présentation alternative ne normalise pas les propositions après les règles sur les quantificateurs mais prend β et η comme axiomes.

Ce système est connu pour être consistant et vérifier l'élimination des coupures [Gir70, Gir72].

5.3 Substitutions explicites

La formulation de la logique d'ordre supérieur que l'on utilise est basée sur les indices de De Bruijn et les substitutions explicites. Les substitutions explicites ont été développées à partir de [ACCL91] pour

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

$$\frac{\Gamma, P, P \vdash \Delta}{\Gamma, P \vdash \Delta}\text{contr-l} \qquad \frac{\Gamma \vdash P, P, \Delta}{\Gamma \vdash P, \Delta}\text{contr-r}$$

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

$$\frac{\Gamma \vdash P, \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, (\Rightarrow P\ Q) \vdash \Delta}{\Rightarrow}\text{-l} \qquad \frac{P, \Gamma \vdash Q, \Delta}{\Gamma \vdash (\Rightarrow P\ Q), \Delta}{\Rightarrow}\text{-r}$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, (\dot{\wedge}\ P\ Q) \vdash \Delta}\dot{\wedge}\text{-l} \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash (\dot{\wedge}\ P\ Q), \Delta}\dot{\wedge}\text{-r}$$

$$\frac{\Gamma, P \vdash \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, (\dot{\vee}\ P\ Q) \vdash \Delta}\dot{\vee}\text{-l} \qquad \frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash (\dot{\vee}\ P\ Q), \Delta}\dot{\vee}\text{-r}$$

$$\frac{\Gamma \vdash P, \Delta}{\Gamma, (\dot{\neg}\ P) \vdash \Delta}\dot{\neg}\text{-l} \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash (\dot{\neg}\ P), \Delta}\dot{\neg}\text{-r}$$

$$\frac{}{\Gamma, \dot{\perp} \vdash \Delta}\dot{\perp}\text{-l}$$

$$\frac{\Gamma, (P\ t)\downarrow \vdash \Delta}{\Gamma, (\dot{\forall}\ P) \vdash \Delta}t\ \dot{\forall}\text{-l} \qquad \frac{\Gamma \vdash (P\ y)\downarrow, \Delta}{\Gamma \vdash (\dot{\forall}\ P), \Delta}\dot{\forall}\text{-r}$$

$$\frac{\Gamma, (P\ y)\downarrow \vdash \Delta}{\Gamma, (\dot{\exists}\ P) \vdash \Delta}\dot{\exists}\text{-l} \qquad \frac{\Gamma \vdash (P\ t)\downarrow, \Delta}{\Gamma \vdash (\dot{\exists}\ P), \Delta}t\ \dot{\exists}\text{-r}$$

où les règles $\dot{\forall}$-r et $\dot{\exists}$-l supposent que y n'apparaît pas libre dans Γ ni dans Δ

FIG. 5.1 – **HOL-λ** : Les règles de déduction de HOL-λ

améliorer la compréhension du mécanisme de substitution, placé au niveau méta dans le λ-calcul classique, et ce afin d'améliorer et de faciliter les implémentations.

En effet, le mécanisme de substitution est loin d'être atomique comme le suggère son traitement au niveau méta. Il est même assez complexe, ce qui a donné lieu à la diversité des calculs. La "famille" $\lambda\sigma$ comporte notamment $\lambda\sigma$ ([ACCL91]), $\lambda\sigma_{SP}$ ([CHL92]) qui est souvent appelé $\lambda\sigma$ et qui est en fait celui que nous allons utiliser ici et $\lambda\sigma_{\Uparrow}$ ([CHL92]). $\lambda\sigma_{\Uparrow}$ atteint l'objectif d'être confluent y compris sur les termes comportant des variables de type substitution.

Un autre objectif, la préservation de la terminaison forte, a fait émerger une autre "famille" de calculs, dans laquelle on retrouve λv ([Les94]) ainsi que λd et λd_n ([Kes96]).

Ces deux objectifs se sont révélés contradictoires, comme le montrent des paires de calculs pour lesquels l'ajout des règles nécessaires à la confluence fait perdre la préservation de la terminaison. On citera ici λs ([KR95]) et λs_e ([KR96]), qui ont la particularité de ne pas comporter de type substitution et d'être infinitaires.

Comparer quelques uns de ces systèmes du point de vue de l'implémentation a été l'objet de mon stage de DEA [Dep97], avec pour résultat que la confluence ou la préservation de la terminaison forte n'a pas de réelle influence sur l'efficacité pratique des calculs.

5.4 $\lambda\sigma$

5.4.1 Le système

Nous allons donc utiliser $\lambda\sigma_{SP}$, que nous appellerons $\lambda\sigma$ par la suite. Ce calcul introduit les substitutions sous forme de listes de termes avec les symboles id, ., $\uparrow$ et $\circ$ pour les construire. Ensuite un nouveau constructeur de termes est introduit _[_] qui permet d'appliquer une substitution explicite à un terme. Les règles de réécriture décrivant l'évaluation du $\lambda\sigma$-calcul sont données en figure 5.2.

Beta	$(\lambda a)b$	$\rightarrow$	$a[b.id]$
Eta	$\lambda(a\ 1)$	$\rightarrow$	b
			if $a =_\sigma b[\uparrow]$
σ-reduction :			
App	$(a\ b)[s]$	$\rightarrow$	$(a[s]\ b[s])$
VarCons	$1[a.s]$	$\rightarrow$	a
Id	$a[id]$	$\rightarrow$	a
Abs	$(\lambda a)[s]$	$\rightarrow$	$\lambda(a[1.(s \circ \uparrow)])$
Clos	$(a[s])[t]$	$\rightarrow$	$a[s \circ t]$
IdL	$id \circ s$	$\rightarrow$	s
ShiftCons	$\uparrow \circ (a.s)$	$\rightarrow$	s
AssEnv	$(s_1 \circ s_2) \circ s_3$	$\rightarrow$	$s_1 \circ (s_2 \circ s_3)$
MapEnv	$(a.s) \circ t$	$\rightarrow$	$a[t].(s \circ t)$
IdR	$s \circ id$	$\rightarrow$	s
VarShift	$1.\uparrow$	$\rightarrow$	id
Scons	$1[s].(\uparrow \circ s)$	$\rightarrow$	s

FIG. 5.2 – Les règles de réécriture du $\lambda\sigma$-calcul

Ce système est confluent sur les termes comportant des variables de type terme, mais pas de type substitution. Il n'est pas fortement terminant sur les termes typés [Mel95], mais il est faiblement terminant, c'est-à-dire qu'il est terminant pour une stratégie adéquate, comme celle qui consiste à σ-normaliser après chaque pas de *Beta* ou *Eta*.

Les sortes simples ne sont plus suffisantes avec les indices de De Bruijn. En effet, on doit donner une sorte non seulement aux termes comme $(\lambda_A\ 1)$ (qui reçoit la sorte $A \rightarrow A$), mais aussi aux termes de la forme 1. Donc, comme décrit dans [DHK00], on a à considérer des sortes de la forme $\Gamma \vdash T$ où T est un type simple et Γ un contexte, c'est-à-dire une liste de types simples permettant de typer les variables libres du terme considéré. On introduit aussi des sortes de la forme $\Gamma \vdash \Delta$ pour typer les substitutions.

5.4.2 Précuisson

A la suite de [DHK00], on utilise une traduction du λ-calcul en $\lambda\sigma$-calcul appelée précuisson. Les variables liées sont traduites par les indices appropriés et les variables libres sont traduites en variables de la théorie du premier ordre, réhaussées par un opérateur $[\uparrow^n]$ approprié selon le contexte dans lequelle elles apparaissent, afin de pouvoir utiliser la substitution du premier ordre. En effet, sans le $[\uparrow^n]$, les variables de la théories du premier ordre sont capturées par les λ.

Définition 5.10 La *précuisson* d'un λ-terme a est le $\lambda\sigma$-terme défini par $a_F = F(a, [\ ])$ où $F(a, l)$ est défini en utilisant la liste de variables l ($[\ ]$ étant la liste vide) par :

- $F((\lambda x.a), l) = \lambda(F(a, x.l))$,
- $F((a\ b), l) = F(a, l)F(b, l)$,
- $F(x, l) = 1[\uparrow^{k-1}]$, si x est la k-ième variable de l
- $F(x, l) = x[\uparrow^n]$ où n est la longueur de l si x est une variable n'apparaissant pas dans l ou une constante.

La précuisson préserve le type et permet d'utiliser une substitution du premier ordre sur les variables de la théorie du premier ordre $\lambda\sigma$. De plus, si $t \to_\beta t'$, alors $t_F \to_{\lambda\sigma} t'_F$ et si $t \to_\eta t'$ alors $t_F \to_{Eta} t'_F$, ce qui permet de simuler une réduction $\beta\eta$ en restant à chaque étape dans l'image de la précuisson et en étant donc capable de reconstruire la réduction $\beta\eta$ à partir de la réduction dans $\lambda\sigma$.

5.5 $HOL_{\lambda\sigma}$

On veut utiliser la logique d'ordre supérieur dans le cadre de la déduction modulo. On y parvient en mettant dans la congruence un calcul de substitutions explicites et un encodage approprié des connecteurs et quantificateurs de la logique d'ordre supérieur [DHK01].

5.5.1 syntaxe

$HOL_{\lambda\sigma}$ est le calcul des séquents modulo défini de la façon suivante :

- La syntaxe du langage consiste en un ensemble infini de sortes de la forme $\Gamma \vdash T$ et $\Gamma \vdash \Delta$ où Γ et Δ sont des séquences de types simples et T est un type simple et contient les symboles de fonction suivants :

1_A^Γ	de sorte	$A.\Gamma \vdash A$
$\alpha_{A\to B,A}^\Gamma$	de rang	$(\Gamma \vdash A \to B, \Gamma \vdash A)\Gamma \vdash B$
$\lambda_{A,B}^\Gamma$	de rang	$(A.\Gamma \vdash B)\Gamma \vdash A \to B$
$[\,]_A^{\Gamma,\Gamma'}$	de rang	$(\Gamma' \vdash A, \Gamma \vdash \Gamma')\Gamma \vdash A$
id^Γ	de sorte	$\Gamma \vdash \Gamma$
$\uparrow_A^\Gamma$	de sorte	$A.\Gamma \vdash \Gamma$
$\cdot_A^{\Gamma,\Gamma'}$	de rang	$(\Gamma \vdash A, \Gamma \vdash \Gamma')\Gamma \vdash A.\Gamma'$
$\circ^{\Gamma,\Gamma',\Gamma''}$	de rang	$(\Gamma \vdash \Gamma'', \Gamma'' \vdash \Gamma')\Gamma \vdash \Gamma'$
$\dot\Rightarrow$	de sorte	$\vdash o \to o \to o$
$\dot\wedge$	de sorte	$\vdash o \to o \to o$
$\dot\vee$	de sorte	$\vdash o \to o \to o$
$\dot\neg$	de sorte	$\vdash o \to o$
$\dot\bot$	de sorte	$\vdash o$
$\dot\forall_A$	de sorte	$\vdash (A \to o) \to o$
$\dot\exists_A$	de sorte	$\vdash (A \to o) \to o$

et un unique symbole de prédicat unaire :

$$\varepsilon \quad \text{de rang} \quad (\vdash o)$$

- La congruence est définie par le système de réécriture de classes noté $\lambda\sigma\mathcal{L}$ et consistant en les règles de réécriture du $\lambda\sigma$-calcul avec les règles logiques $\mathcal{L}$ données en figure 5.3.

$$\begin{aligned}
\varepsilon(\dot\Rightarrow\ x\ y) &\to \varepsilon(x) \Rightarrow \varepsilon(y)\\
\varepsilon(\dot\wedge\ x\ y) &\to \varepsilon(x) \wedge \varepsilon(y)\\
\varepsilon(\dot\vee\ x\ y) &\to \varepsilon(x) \vee \varepsilon(y)\\
\varepsilon(\dot\neg\ x) &\to \neg\varepsilon(x)\\
\varepsilon(\dot\bot) &\to \bot\\
\varepsilon(\dot\forall_T\ x) &\to \forall y\ \varepsilon(x\ y)\\
\varepsilon(\dot\exists_T\ x) &\to \exists y\ \varepsilon(x\ y)
\end{aligned}$$

FIG. 5.3 – Les règles de réécriture $\mathcal{L}$

Comme démontré dans [DHK01], le système $\lambda\sigma\mathcal{L}$ est faiblement terminant et confluent sur les termes ne contenant pas de variables de substitutions et la théorie $HOL_{\lambda\sigma}$ est consistante et la règle de coupure y est redondante.

$HOL_{\lambda\sigma}$ est intentionnellement équivalent à la présentation habituelle de la logique d'ordre supérieur HOL_λ [DHK01]. Cela signifie qu'on n'a pas besoin de considérer comme identiques des fonctions qui calculent le même résultat de façon différente. Ceci est important si on souhaite par la suite extraire un programme d'une preuve réalisée dans ce formalisme, afin de pouvoir en contrôler les propriétés algorithmiques.

Par la suite on utilise $HOL_{\lambda\sigma}$ comme une présentation au premier ordre de la logique d'ordre supérieur.

Notation 5.4 On note $\vdash^{\lambda\sigma}$ les séquents dans $HOL_{\lambda\sigma}$.

5.5.2 Utilisation de la notation classique

Les formules de $HOL_{\lambda\sigma}$ peuvent être dénotées par les formules d'ordre supérieur correspondantes malgré un point de vue légèrement différent comme illustré sur un exemple simple :

$$\forall P\, \exists \tau\, \forall x\, (x \in \tau \Rightarrow P(x)) \tag{5.1}$$

Dans HOL_λ cette proposition s'exprime de la façon suivante :

$$\dot{\forall}\lambda P\, \dot{\exists}\lambda \tau\, \dot{\forall}\lambda x\, (x \in \tau \Rightarrow P(x))$$

En appliquant la précuisson on obtient :

$$\begin{array}{l}\varepsilon(\alpha(\dot{\forall}, \lambda\alpha(\dot{\exists}[\uparrow], \lambda\alpha(\dot{\forall}[\uparrow^2],\\ \lambda\alpha(\alpha(\dot{\Rightarrow}[\uparrow^3], \alpha(\alpha(\in [\uparrow^3], \underline{1}), \underline{2})), \alpha(\underline{3}, \underline{1})))))\end{array}$$

Par $\lambda\sigma\mathcal{L}$-normalisation, et en omettant le symbole de fonction α partout où il ne met pas en évidence que les quantifications sont de premier ordre, on obtient :

$$\forall P\, \exists \tau\, \forall x\, (\varepsilon(x \in \tau) \Rightarrow \varepsilon(\alpha(P, x))) \tag{5.2}$$

Cette forme est très proche de la notation d'ordre supérieur (5.1), à la différence principale que c'est maintenant une proposition du premier ordre. En effet, si on omet d'écrire le symbole de prédication ε et le symbole de fonction α on obtient de nouveau la notation standard. Cela revient à considérer (5.1) comme une notation pour la proposition du premier ordre (5.2).

Chapitre 6

Preuves par récurrence

La preuve par récurrence est une méthode de preuve fondamentale en mathématiques. Sa formalisation la plus générale et naturelle est la récurrence noethérienne. Son usage manuel suit souvent des schémas bien connus, récurrence simple ou complète sur les entiers, récurrence structurelle, ...

Pour sa mécanisation, on a été amené a se poser d'autres questions. En effet la récurrence noethérienne ne s'automatise pas en général et décrire les schémas classiques de façon formelle n'est pas toujours évident, et prouver qu'on s'y ramène l'est encore moins.

Dans le contexte de la réécriture, différentes méthodes ont été créées pour prouver par récurrence. Elles se répartissent en deux familles, récurrence par consistance et récurrence par réécriture. Dans la première famille, on ajoute le but aux hypothèses dans une formalisation adéquate. On vérifie (généralement par complétion) la consistance de l'ensemble, et cette consistance signifie que le but est une conséquence inductive des hypothèses. Dans la deuxième famille, on instancie les variables de récurrence par des ensembles couvrants ([Red90]) ou des ensembles test ([Bou94, BKR95]), et on réécrit le but à l'aide des définitions et des hypothèses de récurrence.

6.1 Récurrence noethérienne

6.1.1 Principe de récurrence noethérienne

On rappelle d'abord les notations et résultats standard sur la récurrence noethérienne [Wec92].

Notation 6.1 On note avec $\triangleq$ les définitions.

Il est équivalent de définir ainsi nos notations ou de définir des prédicats supplémentaires avec des axiomes utilisant l'équivalent logique $\Leftrightarrow$.

Cette équivalence n'est pas vérifiée dans certaines logiques "exotiques", paraconsistantes par exemple.

- Premierement, l'inclusion est modélisée par la proposition :

$$\tau_1 \subseteq \tau_2 \triangleq \forall x\,(x \in \tau_1 \Rightarrow x \in \tau_2).$$

- La propriété pour un élément x d'être *minimal* dans un ensemble τ par rapport à une relation R est défini par :

$$Minimal(R, \tau, x) \triangleq x \in \tau \wedge \neg\exists y\,(y \in \tau \wedge R(x, y)).$$

- Une relation binaire $\mathcal{R}$ est *Noethérienne* où *bien fondée* sur un ensemble τ si tout sous ensemble non vide de τ a au moins un élément minimal pour $\mathcal{R}$:

$$Noeth(R, \tau) \triangleq \forall \tau'((\tau' \subseteq \tau \wedge \exists x\,(x \in \tau')) \Rightarrow \exists y\, Minimal(R, \tau', y)).$$

- Une proposition P est *récurrente* par rapport à la relation R dans l'ensemble τ quand :

$$Rec(P, R, \tau) \triangleq \forall x\,((x \in \tau \wedge \forall y\,((y \in \tau \wedge R(x, y)) \Rightarrow P(y))) \Rightarrow P(x)).$$

- Pour dénoter une proposition vraie sur un ensemble τ, on définit :

$$True(P, \tau) \triangleq \forall x\,(x \in \tau \Rightarrow P(x)).$$

- Le *principe de récurrence noethérienne* pour une proposition P peut maintenant être défini comme :

$$NoethRec(P, R, \tau) \triangleq Rec(P, R, \tau) \Rightarrow True(P, \tau).$$

Proposition 6.1 [Hue86, Wec92] Pour tout R et τ on a :

$$Noeth(R, \tau) \text{ iff } \forall P\,(NoethRec(P, R, \tau)).$$

Ceci permet d'utiliser la proposition $\forall R\,\forall \tau\,(Noeth(R, \tau) \Rightarrow \forall P\,(NoethRec(P, R, \tau)))$, pour rendre le principe de récurrence noethérienne disponible tout en générant explicitement les obligations de preuve de noethérianité.

6.1.2 Conséquence inductive et conséquence inductivement prouvable

Une proposition P valide dans tous les modèles de Herbrand d'une théorie Th_u est appelée une *conséquence inductive* de Th_u, ce qui se note $Th_u \models_{Ind} P$. Une proposition P valide dans tous les modèles d'une théorie Th_u est appelée une *conséquence* de Th_u, ce qui se note $Th_u \models P$. Une théorie telle que $\models_{Ind}$ et $\models$ coïncident est dite *ω-complète*

Quand la théorie Th_u est un ensemble de clauses de Horn, il existe un plus petit modèle de Herbrand (par rapport à l'inclusion) unique qui peut être utilisé comme un représentant canonique. Le cas des axiomes équationnels est encore plus simple puisque le plus petit modèle de Herbrand est l'algèbre initiale $\mathcal{T}(\mathcal{F})/Th_u$. On parle alors de *conséquence initiale*. Une conséquence initiale positive est une conséquence inductive ([BKR95]).

Une *conséquence inductivement prouvable* est une proposition qui peut être formellement dérivée en logique d'ordre supérieur (par exemple en utilisant HOL_λ) avec le principe d'induction et la théorie utilisateur comme contexte.

Une proposition P est donc une *conséquence inductivement prouvable* de Th_u quand

$$\forall R\,\forall \tau\,(Noeth(R, \tau) \Rightarrow \forall P\,(NoethRec(P, R, \tau))), Th_u \vdash P.$$

Les conséquences inductivement prouvables de Th_u sont des conséquences inductives de Th_u, c'est à dire :

$$\begin{array}{l} Th_u \models_{Ind} P \\ \Leftarrow \\ \forall R\,\forall \tau\,(Noeth(R, \tau) \Rightarrow \forall P\,(NoethRec(P, R, \tau))), Th_u \vdash P \end{array}$$

6.1.3 Preuve par récurrence noethérienne

On veut prouver une propriété P par récurrence noethérienne. Le séquent correspondant s'écrit :

$$\forall R\,\forall \tau\,(Noeth(R, \tau) \Rightarrow \forall P\,(NoethRec(P, R, \tau))), Th_u \vdash P$$

La preuve est faite en itérant quatre étapes décrites dans les sections suivantes, chacune de ces étapes nous conduit à faire des choix importants.

- la mise en forme de la propriété vers une propriété universelle est aussi l'occasion de la généraliser si nécessaire,
- la variable de récurrence doit être décisive pour établir la propriété, et notamment son instanciation doit permettre de simplifier le but,
- le schéma d'instanciation doit instancier la variable de récurrence avec un ensemble de valeurs adéquat,
- la relation noethérienne (généralement un quasi-ordre) doit permettre d'obtenir les hypothèses de récurrence nécessaires.

Une façon de choisir simultanément le schéma d'instanciation et la relation noethérienne est de choisir un schéma de récurrence. On peut alors laisser la relation noethérienne partiellement implicite et la preuve de sa noethérianité au niveau méta.

Chacun de ces choix est connu pour être non trivial [Bun01].

6.1.3.1 Mise en forme de la propriété

On va d'abord mettre en forme la propriété à démontrer, pour cela on choisit une propriété Q et éventuellement une propriété Q' telles que $\forall \overline{x}\,(\overline{x} \in \overline{\tau} \Rightarrow Q(\overline{x})) \wedge Q'$ implique P pour des ensembles $\overline{\tau}$ définis par la théorie utilisateur Th_u. Q est la propriété qui va être prouvée par récurrence, Q' est une partie restant à démontrer par d'autres méthodes.

On a donc :

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \vdash P \\ \Longleftarrow \text{[par coupure]} \\ \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall \overline{x}\,(\overline{x} \in \overline{\tau} \Rightarrow Q(\overline{x})) \wedge Q' \vdash P \qquad (6.1) \\ \text{et} \\ \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \vdash \forall \overline{x}\,(\overline{x} \in \overline{\tau} \Rightarrow Q(\overline{x})) \wedge Q', P \qquad (6.2) \end{array}$$

Le séquent (6.1) correspond à la preuve que $\forall \overline{x}\,(\overline{x} \in \overline{\tau} \Rightarrow Q(\overline{x})) \wedge Q'$ implique P, et donc que notre mise en forme est correcte.

On obtient donc le séquent (6.2) comme nouveau but.

Remarque 6.1 En général, on supprimera le P restant par affaiblissement.

Remarque 6.2 Même si P est déjà de la forme $\forall \overline{x}\,(\overline{x} \in \overline{\tau} \Rightarrow Q(\overline{x}))$ on peut avoir à passer par cette étape pour généraliser la propriété à prouver. Si on a par exemple une spécification classique sur les listes :

$$\begin{array}{rcl} append(nil, y) & \rightarrow & y \\ append(cons(x,y),z) & \rightarrow & cons(x, append(y,z)) \\ rev(nil) & \rightarrow & nil \\ rev(cons(x,y)) & \rightarrow & append(rev(y), cons(x,nil)) \\ qrev(nil,x) & \rightarrow & x \\ qrev(cons(x,y),z) & \rightarrow & qrev(y, cons(x,z)) \end{array}$$

rev est la fonction qui permet de renverser une liste. $qrev$ est une fonction qui permet aussi de renverser une liste, mais à l'aide d'un paramètre supplémentaire "accumulateur" afin d'être récursive terminale.

Montrer l'équivalence entre rev et $qrev$, ne peut se faire sous la forme

$$\forall l\,(l \in list \Rightarrow rev(l) \approx qrev(l, nil))$$

parce que l'hypothèse de récurrence obtenue ne permet pas d'aboutir.

En effet, on obtient comme hypothèse de récurrence $rev(l) \approx qrev(l, nil)$, et comme sous-buts :

- $rev(nil) \approx qrev(nil, nil)$, qui se réduit trivialement,
- $rev(cons(x,l)) \approx qrev(cons(x,l), nil)$, qui se réduit
  - en $append(rev(l), cons(x, nil)) \approx qrev(l, cons(x, nil))$,
  - puis en $append(qrev(l, nil), cons(x, nil)) \approx qrev(l, cons(x, nil))$ par l'hypothèse de récurrence,
  - mais on ne peut plus appliquer de règle et la preuve échoue.

On doit donc généraliser en

$$\forall l_1 \forall l_2\,((l_1 \in list \wedge l_2 \in list) \Rightarrow qrev(l_1, l_2) \approx append(rev(l_1), l_2))$$

Trouver une bonne généralisation est un problème complexe qui ne sera pas traité dans cette thèse, mais par exemple dans la thèse de Eric Gascard [Gas02] à qui j'ai emprunté cet exemple.

6.1.3.2 Choix de la variable de récurrence

L'étape précédente nous laisse des buts de la forme :

$$\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \vdash \forall \overline{x}\,(\overline{x} \in \overline{\tau} \Rightarrow Q(\overline{x}))$$

Selon la variable que l'on va maintenant choisir, on pourra avancer plus ou moins vite vers une preuve.

Ce choix est entièrement libre dans la méthode de preuve par récurrence noethérienne et son automatisation constitue un problème important.

Le choix d'un ensemble de variables de récurrence peut être vu comme le choix d'une variable de récurrence dont le type sera le produit cartésien des types des différentes variables.

Ce choix, une fois effectué, se traduit donc simplement de la manière suivante :

$$\begin{array}{l} \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u \\ \vdash \forall \overline{x} \, (\overline{x} \in \overline{\tau} \Rightarrow Q(\overline{x})) \\ \Longleftrightarrow \\ \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u \\ \vdash \forall x_i \, (x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v} \, ((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v}))) \end{array} \quad (6.3)$$

Le choix de la variable de récurrence peut être guidé par la définition, ainsi une définition de l'addition sur les naturels par la droite :

$$\begin{array}{rcl} x+0 & \rightarrow & x \\ x+s(y) & \rightarrow & s(x+y) \end{array}$$

suggère de choisir comme variable de récurrence une variable située à droite du $+$, puisque l'instanciation d'une telle variable permettra d'utiliser les règles de la définition.

6.1.3.3 Choix du schéma d'instanciation

Un schéma d'instanciation est un ensemble de motifs couvrant le type de la variable de récurrence. Instancier la variable de récurrence par chacun de ces motifs va permettre de décomposer le problème afin de le simplifier. Un bon schéma d'instanciation va aussi permettre d'utiliser les hypothèses de récurrence.

Un schéma d'instanciation a la forme générale :

$$SI(\tau_i) = \forall x_i' \, (x_i' \in \tau_i \Rightarrow \bigvee_{j=1\ldots n} \exists \overline{y} \, (\overline{y} \in \overline{\tau_y} \wedge x_i' :=: t_i^j(\overline{y})))$$

Un schéma d'instanciation trivial est donné par la définition du type (constructeurs). D'autres schémas peuvent être déduits de cette définition et d'autres éléments, comme la définition des opérations ou le but à prouver.

Ainsi une définition sur les naturels où $P(s(x))$ est défini en fonction de $P(x)$ suggère un schéma $\{0, s(x)\}$ alors qu'une définition où $P(s(s(x)))$ est défini en fonction de $P(x)$ suggère un schéma $\{0, s(0), s(s(x))\}$. Dans le cas où plusieurs schémas sont suggérés par différents éléments de la définition, il faut choisir ou combiner les schémas, voire trouver ailleurs un schéma moins évident. On notera ici le lien très fort avec la surréduction.

Exemple 6.1 Soit la définition suivante pour la fonction $+$:

$$\begin{array}{rcl} x+0 & \rightarrow & x \\ x+s(y) & \rightarrow & s(x+y) \end{array}$$

le schéma d'instanciation suggéré par cette définition est $\{0, s(x)\}$.

Exemple 6.2 Soit la définition suivante pour la fonction $Pair$:

$$\begin{array}{rcl} Pair(0) & \rightarrow & vrai \\ Pair(s(0)) & \rightarrow & faux \\ Pair(s(s(x))) & \rightarrow & Pair(x) \end{array}$$

le schéma d'instanciation suggéré par cette définition est $\{0, s(0), s(s(x))\}$.

Formellement, on repart de l'étape précédente et on obtient :

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array} \quad (6.3)$$

$\Longleftarrow$ [par *coupure* et affaiblissements]

$$\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \vdash SI(\tau_i) \quad (6.4)$$

et

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i) \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array} \quad (6.5)$$

Le séquent (6.4 représente la preuve de correction du schéma d'instanciation SI, qui devra être prouvée par ailleurs.

On obtient donc le séquent (6.5) comme nouveau but.

Lemme 6.1 Si $SI(\tau_i)$ est un schéma d'instanciation des variables de la sorte τ_i alors pour prouver

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array} \quad (6.3)$$

il suffit de prouver

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i) \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array} \quad (6.5)$$

6.1.3.4 Choix de la relation noethérienne

Le choix de la relation noethérienne $\prec$ détermine les hypothèses de récurrence que l'on va obtenir. En effet on aura une hypothèse de récurrence pour chaque élément relié à x_i' par la relation (inférieur à x_i' dans le cas d'un quasi-ordre).

Formellement, on repart de l'étape précédente et on obtient :

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i) \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array} \quad (6.5)$$

$\Longleftarrow$ [par contraction et instanciation]

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i), \\ Noeth(\prec, \tau_i) \Rightarrow \forall P\,(Ind(P,\prec,\tau_i) \Rightarrow \forall x\,(x \in \tau_i \Rightarrow P(x))), \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array}$$

$\Longleftarrow$ [par *implication à gauche*] et affaiblissements

$$\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i) \vdash Noeth(\prec, \tau_i) \quad (6.6)$$

et

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i), \\ \forall P\,(Ind(P,\prec,\tau_i) \Rightarrow \forall x\,(x \in \tau_i \Rightarrow P(x))), \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array} \quad (6.7)$$

le séquent (6.6) correspond à la preuve que $\prec$ est une relation noethérienne sur τ_i.

On continue donc par :

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i), \\ \forall P\,(Ind(P,\prec,\tau_i) \Rightarrow \forall x\,(x \in \tau_i \Rightarrow P(x))), \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array} \quad (6.7)$$

$\Longleftarrow$ [par instanciation et renommage de variables liées (quantifiées)]

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i), \\ \forall x_i'\,((x_i' \in \tau_i \wedge \forall \underline{x_i'}\,((\underline{x_i'} \in \tau \wedge \underline{x_i'} \prec x_i') \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \\ \quad \Rightarrow Q(\overline{u}, \underline{x_i'}, \overline{v})))) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i', \overline{v}))) \\ \Rightarrow \forall x_i'\,(x_i' \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i', \overline{v}))) \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))\end{array}$$

$\Longleftarrow$ [par *implication à gauche*] et affaiblissements
$\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i),$
$\vdash \forall x'_i\,((x'_i \in \tau_i \wedge \forall \underline{x}'_i\,((\underline{x}'_i \in \tau \wedge \underline{x}'_i \prec x'_i) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v})$
$\Rightarrow Q(\overline{u}, \underline{x}'_i, \overline{v})))) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x'_i, \overline{v})))$
et
$\forall x'_i\,(x'_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x'_i, \overline{v})))$
$\vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v})))$
$\Longleftarrow$ [par *axiome*]
$\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i)$
$\vdash \forall x'_i\,((x'_i \in \tau_i \wedge \forall \underline{x}'_i\,((\underline{x}'_i \in \tau \wedge \underline{x}'_i \prec x'_i) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v})$
$\Rightarrow Q(\overline{u}, \underline{x}'_i, \overline{v})))) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x'_i, \overline{v})))$
$\Longleftarrow$ [par instanciation]
$\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i)$
$\vdash (X'_i \in \tau_i \wedge \forall \underline{x}'_i\,((\underline{x}'_i \in \tau \wedge \underline{x}'_i \prec X'_i) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v})$
$\Rightarrow Q(\overline{u}, \underline{x}'_i, \overline{v})))) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, X'_i, \overline{v})))$
$\Longleftarrow$ [par *implication à droite* et *et à gauche*]
$\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i), X'_i \in \tau_i,$
$\forall \underline{x}'_i\,((\underline{x}'_i \in \tau \wedge \underline{x}'_i \prec X'_i) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, \underline{x}'_i, \overline{v})))$ (6.8)
$\vdash \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, X'_i, \overline{v})))$

Nous pouvons donc conclure :

Lemme 6.2 Si $\prec$ est une relation noethérienne sur τ_i, alors pour prouver

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i) \\ \vdash \forall x_i\,(x_i \in \tau_i \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, x_i, \overline{v}))) \end{array} \quad (6.5)$$

il suffit de prouver

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, SI(\tau_i), X'_i \in \tau_i, \\ \forall \underline{x}'_i\,((\underline{x}'_i \in \tau \wedge \underline{x}'_i \prec X'_i) \Rightarrow \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, \underline{x}'_i, \overline{v}))) \\ \vdash \forall \overline{u} \forall \overline{v}\,((\overline{u} \in \overline{\tau_u} \wedge \overline{v} \in \overline{\tau_v}) \Rightarrow Q(\overline{u}, X'_i, \overline{v}))) \end{array} \quad (6.8)$$

6.2 Récurrence utilisant la réécriture

Les méthodes basées sur la réécriture du premier ordre se classent en deux familles : récurrence par consistance et récurrence par réécriture. Elles ont pour point commun d'utiliser l'ordre bien fondé induit par les règles de réécriture (supposées terminantes) de la théorie utilisateur pour construire leur ordre de récurrence.

6.2.1 Récurrence par réécriture

Dans les méthodes de récurrence par réécriture, on instancie les variables de récurrence par des ensemble couvrants ([Red90]) ou des ensembles test ([Bou94]), et on réécrit le but à l'aide des définitions et des hypothèses de récurrence.

Définition 6.1 ([Red90]) Soit A un ensemble d'axiomes équationnels et $\succ$ un ordre bien fondé stable par substitution. Un ensemble de termes $\{t_i\}_i$ de sorte s est un $\succ$-ensemble couvrant pour s (par rapport à A) si, pour tout terme clos g de sorte s, il existe un t_i et une substitution σ tels que $g =_A \sigma(t_i)$ et $g \succeq \sigma(t_i)$.

Exemple 6.3 Soient $A = \{x+0 = x, x+s(y) = s(x+y)\}$, et $\succ$ l'ordre rpo [Der82] avec pour précédence $+ \succ s \succ 0$.

$\{0, s(x)\}$ est un ensemble $\succ$-couvrant pour Nat.

La notion d'ensemble couvrant est cependant assez vague, spécifiant seulement qu'il faut traiter tous les cas (modulo A). En effet, un ensemble de termes réduit à une variable vérifie la définition. C'est pourquoi Bouhoula, Kounalis et Rusinowitch ont introduit la notion d'ensemble test [BKR95, BR95].

Pour que les ensembles tests soient effectivement utiles, et en particulier pour qu'ils génèrent des buts réductibles, ils ne peuvent comporter de variables qu'à partir d'une certaine profondeur.

Définition 6.2 Soit R un ensemble de règles conditionnelles. On définit :
- profondeur(R) comme la profondeur maximum des membres gauches des règles de R,
- sprofondeur(R) comme la profondeur maximum des positions strictes (non variables) dans les membres gauches des règles de R,

Exemple 6.4 Soit $R = \{x + 0 \to x, x + s(y) \to s(x + y)\}$.
profondeur(R) = 3.
sprofondeur(R) = 2.

Définition 6.3 Soit R un ensemble de règles conditionnelles, soient R_l l'ensemble des règles linéaires à gauche de R et R_{nl} l'ensemble de ses règles non linéaires à gauche.
On définit $D(R)$ comme
- profondeur(R) si profondeur(R_{nl}) < profondeur(R_l) et sprofondeur(R) < profondeur(R_l),
- profondeur(R) + 1 sinon.

Définition 6.4 Si R est un ensemble de règles conditionnelles, alors un *ensemble test* $S(R)$ pour R est un ensemble fini de termes R-irréductibles qui a les propriétés suivantes :

complétude pour tout terme clos R-irréductible s, il existe un terme t dans $S(R)$ et une substitution close σ tels que $\sigma(t) = s$,

transnormalité pour tout terme non-clos t dans $S(R)$ et pour toute position u dans t telle que
- $t|_u$ est un terme non-clos,
- $|u|$ =profondeur(R).

il existe une infinité d'instances closes fortement irréductibles $t_0, t_1, \ldots$ telles que $t_0|_u \neq t_1|_u, \ldots$

couverture tout terme non clos dans $S(R)$ n'a de variables qu'à une profondeur supérieure ou égale à $D(R)$.

Dans le cas de règles portant sur des constructeurs libres, les définitions précédentes peuvent être simplifiées.

Définition 6.5 Soit R un ensemble de règles conditionnelles sur des constructeurs libres.
On définit $D(R)$ comme :
- profondeur(R) − 1 si sprofondeur(R) <profondeur(R) et R est linéaire gauche,
- profondeur(R) sinon.

Définition 6.6 Si R est un ensemble de règles conditionnelles sur des constructeurs libres, alors un *ensemble test* $S(R)$ pour R est un ensemble fini de termes constructeurs R-irréductibles qui a les propriétés suivantes :
- pour tout terme clos R-irréductible s, il existe un terme t dans $S(R)$ et une substitution close σ tels que $\sigma(t) = s$,
- tout terme non clos dans $S(R)$ contient seulement des variables de sortes dont l'ensemble des termes constructeurs clos est infini (il existe au moins un constructeur d'arité au moins 1) et elles ne peuvent apparaître qu'à une profondeur supérieure ou égale à $D(R)$.

Exemple 6.5 Soit $R = \{x + 0 \to x, x + s(y) \to s(x + y)\}$. Un ensemble test pour R est :
$S(R) = \{0, s(x)\}$.

Définition 6.7 Soit H un ensemble d'équations conditionnelles, W un ensemble d'équations et l un terme. On définit la relation de *réécriture supportée* $\mapsto_{H[W]}$ par :

$$l[\sigma(s)] \mapsto_{H[W]} l[\sigma(t)]$$

s'il existe une substitution σ et une équation conditionnelle $a_1 = b_1 \wedge \ldots \wedge a_n = b_n \Rightarrow s = t$ dans H telle que :

1. $\sigma(s) \prec \sigma(t)$,
2. $\forall i \in \{1, \ldots, n\}$ il existe t_i tel que $\sigma(a_i) \mapsto^*_{H \cup W} t_i$ et $\sigma(b_i) \mapsto^*_{H \cup W} t_i$,

La récurrence par ensembles test est réfutationnellement complète quand la théorie est présentée par un ensemble de règles équationnelles convergeant sur les termes clos.

Pour opérationnaliser la méthode sous forme d'un système d'inférence, on définit deux relations. La réécriture supportée permet de réécrire à l'aide d'un ensemble d'équations conditionnelles et en tenant compte des hypothèses de récurrence. La réécriture relaxée permet de réécrire de façon non orientée, si des conditions externes en assurent la validité.

Notation 6.2 On note $a \equiv b$ si a est syntaxiquement identique à b. Soit $\prec$ un ordre noethérien. On note $a \# b$ si $a \not\prec b$ et $a \not\succ b$ et $a \not\equiv b$.

Définition 6.8 Soit H un ensemble d'équations. On définit la relation de *réécriture relaxée* par :

$$g[\theta(u)] \rightsquigarrow g[\theta(v)] \text{ s'il existe } (u = v) \in H \text{ et une substitution } \theta \text{ tels que } \theta(u) \# \theta(v)$$

Définition 6.9 On donne ici un système d'inférence I ([BKR95]) basé sur la notion d'ensemble test. E est l'ensemble des équations à prouver et H l'ensemble des équations déjà réduites que l'on peut utiliser comme hypothèses de récurrence. Soit $\prec$ un ordre noethérien.

générer $(E \cup \{e = e'\}, H) \vdash_I (E \cup (\bigcup_\sigma E_\sigma), H \cup \{e = e'\})$
Pour toute instance-test $\sigma(e) = \sigma(e')$, il existe b tel que :
- soit $\sigma(e) \not\prec \sigma(e')$ et $(\sigma(e) \mapsto_{R[H \cup E \cup e = e']} b)$ et $E_\sigma = \{b = \sigma(e')\}$,
- soit $\sigma(e) \# \sigma(e')$ et $(\sigma(e') \mapsto_{R[H \cup E \cup e = e']} b)$ et $E_\sigma = \{\sigma(e) = b\}$,
- soit $\sigma(e) \equiv \sigma(e')$ et $E_\sigma = \emptyset$.

simplifier$_1$ $(E \cup \{a = b\}, H) \vdash_I (E \cup \{a' = b\}, H)$
Si $a \mapsto_{R[H \cup E \cup \{a=b\}]} a'$.

simplifier$_2$ $(E \cup \{a = b\}, H \cup \{g = h\}) \vdash_I (E \cup \{a' = b\}, H \cup \{g = h\})$
Si $a[\sigma(g)]_u \mapsto_H a' \equiv a[\sigma(h)]_u$ et $(\sigma(g) \lhd a$ ou $g \prec h)$.

simplifier$_3$ $(E \cup \{c = d\} \cup \{a = b\}, H) \vdash_I (E \cup \{c = d\} \cup \{a' = b\}, H)$
Si $a[\sigma(c)]_u \mapsto_E a' \equiv a[\sigma(d)]_u$ et $\sigma(c) \lhd a$.

simplifier$_4$ $(E \cup \{a = b\}, H) \vdash_I (E \cup \{a' = b\}, H)$
Si $a \rightsquigarrow a'$, $a \# b$ et $a' \preceq b$.

effacer $(E \cup \{a = a\}, H) \vdash_I (E, H)$

échec $(E \cup \{e = e'\}, H) \vdash_I \square$
S'il y a une instance-test $\sigma(e) = \sigma(e')$ telle que $\sigma(e) \not\equiv \sigma(e')$ et :
- soit $\sigma(e) \succ \sigma(e')$ et $\sigma(e) \not\downarrow_{R[H \cup E \cup \{a=b\}]}$,
- soit $\sigma(e) \# \sigma(e')$ et $\sigma(e) \not\downarrow_{R[H \cup E \cup \{a=b\}]}$ et $\sigma(e') \not\downarrow_{R[H \cup E \cup \{a=b\}]}$.

Ce système d'inférence est correct et réfutationnellement complet dans le cas où R est un système équationnel convergeant [BKR95].

Exemple 6.6 Voici un exemple de spécification Spike :

```
specification : plus_droite
sorts Nat;

constructors :
  0 : -> Nat;
  s_ : Nat -> Nat;
```

```
defined functions :
  _+_ : Nat Nat -> Nat;

axioms :
  x + 0    -> x;
  x + s(y) -> s(x + y);
```

Exemple 6.7 Voici le script de preuve généré par Spike pour le but $0 + x = x$ avec la spécification précédente :

```
All the rules are oriented !

test set of R :

 -> Nat = {0 ; s(x1)}

induction positions of functions:

 -> + : [[2]]

E0 = {0 + x1 = x1}

Application of generate on:
    0 + x1 = x1
 with test substitutions:
    x1 -> {0; s(x1)}

 1) 0 = 0 ;
 2) s(0 + x1) = s(x1)

Delete  0 = 0

E1 = {s(0 + x1) = s(x1)}

H1 = {0 + x1 = x1}

Simplification of:
    s(0 + x1) = s(x1) by H1:
    s(x1) = s(x1)

E2 = {s(x1) = s(x1)}

H2 = {0 + x1 = x1}

Delete  s(x1) = s(x1)

The initial conjectures are inductive theorems of R.
```

Exemple 6.8 Voici le script de réfutation généré par Spike pour le but $0 + s(x) = x$ avec la même spécification :

```
All the rules are oriented !

test set of R :
```

```
 -> Nat = {0 ; s(x1)}

induction positions of functions:

 -> + : [[2]]


E0 = {0 + s(x1) = x1}

Simplification of:
    0 + s(x1) = x1 by R[H0 U E0]:
    s(0 + x1) = x1

E1 = {s(0 + x1) = x1}

H1 = {}

Application of generate on:
    s(0 + x1) = x1
 with test substitutions:
    x1 -> {0; s(x1)}

 1) s(0) = 0 ;
 2) s(s(0 + x1)) = s(x1)

E2 = {s(0) = 0 ;
      s(s(0 + x1)) = s(x1)}

H2 = {s(0 + x1) = x1}

Simplification of:
    s(s(0 + x1)) = s(x1) by H2:
    s(x1) = s(x1)

E3 = {s(0) = 0 ;
      s(x1) = s(x1)}

H3 = {s(0 + x1) = x1}


No inference rule can be applied to:
    s(0) = 0

One of the initial conjectures is not an inductive theorem of R
provided that R is ground convergent.
```

Spike est un système entièrement automatique. La récurrence par réécriture utilisant les ensembles test lui permet de trouver seul des lemmes qu'il faut parfois spécifier à d'autres prouveurs. Il est de plus capable d'utiliser mutuellement les différentes conjectures comme hypothèses de récurrence.

Il est ainsi possible de réaliser automatiquement avec Spike des preuves difficiles, comme le tour de cartes de Gilbreath (voir [Bou94]).

6.2.2 Récurrence par consistance

La récurrence par consistance est issue des travaux de Musser [Mus80]. La méthode a été employée selon plusieurs variantes, notamment par Huet et Hullot [HH82], Jouannaud et Kounalis [JK89] et Bachmair [Bac88]. Elle a été uniformisée par Comon et Nieuwenhuis par l'introduction de la notion de I-axiomatisation [CN00, Com01].

Le principe de la méthode est d'ajouter ajouter le but aux hypothèses dans une formalisation adéquate (en présence d'une I-axiomatisation). On vérifie (généralement par un processus de saturation comme la complétion) la consistance de l'ensemble, et cette consistance signifie que le but est une conséquence inductive des hypothèses.

La récurrence par consistance ne sera pas traitée dans cette thèse, mais constitue une perspective intéressante et qui semble accessible en utilisant l'approche développée ici. On rappelle ici la propriété qui est à la base de la méthode de récurrence par consistance :

Définition 6.10 ([CN00]) Soit E un ensemble d'axiomes, et ϕ une équation. Une I-axiomatisation A est telle que $\phi \cup A \cup E$ est consistant si et seulement si ϕ est une conséquence inductive de E.

Exemple 6.9 Soit $E = \{x + 0 = x, x + s(y) = s(x + y)\}$, une I-axiomatisation est par exemple $A = \{0 \neq s(x), s(x) = s(y) \Rightarrow x = y\}$.

A l'aide de cette I-axiomatisation, on prouve $0 + x = x$. En effet, la saturation de $0 + x = x$ avec produit deux paires critiques :

- La première, $0 + 0 = 0$, est trivialement réduite en $0 = 0$.
- La seconde, $0 + s(y) = s(y)$,
  - est d'abord réduite en $s(0 + y) = s(y)$ par E,
  - puis en $s(y) = s(y)$ par ce qui correspond à l'hypothèse de récurrence.

On peut également réfuter $x + x = x$. Dans ce cas on a deux paire critiques à nouveau.

- La première, $0 + 0 = 0$, se réduit trivialement.
- La seconde, $s(x) + s(x) = s(x)$,
  - se réduit en $s(s(x) + x) = s(x)$ par E,
  - et en $s(x) + x = x$ par la I-axiomatisation.

  $s(x) + x = x$ génère de nouvelles paire critiques dont $s(0) + 0 = 0$, qui se réduit en $s(0) = 0$, fournissant ainsi un témoin d'inconsistance.

Chapitre 7

Déduction modulo et raisonnement par récurrence

On montre comment le fait de voir l'utilisation du principe de récurrence noethérienne dans un cadre de déduction modulo permet de comprendre la méthode de récurrence par réécriture.

On montre d'abord comment obtenir et mettre en forme de puissantes hypothèses de récurrence, de façon à les internaliser lorsqu'elles sont équationnelles.

On montre ensuite comment un ordre comme celui utilisé par Spike permet de ne pas vérifier explicitement la condition de récurrence.

On termine par le résumé de la méthode et quelques exemples.

7.1 Utiliser l'hypothèse de récurrence

On veut prouver qu'une propriété P est une conséquence inductivement prouvable d'une théorie Th_u.

On va ici mettre en forme les hypothèses de récurrence, de façon à pouvoir les internaliser par la suite. Inspirés par le comportement de la récurrence par réécriture, et notamment de Spike, on va mettre en évidence un maximum d'hypothèses de récurrence et d'usages de celles-ci.

Notation 7.1 On utilise des majuscules X pour les variables libres comme elles sont libérées par les règles $\forall$-d et $\exists$-g sur les quantificateurs (voir Figure 3.1).

Notation 7.2 Quand une théorie utilisateur est une théorie équationnelle, on a aussi besoin d'avoir $Th_{\approx}$ et $Th_{:=:}$ dans le contexte. Pour cela on note $\mathcal{T}h$ la théorie suivante :

$$\forall R \forall \tau \, (Noeth(R, \tau) \Rightarrow \forall P \, (NoethRec(P, R, \tau))), Th_{\approx}, Th_{:=:}$$

7.1.1 Cas d'un but unique

On veut prouver qu'une propriété $\forall \overline{x} \, (\overline{x} \in \overline{\tau} \Rightarrow (C(\overline{x}) \Rightarrow Q(\overline{x})))$ est une conséquence inductivement prouvable d'une théorie utilisateur Th_u. Par souci de lisibilité, on écrit le cas où on a deux variables et une condition. Les résultats s'étendent naturellement à l'aide de propriétés logiques usuelles, notamment l'équivalence entre $A \Rightarrow (B \Rightarrow C)$ et $(A \wedge B) \Rightarrow C$.

Lemme 7.1 Soient une théorie utilisateur Th_u et un ordre $\prec$ noethérien sur un ensemble τ_x, les relations suivantes sont vérifiées :

$$\begin{array}{ll} & \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u \\ \vdash^{\lambda\sigma} & \forall x \forall y \, (x \in \tau_x \Rightarrow (y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \end{array} \tag{7.1}$$

SSI

$$\begin{array}{ll} & \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} & \forall y \, (y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{7.2}$$

SSI

$$\begin{array}{ll} & \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\ & \forall \underline{x} \forall y \, ((\underline{x} \in \tau_x \wedge \underline{x} \prec X \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y)) \\ \vdash^{\lambda\sigma} & \forall y \, (y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{7.3}$$

Preuve: Voir annexe A □

7.1.2 Cas de buts multiples

Si on a deux propriétés portant sur le même ensemble (ou un ensemble et un sous-ensemble de celui-ci), on peut générer une hypothèse de récurrence pour chaque propriété. Cette possibilité est très puissante, et en effet elle est utilisée dans Spike.

L'hypothèse que les propriétés portent sur un ensemble et son sous-ensemble permet d'obtenir l'équivalence entre

$$\forall x \, (x \in \tau_1 \Rightarrow (C_1(x) \Rightarrow Q_1(x))) \wedge \forall y \, (y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))$$

et

$$\forall x \, (x \in \tau_1 \Rightarrow ((C_1(x) \Rightarrow Q_1(x)) \wedge \forall y \, (y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))))).$$

Lemme 7.2 Soient une théorie utilisateur Th_u et un ordre $\prec$ noethérien sur un ensemble τ_1, et soit τ_2 un sous-ensemble de τ_1, les relations suivantes sont vérifiées :

$$\begin{array}{ll} & \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u, \forall x \, (x \in \tau_2 \Rightarrow x \in \tau_1) \\ \vdash^{\lambda\sigma} & \forall x \, (x \in \tau_1 \Rightarrow (C_1(x) \Rightarrow Q_1(x))) \wedge \forall y \, (y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \tag{7.4}$$

SSI

$$\begin{array}{ll} & \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u, \forall x \, (x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1 \\ \vdash^{\lambda\sigma} & (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y \, (y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \tag{7.5}$$

SSI

$$\begin{array}{ll} & \forall R \forall \tau \, (Noeth(R,\tau) \Rightarrow \forall P \, (NoethRec(P,R,\tau))), Th_u, \forall x \, (x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1, \\ & \forall \underline{x} \, ((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y} \, ((\underline{y} \in \tau_2 \wedge \underline{y} \prec X \wedge C_2(\underline{y})) \Rightarrow Q_2(\underline{y})) \\ \vdash^{\lambda\sigma} & (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y \, (y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \tag{7.6}$$

Preuve: Voir annexe B □

7.2 Internalisation

Lorsque les propriétés Q à prouver sont des théorèmes équationnels de la forme $u \approx v$, les hypothèses de récurrence exhibées à la section précédente ont exactement la forme des théories canoniques que nous avons définies au chapitre 3, et peuvent donc être internalisées dans la déduction modulo grâce au lemme 3.2.

Soit

$$Q(x,y) = t_1(x,y) \approx t_2(x,y)$$

L'hypothèse de récurrence

$$\forall \underline{x} \forall y \left((\underline{x} \in \tau_x \wedge \underline{x} \prec X \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y) \right)$$

prend la forme

$$\forall \underline{x} \forall y \left((\underline{x} \in \tau_x \wedge \underline{x} \prec X \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow t_1(\underline{x},y) \approx t_2(\underline{x},y) \right)$$

On note :

$$\mathcal{RE}_{Rec(Q,C,x)}(X) = \ t_1(\underline{x},y) \approx t_2(\underline{x},y) \text{ si } \underline{x} \in \tau_x \wedge \underline{x} \prec X \wedge y \in \tau_y \wedge C(\underline{x},y)$$

la forme internalisée de cette hypothèse de récurrence.

Remarque 7.1 Selon le contexte, C peut être absent.

Soit

$$Q_2(z,w) = t_1(z,w) \approx t_2(z,w)$$

L'hypothèse de récurrence

$$\forall \underline{z} \forall w \left((\underline{z} \in \tau_z \wedge \underline{z} \prec X \wedge w \in \tau_w \wedge C(\underline{z},w)) \Rightarrow Q(\underline{z},w) \right)$$

obtenue en étendant le lemme 7.2 au cas de deux variables prend la forme

$$\forall \underline{z} \forall w \left((\underline{z} \in \tau_z \wedge \underline{z} \prec X \wedge w \in \tau_w \wedge C_2(\underline{z},w)) \Rightarrow t_1(\underline{z},w) \approx t_2(\underline{z},w) \right)$$

On note :

$$\mathcal{RE}_{Rec(Q_2,Q,C_2,z,x)}(X) = \ t_1(\underline{x},y) \approx t_2(\underline{x},y) \text{ si } \underline{z} \in \tau_z \wedge \underline{z} \prec X \wedge w \in \tau_w \wedge C_2(\underline{z},w)$$

la forme internalisée de cette hypothèse de récurrence.

Remarque 7.2 Selon le contexte, C_2 peut être absent.

Exemple 7.1 On se place dans l'arithmétique de Peano. Les définitions pour cela sont données section 7.5.1.

Soit le but

$$\forall x \left(x \in Nat \Rightarrow 0 + x \approx x \right) \wedge \forall y \forall z \left((y \in Nat \wedge z \in Nat) \Rightarrow (y + z \approx z + y) \right)$$

On peut obtenir les hypothèses de récurrence suivantes pour une récurrence sur x :
- $\mathcal{RE}_{Rec(0+x \approx x,x)}(X) = 0 + \underline{x} \approx \underline{x}$ si $\underline{x} \in Nat \wedge \underline{x} \prec X$
- $\mathcal{RE}_{Rec(y+z \approx z+y,0+x \approx x,y,x)}(X) = \underline{y} + z \approx z + \underline{y}$ si $\underline{y} \in Nat \wedge z \in Nat \wedge \underline{y} \prec X$
- $\mathcal{RE}_{Rec(y+z \approx z+y,0+x \approx x,z,x)}(X) = y + \underline{z} \approx \underline{z} + y$ si $y \in Nat \wedge \underline{z} \in Nat \wedge \underline{z} \prec X$

7.3 L'ordre de récurrence

Un point essentiel de la méthode de preuve par récurrence noethérienne est de trouver une relation noethérienne $\prec$ sur laquelle baser la récurrence. Une relation non transitive étant difficile à utiliser, on utilisera généralement une relation d'ordre.

Une première remarque est que l'ordre ne peut pas être compatible avec la congruence, c'est pourquoi $\prec$ sera un symbole protecteur.

Ensuite, si le système de réécriture de classes conditionnel $\mathcal{RE}_{Th_u}$ est noethérien, un choix possible pour l'ordre de récurrence est l'ordre permettant son orientation, ou un ordre le contenant. C'est en effet le choix qui est fait par les méthodes de récurrence par réécriture.

Pour utiliser la forme équationnelle de l'hypothèse d'induction $\mathcal{RE}_{Rec}(X)$ on doit vérifier la condition $\underline{x} \prec X$. Le théorème 7.1 montre que dès que le but a été simplifié, ce n'est plus nécessaire si on utilise pour $x \prec X$

$$x \prec X \Leftrightarrow t_1(x,y) \approx t_2(x,y) <_e t_1(X,y) \approx t_2(X,y)$$

où $<_e$ est l'ordre utilisé dans Spike [Bou94] et défini ci-dessous.

On donne ici deux définitions de l'ordre. On donnera en annexe C deux autres définitions et les preuves établissant l'équivalence de toutes ces définitions.

Définition 7.1 Soit $<$ un ordre noethérien sur les termes.

$$\mathcal{C}(t \approx t') = \begin{cases} (\{t\},\{t'\}) & \text{if } t' < t \\ (\{t'\},\{t\}) & \text{if } t < t' \\ (\{t,t'\},\{\}) & \text{otherwise} \end{cases}$$

et on définit $<_e$ par

$$a \approx b <_e c \approx d \text{ si } \mathcal{C}(a \approx b) \ll_{lex} \mathcal{C}(c \approx d)$$

où $\ll_{lex}$ est l'extension lexicographique de l'extension de $<$ aux ensembles.

Définition 7.2

$$\begin{array}{ll} & t_1 \approx t_2 >_e t_3 \approx t_4 \\ \Leftrightarrow & \\ & \{t_1,t_2\} \gg \{t_3,t_4\} \\ \vee & t_1 \# t_2 \wedge \bigvee \begin{cases} t_1 = t_3 \wedge t_1 > t_4 \\ t_2 = t_3 \wedge t_2 > t_4 \\ t_1 = t_4 \wedge t_1 > t_3 \\ t_2 = t_4 \wedge t_2 > t_3 \end{cases} \end{array}$$

On donne ici les principales propriétés qui caractérisent $<_e$.

Lemme 7.3 $<_e$ est un ordre noethérien.

Preuve: Trivial par construction puisque l'extension aux ensembles et l'extension lexicographique préservent la noethérianité et que la construction de la complexité place toujours le terme le plus grand en première position lexicographique □

Lemme 7.4 $>_e$ respecte la sémantique de commutativité de $\approx$.
Soient t_1 et t_2 des termes et E une équation.

$$t_1 \approx t_2 >_e E \Rightarrow t_2 \approx t_1 >_e E$$
$$E >_e t_1 \approx t_2 \Rightarrow E >_e t_2 \approx t_1$$

Preuve: Il est facile de vérifier que $C(t_1 \approx t_2) = C(t_2 \approx t_1)$. □

Lemme 7.5 Quasi-stabilité par ajout d'un contexte équationnel.
Soient t_1, t_2 et t des termes.

$$t_1 > t_2 \Rightarrow t_1 \approx t >_e t_2 \approx t$$
$$\text{si } t \neq t_1$$

Preuve: Ceci est aussi assez facile à vérifier avec la définition de C, par cas de la comparaison entre t_1 et t d'une part, et de t_2 et t d'autre part. □

Lemme 7.6 Les réductions des deux termes d'une équation ne commutent pas.

$$t_1 \approx t_2 >_e t'_1 \approx t_2 >_e t'_1 \approx t'_2 \not\Rightarrow t_1 \approx t_2 >_e t_1 \approx t'_2 >_e t'_1 \approx t'_2$$

Preuve: Ceci est facilement vérifié à l'aide de la définition. □

Lemme 7.7 $>_e$ est stable par substitution si $>$ l'est.

Preuve: On vérifie facilement que la construction de la complexité préserve la stabilité. □

$<_e$ permet de traiter le cas de buts non orientables, comme la commutativité. En effet, même si les termes ne sont pas comparables, $<_e$ est capable de comparer les équations. De plus on a :

Lemme 7.8 Soient t et t' deux termes tels que $t' \neq t$.
$t \approx t <_e t' \approx t$

Preuve: Cette propriété est facile à vérifier en remarquant que l'on a des ensembles et non des multi-ensembles. □

Cette propriété permet de toujours pouvoir effectuer un pas qui rend une égalité triviale, même lorsque cela ne serait pas permis par l'ordre sur les termes.

Exemple 7.2 Soit s un symbole de fonction d'arité 1 et $+$ un symbole de fonction d'arité 2. Et un ordre lpo avec la précédence $s < +$.
$s(s(x) + y)$ et $s(y + s(x))$ ne sont pas comparables.
Cependant $<_e$ permet de comparer

$$s(x) + s(y) \approx s(s(x) + y) <_e s(x) + s(y) \approx s(y + s(x))$$

puisque $s(x) + s(y)$ n'est pas comparable à $s(y + s(x))$ et que $s(x) + s(y) > s(s(x) + y)$.

Les résultats suivants terminent l'explication du comportement de la récurrence par réécriture dans notre contexte orienté preuve. Ils affirment que quand on utilise l'ordre utilisé dans Spike, les conditions d'application des hypothèses de récurrence sont toujours satisfaites.

Lemme 7.9 Soit un but $\alpha_1 \approx \alpha_2$, et une règle $t_1 \approx t_2$.
On a $\sigma(t_1) \approx \sigma(t_2) <_e \alpha_1 \approx \alpha_2$ si la règle est utilisée :

1. sur un sous-terme strict de α_1
2. après avoir réduit le but en $\alpha'_1 \approx \alpha_2$
   (a) sur le terme α'_1
   (b) sur un sous-terme strict de α_2
   (c) sur α_2 en tête, si $\alpha_1 \not< \alpha_2$ ou $\alpha_1 > \sigma(t_2)$
3. après avoir réduit le but en $\alpha'_1 \approx \alpha'_2$

Remarque 7.3

- C'est sans perte de généralité que les symétries entre α_1 et α_2 d'une part, et t_1 et t_2 d'autre part ont été omises.
- Le cas 2b n'est pas redondant avec le cas 1 à cause du lemme 7.6.

Preuve: Voir annexe C. □

Théorème 7.1 Soit un but $\alpha_1 \approx \alpha_2$, et une hypothèse de récurrence de la forme :

$$t_1(\underline{x}) \approx t_2(\underline{x}) \text{ si } \underline{x} \in \tau \wedge t_1(\underline{x}) \approx t_2(\underline{x}) <_e t_1(X) \approx t_2(X).$$

La condition $t_1(\underline{x}) \approx t_2(\underline{x}) <_e t_1(X) \approx t_2(X)$ est toujours satisfaite si l'hypothèse est utilisée :

1. sur un sous-terme strict de α_1
2. après avoir réduit le but en $\alpha_1' \approx \alpha_2$
   (a) sur le terme α_1'
   (b) sur un sous-terme strict de α_2
   (c) sur α_2 en tête, si $\alpha_1 \not\leq \alpha_2$ ou $\alpha_1 > \sigma(t_2)$
3. après avoir réduit le but en $\alpha_1' \approx \alpha_2'$

Preuve: Le résultat découle du lemme 7.9 □

Ce théorème va nous permettre de ne pas vérifier explicitement la condition de récurrence, dès lors qu'on va utiliser une stratégie qui en vérifie les hypothèses.

Conjecture 7.1 Le lemme 7.9 n'est plus vrai avec un ordre basé sur un ordre de réduction et non de simplification.

Conjecture 7.2 Le théorème 7.1 reste vrai avec un ordre basé sur un ordre de réduction et non de simplification.

7.4 Résumé de la méthode

A l'aide des résultats précédents, notre méthode va consister à :

1. Internaliser tout ou partie de la théorie utilisateur Th_u,
2. Utiliser les résultats de la section 7.1 pour générer et mettre en forme les hypothèses de récurrence,
3. Internaliser les hypothèses de récurrence,
4. Instancier le but à l'aide d'un schéma d'instanciation,
5. Réduire les sous-buts à l'aide de la théorie utilisateur et des hypothèses de récurrence,
6. Recommencer à partir du point 2 tant que cela est nécessaire.

7.5 Quelques exemples

On montre comment prouver avec notre méthode que, dans l'arithmétique de Peano,
- 0 est un élément neutre à gauche et à droite pour l'addition, ce qui est une preuve simple.
- l'addition est commutative, ce qui nécessite une preuve plus élaborée, puisqu'on a un axiome non orientable.
- l'addition est associative, ce qui nous permettra d'illustrer une récurrence sur deux variables.

On enrichit ensuite les définitions avec les booléens et les fonctions *pair* et *impair* définies de façon croisée.

On montre comment prouver simultanément que $Pair(z + z) = vrai$ et que $s(x) + y \approx s(x + y)$, ce qui nous permettra d'illustrer l'utilisation de $s(x) + y \approx s(x + y)$ comme hypothèse de récurrence pour la preuve de $Pair(z + z) = vrai$.

Enfin on donne la spécification d'un exemple plus élaboré qui peut être prouvé par Spike.

7.5.1 Contexte

D'abord, on met en place la théorie de l'arithmétique de Peano. Soit :

- Une sorte Nat avec $0 \in Nat$, $s(x) \in Nat$ et $x + y \in Nat$ pour tout $x \in Nat$ et $y \in Nat$. Ceci est exprimé par :

$$Th^{sort}_{Nat} = \begin{cases} \forall x\,(x \in Nat \Leftrightarrow \\ \quad (x{:=:}0 \vee \exists y(y \in Nat \wedge x{:=:}s(y)) \vee \exists y\,\exists z(y \in Nat \wedge z \in Nat \wedge x{:=:}y + z))) \end{cases}$$

Remarque 7.4 Ceci pourrait aussi être internalisé en utilisant notre capacité à réécrire des propositions.

- Une définition pour le symbole + exprimée par :

$$Th^{def+}_{Nat} = \begin{cases} \forall x\,(x \in Nat \Rightarrow x + 0 \approx x) \\ \forall x \forall y\,((x \in Nat \wedge y \in Nat) \Rightarrow x + s(y) \approx s(x + y)) \end{cases}$$

En utilisant le lemme 3.2, on peut internaliser Th^{def+}_{Nat} sous la forme :

$$\mathcal{RE}^{def+}_{Nat} = \begin{cases} x + 0 \approx x & \text{si } x \in Nat \\ x + s(y) \approx s(x + y) & \text{si } x \in Nat \wedge y \in Nat \end{cases}$$

$\mathcal{RE}^{def+}_{Nat}$ peut être orienté en un système de réécriture confluent et terminant

$$\vec{\mathcal{RE}}^{def+}_{Nat} = \begin{cases} x + 0 \to x & \text{si } x \in Nat \\ x + s(y) \to s(x + y) & \text{si } x \in Nat \wedge y \in Nat \end{cases}$$

7.5.2 Prouvons que 0 est élément neutre à gauche pour l'addition

On va d'abord prouver une proposition simple, que toutes les méthodes peuvent prouver.
On veut prouver la proposition :

$$\forall x(x \in Nat \Rightarrow 0 + x \approx x)$$

On part donc du séquent :

$$\mathcal{T}h, Th^{sort}_{Nat} \vdash_{\mathcal{RE}^{def+}_{Nat}} \forall x(x \in Nat \Rightarrow 0 + x \approx x)$$

Par récurrence, on obtient :

$$\forall x\,(x \in Nat \Leftrightarrow (x{:=:}0 \vee \exists y(y \in Nat \wedge x{:=:}s(y))))$$

la preuve de se réduit à :

$$\mathcal{T}h, Th^{sort}_{Nat} \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(0+x \approx x, x)(0)}} 0 + 0 \approx 0 \tag{7.7}$$

et

$$\mathcal{T}h, Th^{sort}_{Nat}, Y \in Nat \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(0+x \approx x, x)(s(Y))}} 0 + s(Y) \approx s(Y) \tag{7.8}$$

(7.7) est trivial en utilisant $Th_{\approx}$.
(7.8) est aussi prouvé en utilisant $Th_{\approx}$, avec

$$\begin{array}{rcl} 0 + s(Y) \approx s(Y) & \longrightarrow & s(0 + Y) \approx s(Y) \\ & \longrightarrow & s(Y) \approx s(Y) \end{array}$$

Notons que dans l'utilisation de l'hypothèse de récurrence, grâce au théorème 7.1, la condition

$$0 + \underline{Y} \approx \underline{Y} <_e 0 + s(Y) \approx s(Y)$$

n'a pas à être explicitement vérifiée puisque le but à été réduit auparavant.

7.5.3 Prouvons la commutativité de l'addition

On va maintenant prouver une proposition plus difficile, notamment parce qu'il est nécessaire de pouvoir gérer le fait qu'elle n'est pas orientable.

On veut prouver la proposition :

$$\forall x\,(x \in Nat \Rightarrow \forall y\,(y \in Nat \Rightarrow x + y \approx y + x))$$

Par récurrence sur x on obtient :

$$\mathcal{T}h, Th^{sort}_{Nat} \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+y \approx y+x, x)}(0)} \forall y\,(y \in Nat \Rightarrow 0 + y \approx y + 0) \tag{7.9}$$

et

$$\mathcal{T}h, Th^{sort}_{Nat}, Z \in Nat \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+y \approx y+x, x)}(s(Z))} \forall y\,(y \in Nat \Rightarrow s(Z) + y \approx y + s(Z)) \tag{7.10}$$

(7.9) se réduit facilement à l'exemple précédent.

Pour (7.10), une récurrence sur y nous donne :

$$\begin{array}{l} \mathcal{T}h, Th^{sort}_{Nat}, Z \in Nat \\ \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+y \approx y+x, x)}(s(Z)) \cup \mathcal{RE}_{Rec(s(Z)+y \approx s(y+Z), y)}(0)} s(Z) + 0 \approx s(0 + Z) \end{array} \tag{7.11}$$

et

$$\begin{array}{l} \mathcal{T}h, Th^{sort}_{Nat}, Z \in Nat, V \in Nat \\ \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+y \approx y+x, x)}(s(Z)) \cup \mathcal{RE}_{Rec(s(Z)+y \approx s(y+Z), y)}(S(V))} s(Z) + s(V) \approx s(s(V) + Z) \end{array} \tag{7.12}$$

(7.11) est simple, en utilisant le résultat de l'exemple précédent.

(7.12) est réduit de la façon suivante :

$$\begin{array}{rl} & s(Z) + s(V) \approx s(s(V) + Z) \\ \longrightarrow & s(Z) + s(V) \approx s(Z + s(V)) \\ \longrightarrow & s(s(Z) + V) \approx s(s(Z + V)) \end{array}$$

puis $s(s(Z) + V) \approx s(s(Z + V))$ est prouvé par une récurrence facile sur V et on a terminé.

L'étape difficile ici est d'appliquer l'hypothèse de récurrence. En notant $s \# t$ si s est incomparable avec t, et en utilisant un ordre LPO avec la précédence $0 < s < +$, on a :

$$\begin{array}{lcl} s(Z) + s(V) & \# & s(s(V) + Z) \\ s(Z) + s(V) & > & s(Z + s(V)) \end{array}$$

ce qui nous permet d'orienter ce pas, puisque

$$s(Z) + s(V) \approx s(s(V) + Z) >_e s(Z) + s(V) \approx s(Z + s(V)).$$

On a aussi :

$$\begin{array}{lcl} s(Z) + s(V) & > & Z + s(V) \\ s(V) + s(Z) & > & s(V) + Z \end{array}$$

ce qui permet de vérifier que, comme nous l'assure le theorème 7.1, la condition de récurrence

$$Z + s(V) \approx s(V) + Z <_e s(Z) + s(V) \approx s(V) + s(Z)$$

est bien respectée.

7.5.4 Prouvons l'associativité de l'addition

On veut maintenant prouver l'associativité de l'addition $+$.

On peut la prouver par récurrence noethérienne sur une seule variable, mais on va ici procéder comme Spike, et utiliser une récurrence sur deux variables.

On veut prouver la proposition :

$$\forall x\,(x \in Nat \Rightarrow \forall y\,(y \in Nat \Rightarrow \forall z\,(z \in Nat \Rightarrow x + (y + z) \approx (x + y) + z)))$$

On oriente ce but de gauche à droite, en donnant un statut lexicographique de droite à gauche au symbole $+$.

Une récurrence sur y et z réduit la preuve à :

$$\begin{array}{l} \mathcal{T}h, Th^{sort}_{Nat} \\ \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+(y+z)\approx(x+y)+z,[y,z])}([0,0])} \forall x\,(x \in Nat \Rightarrow x + (0 + 0) \approx (x + 0) + 0) \end{array} \tag{7.13}$$

et

$$\begin{array}{l} \mathcal{T}h, Th^{sort}_{Nat}, Z \in Nat \\ \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+(y+z)\approx(x+y)+z,[y,z])}([0,s(Z)])} \forall x\,(x \in Nat \Rightarrow x + (0 + s(Z)) \approx (x + 0) + s(Z)) \end{array} \tag{7.14}$$

et

$$\begin{array}{l} \mathcal{T}h, Th^{sort}_{Nat}, W \in Nat \\ \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+(y+z)\approx(x+y)+z,[y,z])}([s(W),0])} \forall x\,(x \in Nat \Rightarrow x + (s(W) + 0) \approx (x + s(W)) + 0) \end{array} \tag{7.15}$$

et

$$\begin{array}{l} \mathcal{T}h, Th^{sort}_{Nat}, Z \in Nat, W \in Nat \\ \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}_{Rec(x+(y+z)\approx(x+y)+z,[y,z])}([s(W),s(Z)])} \forall x\,(x \in Nat \Rightarrow x + (s(W) + s(Z)) \approx (x + s(W)) + s(Z)) \end{array} \tag{7.16}$$

(7.13) et (7.15) se terminent de façon triviale, par simplification en utilisant la théorie utilisateur.

(7.14) est réduit de la façon suivante :

$$\begin{array}{ll} & x + (0 + s(Z)) \approx (x + 0) + s(Z) \\ \longrightarrow & x + s(0 + Z) \approx (x + 0) + s(Z) \\ \longrightarrow & s(x + (0 + Z)) \approx (x + 0) + s(Z) \\ \longrightarrow & s((x + 0) + Z) \approx (x + 0) + s(Z) \\ \longrightarrow & s(x + Z) \approx (x + 0) + s(Z) \\ \longrightarrow & s(x + Z) \approx x + s(Z) \\ \longrightarrow & s(x + Z) \approx s(x + Z) \end{array}$$

(7.16) est réduit de la façon suivante :

$$\begin{array}{ll} & x + (s(W) + s(Z)) \approx (x + s(W)) + s(Z) \\ \longrightarrow & x + s(s(W) + Z) \approx (x + s(W)) + s(Z) \\ \longrightarrow & s(x + (s(W) + Z)) \approx (x + s(W)) + s(Z) \\ \longrightarrow & s((x + s(W)) + Z) \approx (x + s(W)) + s(Z) \\ \longrightarrow & s(s(x + W) + Z) \approx (x + s(W)) + s(Z) \\ \longrightarrow & s(s(x + W) + Z) \approx s((x + s(W)) + Z) \\ \longrightarrow & s(s(x + W) + Z) \approx s(s(x + W) + Z) \end{array}$$

7.5.5 Un exemple de définition croisée : *Pair* et *Impair*

Bien que ne posant de problème ni dans notre cadre, ni en récurrence par réécriture, les définitions croisées ne sont pas syntaxiquement possibles dans NQTHM et sont gérées par une syntaxe ad hoc dans ACL2.

On complète les définitions de la section 7.5.1 avec :

$$Th^{sort}_{Bool} = \left\{ \begin{array}{l} \forall x\,(x \in Bool \Leftrightarrow \\ \quad (x{:=:}vrai \vee x{:=:}faux \vee \exists y(y \in Nat \wedge x{:=:}Pair(y)) \vee \exists y(y \in Nat \wedge x{:=:}Impair(y)))) \end{array} \right.$$

$$Th^{def(Pair)}_{Bool} = \left\{ \begin{array}{l} Pair(0) \approx vrai \\ \forall x\,(x \in Nat \Rightarrow Pair(s(x)) \approx Impair(x)) \end{array} \right.$$

$$Th^{def(Impair)}_{Bool} = \left\{ \begin{array}{l} Impair(0) \approx faux \\ \forall x\,(x \in Nat \Rightarrow Impair(s(x)) \approx Pair(x)) \end{array} \right.$$

On veut prouver la proposition

$$\forall z\,(z \in Nat \Rightarrow Pair(z+z) \approx vrai) \wedge \forall x \forall y\,((x \in Nat \wedge y \in Nat) \Rightarrow s(x)+y \approx s(x+y))$$

Une récurrence sur z, en utilisant le lemme 7.2, réduit la preuve à :

$$\begin{array}{l} Th, Th^{sort}_{Nat}, Th^{sort}_{Bool} \\ \vdash \\ \quad \mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}^{def(Pair)}_{Bool} \cup \mathcal{RE}^{def(Impair)}_{Bool} \cup \\ \quad \mathcal{RE}_{Rec(Pair(z+z) \approx vrai, z)}(0) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, x, z)}(0) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, y, z)}(0) \\ Pair(0+0) \approx vrai \wedge \forall x \forall y\,((x \in Nat \wedge y \in Nat) \Rightarrow s(x)+y \approx s(x+y)) \end{array}$$

et

$$\begin{array}{l} Th, Th^{sort}_{Nat}, Th^{sort}_{Bool}, W \in Nat \\ \vdash \\ \quad \mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}^{def(Pair)}_{Bool} \cup \mathcal{RE}^{def(Impair)}_{Bool} \cup \\ \quad \mathcal{RE}_{Rec(Pair(z+z) \approx vrai, z)}(s(W)) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, x, z)}(s(W)) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, y, z)}(s(W)) \\ Pair(s(W)+s(W)) \approx vrai \wedge \forall x \forall y\,((x \in Nat \wedge y \in Nat) \Rightarrow s(x)+y \approx s(x+y)) \end{array}$$

Par calcul des séquents on obtient :

$$\begin{array}{l} Th, Th^{sort}_{Nat}, Th^{sort}_{Bool} \\ \vdash \\ \quad \mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}^{def(Pair)}_{Bool} \cup \mathcal{RE}^{def(Impair)}_{Bool} \cup \\ \quad \mathcal{RE}_{Rec(Pair(z+z) \approx vrai, z)}(0) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, x, z)}(0) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, y, z)}(0) \\ Pair(0+0) \approx vrai \end{array} \tag{7.17}$$

et

$$\begin{array}{l} Th, Th^{sort}_{Nat}, Th^{sort}_{Bool}, W \in Nat \\ \vdash \\ \quad \mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}^{def(Pair)}_{Bool} \cup \mathcal{RE}^{def(Impair)}_{Bool} \cup \\ \quad \mathcal{RE}_{Rec(Pair(z+z) \approx vrai, z)}(s(W)) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, x, z)}(s(W)) \cup \\ \quad \mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, y, z)}(s(W)) \\ Pair(s(W)+s(W)) \approx vrai \end{array} \tag{7.18}$$

et

$$\begin{array}{l} Th, Th^{sort}_{Nat}, Th^{sort}_{Bool}, W \in Nat \\ \vdash_{\mathcal{RE}^{def+}_{Nat} \cup \mathcal{RE}^{def(Pair)}_{Bool} \cup \mathcal{RE}^{def(Impair)}_{Bool}} \\ \forall x \forall y\,((x \in Nat \wedge y \in Nat) \Rightarrow s(x)+y \approx s(x+y)) \end{array} \tag{7.19}$$

(7.17) est trivial et (7.19) est prouvé par une récurrence simple.

(7.18) est réduit de la façon suivante :

$$\begin{array}{ll} & Pair(s(W) + s(W)) \approx vrai \\ \longrightarrow & Pair(s(s(W) + W)) \approx vrai \\ \longrightarrow & Impair(s(W) + W) \approx vrai \\ \longrightarrow & Impair(s(W + W)) \approx vrai \\ \longrightarrow & Pair(W + W) \approx vrai \\ \longrightarrow & vrai \approx vrai \end{array}$$

L'étape intéressante ici est

$$Impair(s(W) + W) \approx vrai \longrightarrow Impair(s(W + W)) \approx vrai$$

Elle correspond à l'utilisation de $\mathcal{RE}_{Rec(s(x)+y \approx s(x+y), Pair(z+z) \approx vrai, y, z)}(s(W))$.
Sa condition de récurrence se vérifie trivialement puisque $W < s(W)$.

7.5.6 Un exemple plus élaboré

On donne cette exemple sous la forme d'une spécification Spike. Les sortes *nat*1 et *nat*2 sont deux représentations des entiers naturels. La sorte *ret* représente une retenue dans l'addition *plus*2*r*.

*nat*1 est la représentation habituelle des entiers avec 0 (noté *zero*1) et *s*, *nat*2 est une représentation des entiers en binaire. L'intérêt d'une telle représentation est que la fonction *plus*2 est plus efficace que la fonction *plus*1.

Les constructeurs de *nat*2 sont *zero*2, *f* et *g*. *f* est à comprendre comme $f(x) = 2x + 1$ et *g* est à comprendre comme $g(x) = 2x+2$. On aurait pu prendre $f(x) = 2x$ et $g(x) = 2x+1$ mais les constructeurs *f* et *zero*2 n'auraient alors pas été libres, ce que Spike ne sait pas traiter.

Le fait d'avoir une retenue allant jusqu'à 2 est nécessaire pour donner des définitions orientables facilement avec un ordre syntaxique.

*conv*12 et *conv*21 sont les fonctions de conversion entre les deux représentations des entiers.

```
specification : spec
sorts nat1, nat2, ret2;

constructors :
  zero1 : -> nat1;
  s_ : nat1 -> nat1;

  zero2 : -> nat2;
  f_ : nat2 -> nat2;
  g_ : nat2 -> nat2;

  r0 : -> ret2;
  r1 : -> ret2;
  r2 : -> ret2;

defined functions :
  plus1__ : nat1 nat1 -> nat1;

  plus2__ : nat2 nat2 -> nat2;
  plus2r___ : nat2 nat2 ret2 -> nat2;

  conv21_ : nat2 -> nat1;
  conv12_ : nat1 -> nat2;
axioms :
  plus1(x,zero1) = x;
```

```
plus1(x,s(y)) = s(plus1(x,y));

plus2(x,y) = plus2r(x,y,r0);

plus2r(x    , zero2, r0) = x;
plus2r(zero2, y    , r0) = y;

plus2r(f(x) , f(y) , r0) = g(plus2r(x,y,r0));
plus2r(f(x) , g(y) , r0) = f(plus2r(x,y,r1));
plus2r(g(x) , f(y) , r0) = f(plus2r(x,y,r1));
plus2r(g(x) , g(y) , r0) = g(plus2r(x,y,r1));

plus2r(zero2, zero2, r1) = f(zero2);
plus2r(f(x) , zero2, r1) = g(x);
plus2r(g(x) , zero2, r1) = f(plus2r(x,zero2,r1));

plus2r(zero2, f(y) , r1) = g(y);
plus2r(f(x) , f(y) , r1) = f(plus2r(x,y,r1));
plus2r(g(x) , f(y) , r1) = g(plus2r(x,y,r1));

plus2r(zero2, g(y) , r1) = f(plus2r(zero2,y,r1));
plus2r(f(x) , g(y) , r1) = g(plus2r(x,y,r1));
plus2r(g(x) , g(y) , r1) = f(plus2r(x,y,r2));

plus2r(zero2, zero2, r2) = g(zero2);
plus2r(f(x) , zero2, r2) = f(plus2r(x,zero2,r1));
plus2r(g(x) , zero2, r2) = g(plus2r(x,zero2,r1));

plus2r(zero2, f(y) , r2) = f(plus2r(zero2,y,r1));
plus2r(f(x) , f(y) , r2) = g(plus2r(x,y,r1));
plus2r(g(x) , f(y) , r2) = f(plus2r(x,y,r2));

plus2r(zero2, g(y) , r2) = g(plus2r(zero2,y,r1));
plus2r(f(x) , g(y) , r2) = f(plus2r(x,y,r2));
plus2r(g(x) , g(y) , r2) = g(plus2r(x,y,r2));

conv21(zero2) = zero1;
conv21(f(x)) = s(plus1(conv21(x),conv21(x)));
conv21(g(x)) = s(s(plus1(conv21(x),conv21(x))));

conv12(zero1) = zero2;
conv12(s(x)) = plus2(conv12(x),f(zero2));
```

Le fait que les deux définitions coïncident est représenté par les buts

$$conv21(plus2(x,y)) = plus1(conv21(x), conv21(y))$$

et

$$conv12(plus1(x,y)) = plus2(conv12(x), conv12(y))$$

mais Spike ne parvient pas à prouver cela directement.

Il y parvient cependant avec quelques lemmes, dont

$$conv21(plus2r(x1,x2,r2)) = s(s(plus1(conv21(x1), conv21(x2))))$$

et

$$conv21(plus2r(x1,x2,r1)) = s(plus1(conv21(x1), conv21(x2))).$$

Ces deux buts sont intéressants parce que Spike n'arrive à prouver aucun des deux, mais qu'il parvient à prouver les deux simultanément.

Chapitre 8

Conclusion et perspectives

Apports de la thèse

Logique du premier ordre comme déduction modulo

On a vu comment les principes de la déduction modulo s'appliquent au calcul des séquents lui-même.

En effet, bien qu'il soit considéré comme déduction pure, le calcul des séquents comporte en lui-même une part de calcul venant à la fois d'éléments laissés implicites, comme la gestion des substitutions ou des ensembles (de formules et de séquents).

On peut de plus ajouter du calcul correspondant à des simplifications par des propriétés connues, comme $A \wedge A = A$. On a vu qu'il faut cependant être attentif à ce que ces ajouts ne posent pas de problèmes de contrôle de la déduction modulo, et que notamment le tiers exclus $A \vee \neg A = \top$ ne peut pas être vu comme un calcul sur lequel aucun contrôle ne serait requis.

Ce dernier point, ainsi que la nécessité d'introduire une notion de subsomption, faisait que le système obtenu par Patrick Viry [Vir98] n'avait pas les propriétés de confluence souhaitées.

Il est à remarquer que la notion de subsomption ne correspond pas à la vision traditionnelle d'arbre de preuve, puisqu'elle réunit deux sous-buts. On se trouve alors avec des preuves qui peuvent être vues sous forme de graphe orienté.

Extension de la déduction modulo

Pour gérer la récurrence noethérienne dans le cadre de la déduction modulo, trois extensions à ce cadre ont été nécessaires :

- La première extension est la prise en compte de règles et axiomes conditionnels, nécessaire pour exprimer que l'hypothèse de récurrence ne peut s'appliquer qu'aux instances plus petites que celle que l'on veut prouver.
  Le principe de cette prise en compte est de lancer récursivement l'évaluation des conditions. Bien que le problème de la mise en œuvre d'une telle approche se pose dans le cas général, il est possible de prendre en compte des cas simples où l'on peut assurer le bon comportement du mécanisme. On pourrait aussi considérer la congruence comme une gestion de contraintes qui ne résoud que partiellement le problème posé, mais le fait avec un bon comportement et notamment en temps fini.
- La deuxième extension est la prise en compte du contexte dans lequel travaille la congruence, notamment pour l'évaluation des conditions.
  Si on reprend l'exemple de la règle axiome on a :

$$\frac{}{\Gamma, P \vdash_C Q}\text{axiome} \quad \text{si } P =_C^{\Gamma} Q$$

  On tient ainsi compte du contexte Γ pour évaluer l'équivalence entre A et B.
  Cette prise en compte du contexte est particulièrement importante pour l'évaluation des conditions dans les hypothèse de récurrence.

- La troisième extension est la prise en compte de symboles protecteurs.
  En effet, laisser interagir librement la congruence et l'ordre noethérien introduit par la récurrence suppose que l'ordre soit compatible avec la congruence, ce qui est très restrictif.
  Par exemple, avec les égalités habituelles $x + 0 \asymp x$ et $x + s(y) \asymp s(x + y)$, $s(x) + 0 > s(s(x))$ devrait être équivalent à $s(x) > s(s(x))$. Or un ordre rpo avec la précédence $+ > s > 0$ donne l'orientation $s(x) + 0 > s(s(x))$ et $s(s(x)) > s(x)$.
  Déclarer le prédicat $>$ représentant l'ordre comme symbole protecteur permet d'éviter ce phénomène.

Déduction modulo et raisonnement par récurrence

On a vu comment le cadre de la déduction modulo permet de comprendre dans un même formalisme récurrence noethérienne et récurrence par réécriture.

On parvient ainsi à voir la récurrence par réécriture comme le résultat de l'internalisation des hypothèses de récurrence en déduction modulo. En effet, comme les hypothèses de récurrence viennent enrichir le système à l'aide duquel le but est réduit dans la récurrence par réécriture, les hypothèses de récurrence internalisées viennent enrichir la congruence qui permet de simplifier les étapes de déduction.

On a vu comment comprendre certaines propriétés de la méthode de récurrence par réécriture, et notamment du système Spike :

- L'utilisation d'un but comme hypothèse de récurrence pour la preuve d'un autre but.
  Cette utilisation se justifie en récurrence noethérienne par la possibilité de générer une hypothèse de récurrence à partir d'un autre but lorsque celui-ci porte sur la même sorte (ou sur une sous-sorte).
- La non vérification de la condition d'ordre liée à l'hypothèse de récurrence.
  Ce comportement est validé par le fait qu'en utilisant le même ordre sur les équations que Spike, et en utilisant une stratégie appropriée qui consiste principalement à d'abord réduire le but par une règle déjà présente avant d'utiliser l'hypothèse de récurrence, la condition d'ordre attachée à l'hypothèse de récurrence est nécessairement vérifiée.
  On a donné ici une preuve syntaxique de cette propriété qui était justifiée de façon sémantique dans la méthode de récurrence par réécriture. Les conditions posées sur l'application de l'hypothèse de récurrence semble également moins restrictives que celles pratiquées par Spike, mais il n'est pas évident que cela ait des conséquences pratiques.

Perspectives

Les perspectives de ce travail sont :

- Premièrement de raffiner et de compléter la compréhension de la récurrence par réécriture, et notamment de comprendre comment regrouper les sous-buts issus des schémas d'instanciation en un seul but multiple.
  En effet, pour pouvoir générer des hypothèses de récurrence fortes entre les différents sous-buts, il faut les regrouper en un but multiple. Mais il faut comprendre ce que deviennent les hypothèses de récurrence lors de ce regroupement, puisqu'elles ont été instanciées lors de l'utilisation du schéma d'instanciation.
  Si on prend par exemple la preuve de commutativité de l'addition, les deux sous-buts sont $0 + y \approx y + 0$ avec l'hypothèse de récurrence $\underline{x} + y \approx y + \underline{x}$ si $\underline{x} \prec 0$ et $s(z) + y \approx y + s(z)$ avec l'hypothèse de récurrence $\underline{x} + y \approx y + \underline{x}$ si $\underline{x} \prec s(z)$.
  On voudrait donc regrouper en $0 + y \approx y + 0 \wedge s(z) + y \approx y + s(z)$, avec comme hypothèses de récurrence $\underline{x} + y \approx y + \underline{x}$ si $\underline{x} \prec 0$ et $\underline{x} + y \approx y + \underline{x}$ si $\underline{x} \prec s(z)$. Mais la preuve formelle d'un tel regroupement reste à trouver.

- Deux points doivent être implémentés :
  - La vue en déduction modulo du calcul des séquents du premier ordre. En effet, cette vue en déduction modulo permet d'isoler un ensemble de règles dont l'application devra être contrôlée par une stratégie, les autres règles pouvant être appliquées sans contrôle selon un processus de normalisation.
  - La méthode de récurrence en déduction modulo. Il faudra pour cela développer un système recherche de preuve basé sur notre méthode. Ce système pourrait utiliser la surréduction, qui correspond bien au principe instanciation-réduction qui apparaît dans la récurrence.
    Ce travail a été commencé et a donné lieu à l'implémentation d'un prototype. Ce prototype devra évoluer pour intégrer un ordre puissant, du type de celui utilisé par Spike, afin de pouvoir gérer des axiomes non orientables.
- La coopération sceptique entre un assistant de preuve comme Coq et un prouveur automatique implémentant la méthode de récurrence en déduction modulo.
  ELAN est un choix de langage d'implémentation naturel parce que des travaux récents [Ngu02] permettent la coopération entre Coq et ELAN dans le cas de la logique équationnelle.
- Une autre voie intéressante est d'étudier d'autres méthodes de preuve par récurrence, notamment la preuve par consistance.
  En effet la récurrence par consistance est également basée sur la réécriture et semble donc pouvoir s'interpréter dans notre cadre, en utilisant également par exemple les I-Axiomatisations de [CN00]. La récurrence par consistance et un processus réfutationnel, et la reconstruction de ses étapes de récurrence s'avère plus difficile que dans le cas de la récurrence par réécriture. Le lien entre les I-Axiomatisations, qui caractérisent le modèle initial, et le principe de récurrence noethérienne doit notamment être précisé.

Annexe A

Preuve du lemme 7.1

Lemme 7.1 Soient une théorie utilisateur Th_u et un ordre $\prec$ noethérien sur un ensemble τ_x, les relations suivantes sont vérifiées :

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\ \vdash^{\lambda\sigma} & \forall x \forall y\,(x \in \tau_x \Rightarrow (y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \end{array} \quad (7.1)$$

$$SSI$$

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} & \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \quad (7.2)$$

$$SSI$$

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\ & \forall \underline{x} \forall y\,((\underline{x} \in \tau_x \wedge \underline{x} \prec X \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y)) \\ \vdash^{\lambda\sigma} & \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \quad (7.3)$$

Preuve de la partie "(7.1) si (7.2)"

Dans $HOL_{\lambda\sigma}$ on a :

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\ \vdash^{\lambda\sigma} & \forall x \forall y\,(x \in \tau_x \Rightarrow (y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \end{array} \quad (7.1)$$

$\Longleftrightarrow$

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\ \vdash^{\lambda\sigma} & \forall x\,(x \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \end{array}$$

$\Longleftarrow$ [par instanciation]

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\ \vdash^{\lambda\sigma} & X \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array}$$

$\Longleftarrow$ [par *implication à droite*]

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} & \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \quad (7.2)$$

Preuve de la partie "(7.2) si (7.1)"

Dans $HOL_{\lambda\sigma}$ on a :

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{7.2}$$

$\Longleftarrow$ [par coupure]

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} \forall x\,(x \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))), \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{A.1}$$

et

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\ \forall x\,(x \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{A.2}$$

On prouve facilement :

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\ \forall x\,(x \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{A.2}$$

$\Longleftarrow$ [par instanciation]

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\ X \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array}$$

$\Longleftarrow$ [par *implication à gauche*]

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} X \in \tau_x, \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array}$$

et

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\ \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array}$$

Donc on continue avec :

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} \forall x\,(x \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))), \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{A.1}$$

$\Longleftarrow$ [par affaiblissement]

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} \forall x\,(x \in \tau_x \Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \end{array}$$

$\Longleftrightarrow$

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\ \vdash^{\lambda\sigma} \forall x \forall y\,(x \in \tau_x \Rightarrow (y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \end{array} \tag{7.1}$$

Preuve de la partie "(7.2) si (7.3)"

Dans $HOL_{\lambda\sigma}$ on a, en déroulant l'application du principe de récurrence, et en mettant en forme l'hypothèse de récurrence obtenue :

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{7.2}$$

$\Longleftarrow \left[\begin{array}{l} \text{par contraction et instanciation,} \\ \text{l'ensemble et la relation de récurrence sont choisis} \end{array}\right]$

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, Noeth(\prec,\tau_x) \Rightarrow \forall P\,(NoethRec(P,\prec,\tau_x)) \\ \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array}$$

$\Longleftarrow$ [par *implication à gauche*]

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\ \vdash^{\lambda\sigma} Noeth(\prec,\tau_x), \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{A.3}$$

et

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \forall P\,(NoethRec(P,\prec,\tau_x)) \\ \vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \end{array} \tag{A.4}$$

Puisqu'on suppose $\prec$ noethérien sur τ_x, on a bien :

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, X\in\tau_x\\ \vdash^{\lambda\sigma} Noeth(\prec,\tau_x), \forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}\qquad (A.3)$$

Donc on continue avec :

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, X\in\tau_x, \forall P\,(NoethRec(P,\prec,\tau_x))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}\qquad (A.4)$$

$\Longleftarrow$ [Par instanciation et renommage de variables liées (quantifiées)]

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, X\in\tau_x,\\ \forall x\,((x\in\tau_x\wedge\forall\underline{x}\,((\underline{x}\in\tau_x\wedge\underline{x}\prec x)\Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(\underline{x},y)\Rightarrow Q(\underline{x},y)))))\\ \Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y))))\\ \Rightarrow\forall x\,(x\in\tau_x\Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y))))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}$$

$\Longleftrightarrow$

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, X\in\tau_x,\\ \forall x\,((x\in\tau_x\wedge\forall\underline{x}\,\forall y\,((\underline{x}\in\tau_x\wedge\underline{x}\prec x\wedge y\in\tau_y\wedge C(\underline{x},y))\Rightarrow Q(\underline{x},y)))\\ \Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y))))\\ \Rightarrow\forall x\,(x\in\tau_x\Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y))))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}$$

$\Longleftarrow$ [par *implication à gauche*]

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, X\in\tau_x\\ \vdash^{\lambda\sigma}\forall x\,((x\in\tau_x\wedge\forall\underline{x}\,\forall y\,((\underline{x}\in\tau_x\wedge\underline{x}\prec x\wedge y\in\tau_y\wedge C(\underline{x},y))\Rightarrow Q(\underline{x},y)))\\ \Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y)))), \forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}\qquad (A.5)$$

et

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, X\in\tau_x,\\ \forall x\,(x\in\tau_x\Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y))))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}\qquad (A.6)$$

On prouve facilement

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, X\in\tau_x,\\ \forall x\,(x\in\tau_x\Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y))))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}\qquad (A.6)$$

$\Longleftarrow$ [par affaiblissement]

$$\begin{array}{l}X\in\tau_x, \forall x\,(x\in\tau_x\Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(x,y)\Rightarrow Q(x,y))))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}$$

$\Longleftarrow$ [par instanciation]

$$\begin{array}{l}X\in\tau_x, X\in\tau_x\Rightarrow\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}$$

$\Longleftarrow$ [par *implication à gauche*]

$$X\in\tau_x\vdash^{\lambda\sigma} X\in\tau_x, \forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))$$

et

$$\begin{array}{l}X\in\tau_x, \forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\\ \vdash^{\lambda\sigma}\forall y\,(y\in\tau_y\Rightarrow(C(X,y)\Rightarrow Q(X,y)))\end{array}$$

Donc on continue avec :

$$\begin{array}{l}
\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \\
\vdash^{\lambda\sigma} \forall x\,((x \in \tau_x \wedge \forall \underline{x} \forall y\,((\underline{x} \in \tau_x \wedge \underline{x} \prec x \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y))) \qquad (A.5) \\
\Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))), \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \\
\Longleftarrow \text{[par affaiblissement]} \\
\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\
\vdash^{\lambda\sigma} \forall x\,((x \in \tau_x \wedge \forall \underline{x} \forall y\,((\underline{x} \in \tau_x \wedge \underline{x} \prec x \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y))) \\
\Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(x,y) \Rightarrow Q(x,y)))) \\
\Longleftarrow \text{[par instanciation]} \\
\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u \\
\vdash^{\lambda\sigma} (X \in \tau_x \wedge \forall \underline{x} \forall y\,((\underline{x} \in \tau_x \wedge \underline{x} \prec x \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y))) \\
\Rightarrow \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \\
\Longleftarrow \text{[par } \textit{implication à droite}\text{]} \\
\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \\
(X \in \tau_x \wedge \forall \underline{x} \forall y\,((\underline{x} \in \tau_x \wedge \underline{x} \prec x \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y))) \\
\vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \\
\Longleftarrow \text{[par } \textit{et à gauche}\text{]} \\
\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\
\forall \underline{x} \forall y\,((\underline{x} \in \tau_x \wedge \underline{x} \prec x \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y)) \qquad (7.3) \\
\vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y)))
\end{array}$$

Preuve de la partie "(7.3) si (7.2)"

Dans $HOL_{\lambda\sigma}$ on a :

$$\begin{array}{l}
\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x, \\
\forall \underline{x} \forall y\,((\underline{x} \in \tau_x \wedge \underline{x} \prec x \wedge y \in \tau_y \wedge C(\underline{x},y)) \Rightarrow Q(\underline{x},y)) \qquad (7.3) \\
\vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y))) \\
\Longleftarrow \text{[par affaiblissement]} \\
\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, X \in \tau_x \qquad (7.2) \\
\vdash^{\lambda\sigma} \forall y\,(y \in \tau_y \Rightarrow (C(X,y) \Rightarrow Q(X,y)))
\end{array}$$

Annexe B

Preuve du lemme 7.2

Lemme 7.2 Soient une théorie utilisateur Th_u et un ordre $\prec$ noethérien sur un ensemble τ_1, et soit τ_2 un sous-ensemble de τ_1, les relations suivantes sont vérifiées :

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1) \\ \vdash^{\lambda\sigma} & \forall x\,(x \in \tau_1 \Rightarrow (C_1(x) \Rightarrow Q_1(x))) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \quad (7.4)$$

$$SSI$$

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1 \\ \vdash^{\lambda\sigma} & (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \quad (7.5)$$

$$SSI$$

$$\begin{array}{ll} & \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1, \\ & \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y}\,((\underline{y} \in \tau_2 \wedge \underline{y} \prec X \wedge C_2(\underline{y})) \Rightarrow Q_2(\underline{y})) \\ \vdash^{\lambda\sigma} & (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \quad (7.6)$$

Preuve: **Preuve de la partie "(7.4) si (7.5)"**

Dans $HOL_{\lambda\sigma}$ on a :

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1) \\ \vdash^{\lambda\sigma} \forall x\,(x \in \tau_1 \Rightarrow (C_1(x) \Rightarrow Q_1(x))) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \quad (7.4)$$

$\Longleftrightarrow$

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1) \\ \vdash^{\lambda\sigma} \forall x\,(x \in \tau_1 \Rightarrow ((C_1(x) \Rightarrow Q_1(x)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))))) \end{array}$$

$\Longleftarrow$ [par instanciation]

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1) \\ \vdash^{\lambda\sigma} X \in \tau_1 \Rightarrow ((C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))) \end{array}$$

$\Longleftarrow$ [par implication à droite]

$$\begin{array}{l} \forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1 \\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))) \end{array} \quad (7.5)$$

Preuve de la partie "(7.4) si (7.5)"

Dans $HOL_{\lambda\sigma}$ on a :

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1\\ \vdash^{\lambda\sigma}(C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\qquad(7.5)$$

$\Longleftarrow$ [par coupure]

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1\\ \vdash^{\lambda\sigma}\forall x\,(x\in\tau_1\Rightarrow((C_1(x)\Rightarrow Q_1(x))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y))))),\\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\qquad(B.1)$$

et

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall x\,(x\in\tau_1\Rightarrow((C_1(x)\Rightarrow Q_1(x))\wedge\forall x\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}(C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\qquad(B.2)$$

On prouve facilement :

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall x\,(x\in\tau_1\Rightarrow((C_1(x)\Rightarrow Q_1(x))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}(C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\qquad(B.2)$$

$\Longleftarrow$ [par instanciation]

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ X\in\tau_1\Rightarrow((C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y))))\\ \vdash^{\lambda\sigma}(C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}$$

$\Longleftarrow$ [par *implication à gauche*]

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \vdash^{\lambda\sigma}X\in\tau_1, (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}$$

et

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\\ \vdash^{\lambda\sigma}(C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}$$

Donc on continue avec :

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1\\ \vdash^{\lambda\sigma}\forall x\,(x\in\tau_1\Rightarrow((C_1(x)\Rightarrow Q_1(x))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y))))),\\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\qquad(B.1)$$

$\Longleftarrow$ [par affaiblissement]

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1)\\ \vdash^{\lambda\sigma}\forall x\,(x\in\tau_1\Rightarrow((C_1(x)\Rightarrow Q_1(x))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\end{array}$$

$\Longleftrightarrow$

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1)\\ \vdash^{\lambda\sigma}\forall x\,(x\in\tau_1\Rightarrow(C_1(x)\Rightarrow Q_1(x)))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\qquad(7.4)$$

Preuve de la partie "(7.5) si (7.6)"

Dans $HOL_{\lambda\sigma}$ on a :

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1\\ \vdash^{\lambda\sigma}\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\\ \Longleftarrow \text{[par le lemme 7.1]}\\ \forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\\ \Longleftarrow \text{[par coupure]}\\ \forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow(C_1(\underline{x})\Rightarrow Q_1(\underline{x}))),\\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\\ \text{et}\\ \forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow(C_1(\underline{x})\Rightarrow Q_1(\underline{x}))),\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\tag{7.5}$$

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow(C_1(\underline{x})\Rightarrow Q_1(\underline{x}))),\\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\tag{B.3}$$

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow(C_1(\underline{x})\Rightarrow Q_1(\underline{x}))),\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\end{array}\tag{B.4}$$

On prouve facilement :

$$\begin{array}{l}\forall R\,\forall\tau\,(Noeth(R,\tau)\Rightarrow\forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x\in\tau_2\Rightarrow x\in\tau_1), X\in\tau_1,\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow(C_1(\underline{x})\Rightarrow Q_1(\underline{x}))),\\ (C_1(X)\Rightarrow Q_1(X))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))\\ \Longleftarrow \text{[par affaiblissement]}\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow(C_1(\underline{x})\Rightarrow Q_1(\underline{x})))\\ \Longleftarrow \text{[par instanciation]}\\ \forall\underline{x}\,((\underline{x}\in\tau_1\wedge\underline{x}\prec X)\Rightarrow((C_1(\underline{x})\Rightarrow Q_1(\underline{x}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma}\ (\underline{X}\in\tau_1\wedge\underline{X}\prec X)\Rightarrow(C_1(\underline{X})\Rightarrow Q_1(\underline{X}))\\ \Longleftarrow \text{[par instanciation]}\\ (\underline{X}\in\tau_1\wedge\underline{X}\prec X)\Rightarrow((C_1(\underline{X})\Rightarrow Q_1(\underline{X}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y))))\\ \vdash^{\lambda\sigma}\ (\underline{X}\in\tau_1\wedge\underline{X}\prec X)\Rightarrow(C_1(\underline{X})\Rightarrow Q_1(\underline{X}))\\ \Longleftarrow \text{[par implication à droite]}\\ (\underline{X}\in\tau_1\wedge\underline{X}\prec X)\Rightarrow((C_1(\underline{X})\Rightarrow Q_1(\underline{X}))\wedge\forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y)))), \underline{X}\in\tau_1\wedge\underline{X}\prec X\\ \vdash^{\lambda\sigma}\ C_1(\underline{X})\Rightarrow Q_1(\underline{X})\\ \Longleftarrow \text{[par } \textit{implication à gauche}\text{]}\\ \underline{X}\in\tau_1\wedge\underline{X}\prec X\vdash^{\lambda\sigma}\underline{X}\in\tau_1\wedge\underline{X}\prec X, C_1(\underline{X})\Rightarrow Q_1(\underline{X})\\ \text{et}\\ C_1(\underline{X})\Rightarrow Q_1(\underline{X}), \forall y\,(y\in\tau_2\Rightarrow(C_2(y)\Rightarrow Q_2(y))), \underline{X}\in\tau_1\wedge\underline{X}\prec X\vdash^{\lambda\sigma} C_1(\underline{X})\Rightarrow Q_1(\underline{X})\end{array}\tag{B.3}$$

Donc on continue avec :

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow (C_1(\underline{x}) \Rightarrow Q_1(\underline{x}))),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{B.4}$$

$\Longleftrightarrow$

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

$\Longleftarrow$ [par coupure]

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma} \forall \underline{y}\,((\underline{y} \in \tau_1 \wedge \underline{y} \prec X) \Rightarrow \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))),\\ (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{B.5}$$

et

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y}\,((\underline{y} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{B.6}$$

On prouve facilement :

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma} \forall \underline{y}\,((\underline{y} \in \tau_1 \wedge \underline{y} \prec X) \Rightarrow \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))),\\ (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{B.5}$$

$\Longleftarrow$ [par affaiblissement]

$$\begin{array}{l}\forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma} \forall \underline{y}\,((\underline{y} \in \tau_1 \wedge \underline{y} \prec X) \Rightarrow \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))))\end{array}$$

$\Longleftarrow$ [par instanciation]

$$\begin{array}{l}\forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))))\\ \vdash^{\lambda\sigma} (\underline{Y} \in \tau_1 \wedge \underline{Y} \prec X) \Rightarrow \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

$\Longleftarrow$ [par instanciation]

$$\begin{array}{l}(\underline{Y} \in \tau_1 \wedge \underline{Y} \prec X) \Rightarrow ((C_1(\underline{Y}) \Rightarrow Q_1(\underline{Y})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))))\\ \vdash^{\lambda\sigma} (\underline{Y} \in \tau_1 \wedge \underline{Y} \prec X) \Rightarrow \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

$\Longleftarrow$ [par implication à droite]

$$\begin{array}{l}(\underline{Y} \in \tau_1 \wedge \underline{Y} \prec X) \Rightarrow ((C_1(\underline{Y}) \Rightarrow Q_1(\underline{Y})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))), \underline{Y} \in \tau_1 \wedge \underline{Y} \prec X\\ \vdash^{\lambda\sigma} \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

$\Longleftarrow$ [par *implication à gauche*]

$$\begin{array}{l}\underline{Y} \in \tau_1 \wedge \underline{Y} \prec X\\ \vdash^{\lambda\sigma} \underline{Y} \in \tau_1 \wedge \underline{Y} \prec X, \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

et

$$\begin{array}{l}C_1(\underline{Y}) \Rightarrow Q_1(\underline{Y}), \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))), \underline{Y} \in \tau_1 \wedge \underline{X} \prec Y\\ \vdash^{\lambda\sigma} \forall x\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

Donc on continue avec :

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y}\,((\underline{y} \in \tau_1 \wedge \underline{y} \prec X) \Rightarrow \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))))) \\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{B.6}$$

$\Longleftrightarrow$

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y} \forall y\,((\underline{y} \in \tau_1 \wedge \underline{y} \prec X \wedge y \in \tau_2 \wedge C_2(y)) \Rightarrow Q_2(y)),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))))) \\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

$\Longleftarrow$

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y}\,((\underline{y} \in \tau_2 \wedge \underline{y} \prec X \wedge C_2(\underline{y})) \Rightarrow Q_2(\underline{y})),\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X) \Rightarrow ((C_1(\underline{x}) \Rightarrow Q_1(\underline{x})) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y))))) \\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array}$$

$\Longleftarrow$ [par affaiblissement]

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y}\,((\underline{y} \in \tau_2 \wedge \underline{y} \prec X \wedge C_2(\underline{y})) \Rightarrow Q_2(\underline{y})) \\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{7.6}$$

Preuve de la partie "(7.6) si (7.5)"

Dans $HOL_{\lambda\sigma}$ on a :

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \forall \underline{x}\,((\underline{x} \in \tau_1 \wedge \underline{x} \prec X \wedge C_1(\underline{x})) \Rightarrow Q_1(\underline{x})), \forall \underline{y}\,((\underline{y} \in \tau_2 \wedge \underline{y} \prec X \wedge C_2(\underline{y})) \Rightarrow Q_2(\underline{y})) \\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{7.6}$$

$\Longleftarrow$ [par affaiblissement]

$$\begin{array}{l}\forall R \forall \tau\,(Noeth(R,\tau) \Rightarrow \forall P\,(NoethRec(P,R,\tau))), Th_u, \forall x\,(x \in \tau_2 \Rightarrow x \in \tau_1), X \in \tau_1,\\ \vdash^{\lambda\sigma} (C_1(X) \Rightarrow Q_1(X)) \wedge \forall y\,(y \in \tau_2 \Rightarrow (C_2(y) \Rightarrow Q_2(y)))\end{array} \tag{7.5}$$

□

Annexe C

Preuves des résultats concernant l'ordre de récurrence

On donne ici plusieurs définitions alternatives pour $<_e$, qui nous serviront à prouver les principaux résultats.

On prouvera qu'elles sont équivalentes à la définition 7.1.

Dans toutes ces définitions, soit $<$ un ordre noethérien sur les termes.

Définition C.1

$$
\begin{array}{ll}
 & t_1 \approx t_2 >_e t_3 \approx t_4 \\
\Leftrightarrow & \\
 & (\\
\hline
 & t_1 > t_2 \wedge t_3 > t_4 \wedge (t_1 > t_3 \vee (t_1 = t_3 \wedge t_2 > t_4)) \\
\vee & t_1 > t_2 \wedge t_3 < t_4 \wedge (t_1 > t_4 \vee (t_1 = t_4 \wedge t_2 > t_3)) \\
\vee & t_1 > t_2 \wedge t_3 = t_4 \wedge t_1 \geq t_4^3 \\
\vee & t_1 > t_2 \wedge t_3 \# t_4 \wedge t_1 > t_3 \wedge t_1 > t_4 \\
\hline
\vee & t_1 < t_2 \wedge t_3 > t_4 \wedge (t_2 > t_3 \vee (t_2 = t_3 \wedge t_1 > t_4)) \\
\vee & t_1 < t_2 \wedge t_3 < t_4 \wedge (t_2 > t_4 \vee (t_2 = t_4 \wedge t_1 > t_3)) \\
\vee & t_1 < t_2 \wedge t_3 = t_4 \wedge t_2 \geq t_4^3 \\
\vee & t_1 < t_2 \wedge t_3 \# t_4 \wedge t_2 > t_3 \wedge t_2 > t_4 \\
\hline
\vee & t_1 = t_2 \wedge t_3 > t_4 \wedge t_2^1 > t_3 \\
\vee & t_1 = t_2 \wedge t_3 < t_4 \wedge t_2^1 > t_4 \\
\vee & t_1 = t_2 \wedge t_3 = t_4 \wedge t_2^1 > t_4^3 \\
\vee & t_1 = t_2 \wedge t_3 \# t_4 \wedge t_2^1 > t_3 \wedge t_2^1 > t_4 \\
\hline
\vee & t_1 \# t_2 \wedge t_3 > t_4 \wedge (t_1 \geq t_3 \vee t_2 \geq t_3) \\
\vee & t_1 \# t_2 \wedge t_3 < t_4 \wedge (t_1 \geq t_4 \vee t_2 \geq t_4) \\
\vee & t_1 \# t_2 \wedge t_3 = t_4 \wedge (t_1 \geq t_4^3 \vee t_2 \geq t_4^3) \\
\vee & t_1 \# t_2 \wedge t_3 \# t_4 \wedge (t_1 > t_3 \vee t_2 > t_3 \vee t_1 > t_4 \vee t_2 > t_4) \\
 & \qquad\qquad \wedge (t_1 \geq t_3 \vee t_2 \geq t_3) \wedge (t_1 \geq t_4 \vee t_2 \geq t_4) \\
 &)
\end{array}
$$

Définition C.2

$$
\begin{array}{ll}
 & t_1 \approx t_2 >_e t_3 \approx t_4 \\
\Leftrightarrow & \\
 & (t_1 > t_3 \vee t_2 > t_3) \wedge (t_1 > t_4 \vee t_2 > t_4) \\
\vee & t_1 \neq t_2 \wedge \bigvee \left\{ \begin{array}{l} t_1 = t_3 \wedge (t_2 > t_4 \vee t_3 = t_4 \vee (t_1 > t_4 \wedge t_1 \# t_2)) \\ t_2 = t_3 \wedge (t_1 > t_4 \vee t_3 = t_4 \vee (t_2 > t_4 \wedge t_1 \# t_2)) \\ t_1 = t_4 \wedge (t_2 > t_3 \vee t_3 = t_4 \vee (t_1 > t_3 \wedge t_1 \# t_2)) \\ t_2 = t_4 \wedge (t_1 > t_3 \vee t_3 = t_4 \vee (t_2 > t_3 \wedge t_1 \# t_2)) \end{array} \right.
\end{array}
$$

On commence par donner une forme simplifiée de $\gg$, dans le cas d'ensembles de deux termes incomparables, puisque c'est le cas le plus difficile rencontré avec $<_e$.

Lemme C.1

$$\begin{array}{ll} & t_1 \# t_2 \wedge t_3 \# t_4 \\ \Rightarrow & (\{t_1, t_2\} \gg \{t_3, t_4\} \\ \Leftrightarrow & ((t_1 > t_3 \vee t_2 > t_3 \vee t_1 > t_4 \vee t_2 > t_4) \\ \wedge & (t_1 \geq t_3 \vee t_2 \geq t_3) \\ \wedge & (t_1 \geq t_4 \vee t_2 \geq t_4))) \end{array}$$

Preuve: Par la définition de $\gg$, on a :

$$\begin{array}{ll} & t_1 \# t_2 \wedge t_3 \# t_4 \\ \Rightarrow & (\{t_1, t_2\} \gg \{t_3, t_4\} \\ \Leftrightarrow & ((t_1 > t_3 \vee t_2 > t_3) \wedge (t_1 > t_4 \vee t_2 > t_4)) \\ \vee & (t_1 = t_3 \wedge t_2 > t_4) \\ \vee & (t_2 = t_3 \wedge t_1 > t_4) \\ \vee & (t_1 = t_4 \wedge t_2 > t_3) \\ \vee & (t_2 = t_4 \wedge t_1 > t_3)) \end{array}$$

En remarquant qu'ils n'introduisent pas d'inconsistance, on ajoute les cas dont on a besoin pour recombiner :

$$\begin{array}{ll} & t_1 \# t_2 \wedge t_3 \# t_4 \\ \Rightarrow & (\{t_1, t_2\} \gg \{t_3, t_4\} \\ \Leftrightarrow & ((t_1 > t_3 \vee t_2 > t_3) \wedge (t_1 > t_4 \vee t_2 > t_4)) \\ \vee & ((t_1 = t_3 \vee t_2 = t_3) \wedge t_2 > t_4) \\ \vee & ((t_1 = t_3 \vee t_2 = t_3) \wedge t_1 > t_4) \\ \vee & ((t_1 = t_4 \vee t_2 = t_4) \wedge t_2 > t_3) \\ \vee & ((t_1 = t_4 \vee t_2 = t_4) \wedge t_1 > t_3) \end{array}$$

On peut maintenant recombiner, pour obtenir :

$$\begin{array}{ll} & t_1 \# t_2 \wedge t_3 \# t_4 \\ \Rightarrow & (\{t_1, t_2\} \gg \{t_3, t_4\} \\ \Leftrightarrow & (((t_1 = t_3 \vee t_2 = t_3) \wedge (t_1 > t_4 \vee t_2 > t_4)) \\ \vee & ((t_1 = t_4 \vee t_2 = t_4) \wedge (t_1 > t_3 \vee t_2 > t_3)) \\ \vee & ((t_1 > t_3 \vee t_2 > t_3) \wedge (t_1 > t_4 \vee t_2 > t_4)))) \end{array}$$

Recombinant encore, on obtient :

$$\begin{array}{ll} & t_1 \# t_2 \wedge t_3 \# t_4 \\ \Rightarrow & (\{t_1, t_2\} \gg \{t_3, t_4\} \\ \Leftrightarrow & (((t_1 > t_4 \vee t_2 > t_4) \wedge (t_1 \geq t_3 \vee t_2 \geq t_3)) \\ \vee & ((t_1 > t_3 \vee t_2 > t_3) \wedge (t_1 \geq t_4 \vee t_2 \geq t_4)))) \end{array}$$

Maintenant, comme on veut une forme conjonctive, on étend pour obtenir :

$$\begin{array}{ll} & t_1 \# t_2 \wedge t_3 \# t_4 \\ \Rightarrow & (\{t_1, t_2\} \gg \{t_3, t_4\} \\ \Leftrightarrow & ((t_1 > t_3 \vee t_2 > t_3 \vee t_1 > t_4 \vee t_2 > t_4) \\ \wedge & (t_1 > t_3 \vee t_2 > t_3 \vee t_1 \geq t_3 \vee t_2 \geq t_3) \\ \wedge & (t_1 > t_4 \vee t_2 > t_4 \vee t_1 \geq t_4 \vee t_2 \geq t_4) \\ \wedge & (t_1 \geq t_3 \vee t_2 \geq t_3 \vee t_1 \geq t_4 \vee t_2 \geq t_4))) \end{array}$$

Un dernier jeu de simplifications nous permet finalement d'obtenir :

$$\begin{array}{ll} & t_1 \# t_2 \wedge t_3 \# t_4 \\ \Rightarrow & (\{t_1, t_2\} \gg \{t_3, t_4\} \\ \Leftrightarrow & ((t_1 > t_3 \vee t_2 > t_3 \vee t_1 > t_4 \vee t_2 > t_4) \\ \wedge & (t_1 \geq t_3 \vee t_2 \geq t_3) \\ \wedge & (t_1 \geq t_4 \vee t_2 \geq t_4))) \end{array}$$

□

Lemme C.2 La définition C.1 est équivalente à la définition 7.1.

Preuve: Cela vient facilement en étendant $t_1 \approx t_2 >_e t_3 \approx t_4$ depuis la définition 7.1. On procède par cas :

- si $t_1 > t_2$, on a $C(t_1 \approx t_2) = (\{t_1\}, \{t_2\})$
  - si $t_3 > t_4$, on a $C(t_3 \approx t_4) = (\{t_3\}, \{t_4\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_1 > t_3 \vee (t_1 = t_3 \wedge t_2 > t_4)$$
  - si $t_3 < t_4$, on a $C(t_3 \approx t_4) = (\{t_4\}, \{t_3\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_1 > t_4 \vee (t_1 = t_4 \wedge t_2 > t_3)$$
  - si $t_3 = t_4 = t_4^3$, on a $C(t_3 \approx t_4) = (\{t_4^3\}, \{\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_1 \geq t_4^3$$
  - si $t_3 \# t_4$, on a $C(t_3 \approx t_4) = (\{t_3, t_4\}, \{\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_1 > t_3 \wedge t_1 > t_4$$
- si $t_1 < t_2$, on a $C(t_1 \approx t_2) = (\{t_2\}, \{t_1\})$
  - si $t_3 > t_4$, on a $C(t_3 \approx t_4) = (\{t_3\}, \{t_4\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2 > t_3 \vee (t_2 = t_3 \wedge t_1 > t_4)$$
  - si $t_3 < t_4$, on a $C(t_3 \approx t_4) = (\{t_4\}, \{t_3\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2 > t_4 \vee (t_2 = t_4 \wedge t_1 > t_3)$$
  - si $t_3 = t_4 = t_4^3$, on a $C(t_3 \approx t_4) = (\{t_4^3\}, \{\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2 \geq t_4^3$$
  - si $t_3 \# t_4$, on a $C(t_3 \approx t_4) = (\{t_3, t_4\}, \{\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2 > t_3 \wedge t_2 > t_4$$
- si $t_1 = t_2 = t_2^1$, on a $C(t_1 \approx t_2) = (\{t_2^1\}, \{\})$
  - si $t_3 > t_4$, on a $C(t_3 \approx t_4) = (\{t_3\}, \{t_4\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2^1 > t_3$$
  - si $t_3 < t_4$, on a $C(t_3 \approx t_4) = (\{t_4\}, \{t_3\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2^1 > t_4$$
  - si $t_3 = t_4 = t_4^3$, on a $C(t_3 \approx t_4) = (\{t_4^3\}, \{\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2^1 > t_4^3$$
  - si $t_3 \# t_4$, on a $C(t_3 \approx t_4) = (\{t_3, t_4\}, \{\})$, et donc
    $$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_2^1 > t_3 \wedge t_2^1 > t_4$$

- si $t_1 \# t_2$, on a $C(t_1 \approx t_2) = (\{t_1, t_2\}, \{\})$
  - si $t_3 > t_4$, on a $C(t_3 \approx t_4) = (\{t_3\}, \{t_4\})$, et donc

$$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_1 \geq t_3 \vee t_2 \geq t_3$$

  - si $t_3 < t_4$, on a $C(t_3 \approx t_4) = (\{t_4\}, \{t_3\})$, et donc

$$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_1 \geq t_4 \vee t_2 \geq t_4$$

  - si $t_3 = t_4 = t_4^3$, on a $C(t_3 \approx t_4) = (\{t_4^3\}, \{\})$, et donc

$$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow t_1 \geq t_4^3 \vee t_2 \geq t_4^3$$

  - si $t_3 \# t_4$, on a $C(t_3 \approx t_4) = (\{t_3, t_4\}, \{\})$, et donc on est dans le contexte du lemme C.1 et

$$t_1 \approx t_2 >_e t_3 \approx t_4 \Leftrightarrow \{t_1, t_2\} \gg \{t_3, t_4\}$$

On finit la preuve en rassemblant tout. □

Lemme C.3 La définition C.2 est équivalente à la définition C.1.

Preuve: Ceci est facilement vérifié en simplifiant la définition C.1 pour chaque cas de la comparaison de t_1 avec t_2 et de t_3 avec t_4. □

Lemme C.4 La définition 7.2 est équivalente à la définition C.2.

Preuve: Ceci est facilement vérifié sur les définitions C.2 et 7.2. □

Lemme C.5 Quand les équations ont un terme en commun, on obtient une définition simplifiée dont on se servira pour montrer les résultats principaux.

Soient t_1, t_2 et t des termes.

$t_1 \approx t >_e t_3 \approx t \Leftrightarrow t_1 \neq t \wedge (t_1 > t_3 \vee t_3 = t \vee (t_3 < t \wedge t_1 \# t))$

Preuve: Ceci est facilement vérifié avec la définition C.2. □

Lemme C.6 Soit un but $\alpha_1 \approx \alpha_2$, et une règle $t_1 \approx t_2$.

On a $\sigma(t_1) \approx \sigma(t_2) <_e \alpha_1 \approx \alpha_2$ si la règle est utilisée :

1. sur un sous-terme strict de α_1
2. après avoir réduit le but en $\alpha_1' \approx \alpha_2$
   (a) sur le terme α_1'
   (b) sur un sous-terme strict de α_2
   (c) sur α_2 en tête, si $\alpha_1 \not< \alpha_2$ ou $\alpha_1 > \sigma(t_2)$
3. après avoir réduit le but en $\alpha_1' \approx \alpha_2'$

Preuve: On suit les cas de l'énoncé :

1. La règle est utilisée sur un sous-terme strict de α_1.
   On a par hypothèse :

$$\alpha_1 \approx \alpha_2 = \alpha_1[\sigma(t_1)]_\omega \approx \alpha_2$$

   et puisque $\alpha_1[\sigma(t_1)]_\omega \approx \alpha_2 \rightarrow \alpha_1[\sigma(t_2)]_\omega \approx \alpha_2$, on a :

$$\alpha_1[\sigma(t_1)]_\omega \approx \alpha_2 >_e \alpha_1[\sigma(t_2)]_\omega \approx \alpha_2 \tag{C.1}$$

   Par la propriété de sous-terme de l'ordre de simplification, on a :

$$\alpha_1[\sigma(t_1)]_\omega > \sigma(t_1)$$
$$\alpha_1[\sigma(t_2)]_\omega > \sigma(t_2)$$

Donc

$$\alpha_1 > \sigma(t_1) \text{ by } \alpha_1 = \alpha_1[\sigma(t_1)]_\omega > \sigma(t_1) \tag{C.2}$$

et par le lemme C.5 appliqué à (C.1), on peut déterminer deux cas :
- Soit $\alpha_1[\sigma(t_1)]_\omega > \alpha_1[\sigma(t_2)]_\omega$.
  Dans ce cas on a

$$\alpha_1 > \sigma(t_2) \text{ par } \alpha_1 = \alpha_1[\sigma(t_1)]_\omega > \alpha_1[\sigma(t_2)]_\omega > \sigma(t_2) \tag{C.3}$$

  Pour résumer on a $\alpha_1 > \sigma(t_1)$ (C.2) et $\alpha_1 > \sigma(t_2)$ (C.3),
  donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
- Soit $\alpha_1[\sigma(t_2)]_\omega \leq \alpha_2$.
  Dans ce cas on a

$$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 \geq \alpha_1[\sigma(t_2)]_\omega > \sigma(t_2) \tag{C.4}$$

  Pour résumer on a $\alpha_1 > \sigma(t_1)$ (C.2) et $\alpha_2 > \sigma(t_2)$ (C.4),
  donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.

2. La règle est utilisée après avoir réduit le but en $\alpha_1' \approx \alpha_2$,

   (a) sur le terme α_1'
   Puisque $\alpha_1 \approx \alpha_2 \to^n \alpha_1' \approx \alpha_2$, on a :

$$\alpha_1 \approx \alpha_2 >_e \alpha_1' \approx \alpha_2 \tag{C.5}$$

   par hypothèse, on a :

$$\alpha_1' \approx \alpha_2 = \alpha_1'[\sigma(t_1)]_\omega \approx \alpha_2$$

   et puisque $\alpha_1'[\sigma(t_1)]_\omega \approx \alpha_2 \to \alpha_1'[\sigma(t_2)]_\omega \approx \alpha_2$, on a :

$$\alpha_1'[\sigma(t_1)]_\omega \approx \alpha_2 >_e \alpha_1'[\sigma(t_2)]_\omega \approx \alpha_2 \tag{C.6}$$

   Par la propriété de sous-terme de l'ordre de simplification, on a :

$$\alpha_1'[\sigma(t_1)]_\omega \geq \sigma(t_1)$$
$$\alpha_1'[\sigma(t_2)]_\omega \geq \sigma(t_2)$$

   Par le lemme C.5 appliqué à (C.5), on a

$$\alpha_1 \neq \alpha_2 \tag{C.7}$$

   et trois cas :
   - Le premier cas est $\alpha_1 > \alpha_1'$.
     Dans ce cas on a

$$\alpha_1 > \sigma(t_1) \text{ par } \alpha_1 > \alpha_1' = \alpha_1'[\sigma(t_1)]_\omega \geq \sigma(t_1) \tag{C.8}$$

     et par le lemme C.5 appliqué à (C.6), on peut déterminer deux sous-cas :
     - Soit $\alpha_1'[\sigma(t_1)]_\omega > \alpha_1'[\sigma(t_2)]_\omega$.
       Dans ce cas on a

$$\alpha_1 > \sigma(t_2) \text{ par } \alpha_1 > \alpha_1' = \alpha_1'[\sigma(t_1)]_\omega > \alpha_1'[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.9}$$

       Pour résumer on a $\alpha_1 > \sigma(t_1)$ (C.8) et $\alpha_1 > \sigma(t_2)$ (C.9),
       donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
     - Soit $\alpha_1'[\sigma(t_2)]_\omega \leq \alpha_2$.
       Dans ce cas on a

$$\alpha_2 \geq \sigma(t_2) \text{ by } \alpha_2 \geq \alpha_1'[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.10}$$

       Pour résumer on a $\alpha_1 > \sigma(t_1)$ (C.8), $\alpha_2 \geq \sigma(t_2)$ (C.10) et $\alpha_1 \neq \alpha_2$ (C.7),
       donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.

- Le deuxième cas serait $\alpha_1' = \alpha_2$, mais il n'est pas possible, puisqu'il contredit (C.6).
- Le troisième cas est $\alpha_1' < \alpha_2 \wedge \alpha_1 \# \alpha_2$.
  Dans ce cas on a

$$\alpha_2 > \sigma(t_1) \text{ par } \alpha_2 > \alpha_1' = \alpha_1'[\sigma(t_1)]_\omega \geq \sigma(t_1) \tag{C.11}$$

  et par le lemme C.5 appliqué à (C.6), on peut déterminer deux cas :
  - Soit $\alpha_1'[\sigma(t_1)]_\omega > \alpha_1'[\sigma(t_2)]_\omega$.
    Dans ce cas on a

$$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 > \alpha_1' = \alpha_1'[\sigma(t_1)]_\omega > \alpha_1'[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.12}$$

    Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.11) et $\alpha_2 > \sigma(t_2)$ (C.12),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
  - Soit $\alpha_1'[\sigma(t_2)]_\omega \leq \alpha_2$.
    Dans ce cas on a

$$\alpha_2 \geq \sigma(t_2) \text{ par } \alpha_2 \geq \alpha_1'[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.13}$$

    Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.11), $\alpha_2 \geq \sigma(t_2)$ (C.13) et $\alpha_1 \# \alpha_2$ (hypothèse),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.

(b) sur un sous-terme strict de α_2
Puisque $\alpha_1 \approx \alpha_2 \to^n \alpha_1' \approx \alpha_2$, on a :

$$\alpha_1 \approx \alpha_2 >_e \alpha_1' \approx \alpha_2 \tag{C.14}$$

par hypothèse, on a :

$$\alpha_1' \approx \alpha_2 = \alpha_1' \approx \alpha_2[\sigma(t_1)]_\omega$$

et puisque $\alpha_1' \approx \alpha_2[\sigma(t_1)]_\omega \to \alpha_1' \approx \alpha_2[\sigma(t_2)]_\omega$, on a :

$$\alpha_1' \approx \alpha_2[\sigma(t_1)]_\omega >_e \alpha_1' \approx \alpha_2[\sigma(t_2)]_\omega \tag{C.15}$$

Par la propriété de sous-terme de l'ordre de simplification, on a :

$$\alpha_2[\sigma(t_1)]_\omega > \sigma(t_1)$$
$$\alpha_2[\sigma(t_2)]_\omega > \sigma(t_2)$$

donc

$$\alpha_2 > \sigma(t_1) \text{ par } \alpha_2 = \alpha_2[\sigma(t_1)]_\omega > \sigma(t_1) \tag{C.16}$$

et par le lemme C.5 appliqué à (C.15), on peut déterminer deux cas :
- $\alpha_2[\sigma(t_1)]_\omega > \alpha_2[\sigma(t_2)]_\omega$.
  Dans ce cas on a

$$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 = \alpha_2[\sigma(t_1)]_\omega > \alpha_2[\sigma(t_2)]_\omega > \sigma(t_2) \tag{C.17}$$

  Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.16) et $\alpha_2 > \sigma(t_2)$ (C.17),
  donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
- Soit $\alpha_2[\sigma(t_2)]_\omega \leq \alpha_1'$.
  Dans ce cas par le lemme C.5 appliqué à (C.14), on peut déterminer deux sous-cas :
  - $\alpha_1 > \alpha_1'$.
    Dans ce cas on a

$$\alpha_1 > \sigma(t_2) \text{ par } \alpha_1 > \alpha_1' \geq \alpha_2[\sigma(t_2)]_\omega > \sigma(t_2) \tag{C.18}$$

    Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.16) et $\alpha_1 > \sigma(t_2)$ (C.18),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.

- $\alpha_1' \leq \alpha_2$.
  Dans ce cas on a

$$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 \geq \alpha_1' \geq \alpha_2[\sigma(t_2)]_\omega > \sigma(t_2) \tag{C.19}$$

  Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.16) et $\alpha_2 > \sigma(t_2)$ (C.19),
  donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.

(c) sur α_2 en tête, si $\alpha_1 \not< \alpha_2$ ou $\alpha_1 > \sigma(t_2)$
Puisque $\alpha_1 \approx \alpha_2 \rightarrow^n \alpha_1' \approx \alpha_2$, on a :

$$\alpha_1 \approx \alpha_2 >_e \alpha_1' \approx \alpha_2 \tag{C.20}$$

par hypothèse, on a :

$$\alpha_1' \approx \alpha_2 = \alpha_1' \approx \sigma(t_1)$$

et puisque $\alpha_1' \approx \sigma(t_1) \rightarrow \alpha_1' \approx \sigma(t_2)$, on a :

$$\alpha_1' \approx \sigma(t_1) >_e \alpha_1' \approx \sigma(t_2) \tag{C.21}$$

Par le lemme C.5 appliqué à (C.21), on a :

$$\alpha_1 \neq \alpha_2 \tag{C.22}$$

on peut déterminer deux cas :
- $\sigma(t_1) > \sigma(t_2)$.
  Dans ce cas on a

$$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 = \sigma(t_1) > \sigma(t_2) \tag{C.23}$$

  On doit procéder par cas entre α_1 et α_2 :
  - $\alpha_1 = \alpha_2$.
    Ceci contredit (C.20) et donc ne peut pas être le cas.
  - $\alpha_1 \# \alpha_2$.
    Dans ce cas on a $\alpha_2 = \sigma(t_1)$ (hypothèse) $\alpha_2 > \sigma(t_2)$ (C.23) et $\alpha_1 \# \alpha_2$,
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
  - $\alpha_1 > \alpha_2$.
    Dans ce cas on a $\alpha_1 > \sigma(t_1)$ et $\alpha_1 > \sigma(t_2)$,
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
  - $\alpha_1 < \alpha_2$.
    Dans ce cas on a $\alpha_2 > \alpha_1$, $\sigma(t_1) > \sigma(t_2)$, $\alpha_2 = \sigma(t_1)$ et $\alpha_1 > \sigma(t_2)$,
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
- Soit $\sigma(t_2) \leq \alpha_1'$.
  Dans ce cas par le lemme C.5 appliqué à (C.20), on a trois sous-cas :
  - $\alpha_1 > \alpha_1'$.
    Dans ce cas on a

$$\alpha_1 > \sigma(t_2) \text{ par } \alpha_1 > \alpha_1' \geq \sigma(t_2) \tag{C.24}$$

    Pour résumer on a $\alpha_2 = \sigma(t_1)$, $\alpha_1 > \sigma(t_2)$ (C.24) et $\alpha_1 \neq \alpha_2$ (C.22),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
  - $\alpha_1' = \alpha_2$.
    Ceci contredit (C.21) et donc ne peut pas être le cas.
  - $\alpha_1' < \alpha_2 \wedge \alpha_1 \# \alpha_2$.
    Dans ce cas on a

$$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 > \alpha_1' \geq \sigma(t_2) \tag{C.25}$$

    Pour résumer on a $\alpha_2 = \sigma(t_1)$ $\alpha_2 > \sigma(t_2)$ (C.25) et $\alpha_1 \# \alpha_2$ (hypothèse),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.

3. La règle est utilisée après avoir réduit le but en $\alpha'_1 \approx \alpha'_2$.
Puisque $\alpha_1 \approx \alpha_2 \rightarrow^n \alpha'_1 \approx \alpha'_2$, on a :

$$\alpha_1 \approx \alpha_2 >_e \alpha'_1 \approx \alpha'_2 \tag{C.26}$$

par hypothèse, on a :

$$\alpha'_1 \approx \alpha'_2 = \alpha'_1[\sigma(t_1)]_\omega \approx \alpha'_2$$

et puisque $\alpha'_1[\sigma(t_1)]_\omega \approx \alpha'_2 \rightarrow \alpha'_1[\sigma(t_2)]_\omega \approx \alpha'_2$, on a :

$$\alpha'_1[\sigma(t_1)]_\omega \approx \alpha'_2 >_e \alpha'_1[\sigma(t_2)]_\omega \approx \alpha'_2 \tag{C.27}$$

Par la propriété de sous-terme de l'ordre de simplification, on a :

$$\alpha'_1[\sigma(t_1)]_\omega \geq \sigma(t_1)$$
$$\alpha'_1[\sigma(t_2)]_\omega \geq \sigma(t_2)$$

Par la définition C.2 appliquée à (C.26), on peut déterminer trois cas :
- $\alpha_1 > \alpha'_1$.
  Dans ce cas on a
  $$\alpha_1 > \sigma(t_1) \text{ par } \alpha_1 > \alpha'_1 = \alpha'_1[\sigma(t_1)]_\omega \geq \sigma(t_1) \tag{C.28}$$
  et par le lemme C.5 appliqué à (C.27), on peut déterminer deux sous-cas :
  - Soit $\alpha'_1[\sigma(t_1)]_\omega > \alpha'_1[\sigma(t_2)]_\omega$.
    Dans ce cas on a
    $$\alpha_1 > \sigma(t_2) \text{ par } \alpha_1 > \alpha'_1 = \alpha'_1[\sigma(t_1)]_\omega > \alpha'_1[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.29}$$
    Pour résumer on a $\alpha_1 > \sigma(t_1)$ (C.28) et $\alpha_1 > \sigma(t_2)$ (C.29),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
  - Soit $\alpha'_1[\sigma(t_2)]_\omega \leq \alpha'_2$.
    Dans ce cas, en reprenant la définition C.2 appliquée à (C.26), on a deux sous-cas
    * Soit $\alpha_1 > \alpha'_2$.
      Dans ce cas on a
      $$\alpha_1 > \sigma(t_2) \text{ par } \alpha_1 > \alpha'_2 \geq \alpha'_1[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.30}$$
      Pour résumer on a $\alpha_1 > \sigma(t_1)$ (C.28) et $\alpha_1 > \sigma(t_2)$ (C.30),
      donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
    * Soit $\alpha_2 > \alpha'_2$.
      Dans ce cas on a
      $$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 > \alpha'_2 \geq \alpha'_1[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.31}$$
      Pour résumer on a $\alpha_1 > \sigma(t_1)$ (C.28) et $\alpha_2 > \sigma(t_2)$ (C.31),
      donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
- $\alpha_2 > \alpha'_1$.
  Dans ce cas on a
  $$\alpha_2 > \sigma(t_1) \text{ par } \alpha_2 > \alpha'_1 = \alpha'_1[\sigma(t_1)]_\omega \geq \sigma(t_1) \tag{C.32}$$
  et par le lemme C.5 appliqué à (C.27), on peut déterminer deux sous-cas :
  - Soit $\alpha'_1[\sigma(t_1)]_\omega > \alpha'_1[\sigma(t_2)]_\omega$.
    Dans ce cas on a
    $$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 > \alpha'_1 = \alpha'_1[\sigma(t_1)]_\omega > \alpha'_1[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.33}$$
    Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.32) et $\alpha_2 > \sigma(t_2)$ (C.33),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.

- Soit $\alpha'_1[\sigma(t_2)]_\omega \leq \alpha'_2$.
  Dans ce cas, en reprenant la définition C.2 appliquée à (C.26), on a deux sous-cas
  * Soit $\alpha_1 > \alpha'_2$.
    Dans ce cas on a

$$\alpha_1 > \sigma(t_2) \text{ par } \alpha_1 > \alpha'_2 \geq \alpha'_1[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.34}$$

    Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.32) et $\alpha_1 > \sigma(t_2)$ (C.34),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
  * Soit $\alpha_2 > \alpha'_2$.
    Dans ce cas on a

$$\alpha_2 > \sigma(t_2) \text{ par } \alpha_2 > \alpha'_2 \geq \alpha'_1[\sigma(t_2)]_\omega \geq \sigma(t_2) \tag{C.35}$$

    Pour résumer on a $\alpha_2 > \sigma(t_1)$ (C.32) et $\alpha_2 > \sigma(t_2)$ (C.35),
    donc $\alpha_1 \approx \alpha_2 >_e \sigma(t_1) \approx \sigma(t_2)$.
- $\alpha_1 \neq \alpha_2$.
  On a quatre sous-cas : $\alpha_1 = \sigma(t_1)$, $\alpha_1 = \sigma(t_2)$, $\alpha_2 = \sigma(t_1)$ and $\alpha_2 = \sigma(t_2)$.
  Chacun d'eux contredit les hypothèses et donc ne peut se produire.

□

Index

Bibliographie

[ACCL91] Martin Abadi, Lucas Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4) :375–416, 1991.

[Bac88] L. Bachmair. Proof by consistency in equational theories. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science, Edinburgh (UK)*, pages 228–233, 1988.

[Bar84] H. P. Barendregt. *The Lambda-Calculus, its syntax and semantics.* Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1984. Second edition.

[BB97] Henk Barendregt and Erik Barendsen. Autartik computations in formal proofs. `ftp ://ftp.cs.kun.nl/pub/CompMath.Found/computations.ps.Z`, 1997.

[BKK+98] Peter Borovanský, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, and Christophe Ringeissen. An overview of ELAN. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the second International Workshop on Rewriting Logic and Applications*, volume 15, `http ://www.elsevier.nl/locate/entcs/volume15.html`, Pont-à-Mousson (France), September 1998. Electronic Notes in Theoretical Computer Science. http ://www.loria.fr/ELAN/.

[BKR95] A. Bouhoula, E. Kounalis, and M. Rusinowitch. Automated Mathematical Induction. *Journal of Logic and Computation*, 5(5) :631–668, 1995. Also available as Technical Report 1636, INRIA, Nancy (France).

[BMM02] Roberto Bruni, José Meseguer, and Ugo Montanari. Tiling transactions in rewriting logic. In Fabio Gadducci and Ugo Montanari, editors, *Proceedings of the 4^{th} International Workshop on Rewriting Logic and Its Applications*, volume 71, pages 43–62, Pisa, Italy, September 2002. Elsevier Science. Preliminary version.

[Bou94] A. Bouhoula. *Preuves Automatiques par Récurrence dans les Théories Conditionnelles.* Thèse de Doctorat d'Université, Université Nancy 1, mars 1994.

[BR95] Adel Bouhoula and Michaël Rusinowitch. Implicit induction in conditional theories. *Journal of Automated Reasoning*, 14(2) :189–235, 1995.

[Bun01] A. Bundy. The automation of proof by mathematical induction. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 13, pages 845–911. Elsevier Science, 2001.

[CHL92] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence Properties of Weak and Strong Calculi of Explicit Substitutions. Technical Report 1617, INRIA, février 1992.

[CM96] Evelyne Contejean and Claude Marché. CiME : Completion modulo E. In Harald Ganzinger, editor, *7^{th} International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 416–419, New Brunswick, NJ, USA, July 1996. Springer-Verlag. System Description available at http ://www.lri.fr/~demons/cime.html.

[CN00] H. Comon and R. Nieuwenhuis. Induction = I-Axiomatization + First-order Consistency. *Information and Computation*, 159(1-2) :151–186, 2000.

[Com01] H. Comon. Inductionless induction. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 14, pages 913–962. Elsevier Science, 2001.

[dB72] N. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 34(5) :381–392, 1972.

[Dep97] Eric Deplagne. Une comparaison d'efficacité entre différents calculs de substitutions explicites. Rapport de DEA, LIFO, Université d'Orléans, September 1997.

[Dep00] Eric Deplagne. Sequent calculus viewed modulo. In Catherine Pilière, editor, *Proceedings of the ESSLLI-2000 Student Session*, pages 66–76, Birmingham, England, August 2000. FoLLI, University of Birmingham.

[Der82] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17 :279–301, 1982.

[DHK98] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. Rapport de Recherche 3400, Institut National de Recherche en Informatique et en Automatique, April 1998. `ftp{:}//ftp.inria.fr/INRIA/publication/RR/RR-3400.ps.gz`.

[DHK00] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher-order unification via explicit substitutions. *Information and Computation*, 157(1/2) :183–235, 2000.

[DHK01] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11(1) :21–45, 2001.

[DO90] N. Dershowitz and M. Okada. A rationale for conditional equational programming. *Theoretical Computer Science*, 75 :111–138, 1990.

[Dow99] Gilles Dowek. *La part du Calcul.* Thèse d'habilitation, Université de Paris 7, 1999.

[Gal86] Jean H. Gallier. *Logic for Computer Science : Foundations of Automatic Theorem Proving*, volume 5 of *Computer Science and Technology Series.* Harper & Row, New York, 1986.

[Gas02] Eric Gascard. *Méthodes pour la Vérification Formelle de systèmes matériels et logiciels à architecture régulière.* Thèse de doctorat, Universté de provence, July 2002.

[Gir70] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *J.E. Fenstad (Ed.), Second Scandinavian Logic Symposium.* North-Holland, 1970.

[Gir72] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur.* PhD thesis, Paris VII, 1972.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1989.

[HH82] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25(2) :239–266, October 1982. Preliminary version in Proceedings 21st Symposium on Foundations of Computer Science, IEEE, 1980.

[Hue86] Gérard Huet. Induction principles formalized in the calculus of constructions. In K. Fuchi and M. Nivat, editors, *Programming of Future Generation Computers : Proceedings of the first Franco-Japanese Symposium on Programming of Future Generation Computers*, pages 205–216, Tokyo, Japan, October 1986. Elsevier Science Publishers B. V. (North-Holland) 1988.

[JK86] J.-P. Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15(4) :1155–1194, 1986. Preliminary version in Proceedings 11th ACM Symposium on Principles of Programming Languages, Salt Lake City (USA), 1984.

[JK89] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82 :1–33, 1989.

[Kes96] Delia Kesner. Confluence properties of extensional and non-extensional λ-calculi with explicit substitutions. In Harald Ganzinger, editor, *Proceedings of the 7^{th} International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 184–199, New Brunswick (New Jersey, USA), July 1996. Springer-Verlag.

[KR95] Fairouz Karmareddine and Alejandro Ríos. A λ-calculus à la de bruijn with explicit substitutions. In Manuel V. Hermenegildo and S. Doaitse Swierstra, editors, *Proceedings of the 7^{th} International Symposium on Programming Language Implementation and Logic Programming*, volume 982 of *Lecture Notes in Computer Science*, pages 45–62, Utrecht (Netherlands), September 1995. Springer-Verlag.

[KR96] Fairouz Karmareddine and Alejandro Ríos. Generalized beta-reduction and explicit substitutions. In Herbert Kuchen and S. Doaitse Swierstra, editors, *Proceedings of the 8^{th} International Symposium on Programming Language Implementation and Logic Programming*, volume 1140 of *Lecture Notes in Computer Science*, pages 378–392, Aachen (Germany), September 1996. Springer-Verlag.

[Kri93] J.-L. Krivine. *Lambda calculus, types and models.* Ellis Horwood, 1993.

[Les94] Pierre Lescanne. From λ_σ to λ_υ a journey through calculi of explicit substitutions. In *Proceedings of the 21^{st} Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 60–69, Portland (Oregon, USA), January 1994. ACM Press, New York.

[Mel95] P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate. In M. Dezani, editor, *Int. Conf. on Typed Lambda Calculus and Applications*, volume 902 of *Lecture Notes in Computer Science*, pages 328–334. Springer-Verlag, 1995.

[Mid89] A. Middledorp. A sufficient condition for the termination of the direct sum of term rewrite systems. In *Proceedings of the 4^{th} annual IEEE Symposium on Logic in Computer Science (LICS89)*, pages 396–401. IEEE Computer Society Press, 1989.

[Mus80] D. R. Musser. On proving inductive properties of abstract data types. In *Proceedings 7th ACM Symp. on Principles of Programming Languages*, pages 154–162. ACM, 1980.

[Ngu02] Quang-Huy Nguyen. *Calcul de réécriture et automatisation du raisonnement dans les assistants de preuve.* Thèse de doctorat, Université Nancy 1, October 2002.

[Ohl93] E. Ohlebusch. A simple proof of sufficient conditions for the termination of the disjoint union of term rewriting systems. *Bulletin of the European Association for Theoretical Computer Science*, 50 :223–228, 1993.

[Poi02] Henri Poincaré. *La science et l'hypothèse.* Flammarion (1968), 1902.

[PS81] G. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28 :233–264, 1981.

[Red90] U. S. Reddy. Term rewriting induction. In M. E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction, Kaiserslautern (Germany)*, volume 449 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 1990.

[Rus87] M. Rusinowitch. On termination of the direct sum of term-rewriting systems. *Information Processing Letters*, 26(1) :65–70, October 1987.

[Toy87] Y. Toyama. On the church-rosser property for the direct sum of term rewriting systens. *Journal of the ACM*, 34(1) :128–143, January 1987.

[Urb01] Xavier Urbain. *Approche incrémentale des preuves automatiques de terminaison.* Thèse de doctorat, Université Paris-Sud, Orsay, France, October 2001.

[Vir95] Patrick Viry. Rewriting modulo a rewrite system. Technical Report TR-20/95, University of Pise, December 1995.

[Vir98] Patrick Viry. Adventures in sequent calculus modulo equations. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the 2^{nd} International Workshop on Rewriting Logic and its Applications*, volume 15 of *Electronic Notes in Theoretical Computer Science*, pages 367–378, Pont-à-Mousson (France), September 1998. Elsevier Science.

[Vir02] Patrick Viry. Equational rules for rewriting logic. *Theoretical Computer Science*, 285(2) :487–517, August 2002.

[Wec92] Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, 1992.

[Wir00] Claus-Peter Wirth. Descente Infinie + Deduction. Technical Report 737/2000, FB Informatik, Universität Dortmund, August 2000.

Abstract

Proof by induction is a key proof method, as much because of the many proofs in mathematical domain which use it as because the properties we want to prove on a programm or its specification are often to be proved by induction. It is also the case for properties of protocols, as the absence of possibility for intruding, whose proofs also need induction, on the length of the messages for instance.

The methods for proof by induction, and thus the systems which implement them, are very different. The noetherian induction method is very general and its theoretical roots are clear. But its automation is difficult, which lead to the conception of proof assistants which are deeply or fully guided by the user. The induction by rewriting method on the contrary is by nature very automated.

The link between noetherian induction and induction by rewriting is generally established at a semantical level. This thesis establishes this link at the proof level using the deduction modulo formalism. Proof by induction by rewriting is then seen as the result of the internalization of induction hypotheses in deduction modulo.

To be able to establish this link, the formalism of deduction modulo is extended to congruences represented by conditional rules and equations. The evaluation of these conditions lead us to also take into account the context in which the congruence is applied. Finally, since the induction ordering cannot be made compatible with the congruence, we had to define it as protective, i.e. blocking the application of the congruence.

The results obtained about the internalisation of the induction hypotheses enable to explain some of the behavior of the induction by rewriting method : the use of one goal as an induction hypothesis to prove another goal and the possibility not to explicitly check the induction condition.

The link established in this thesis, because it lies at the proof level, will also enable systems to cooperate in a skeptical mode where the answer is not simply "yes" or "no" but a proofterm with which the correctness of the proof can be verified thanks to the Curry-Howard isomorphism.

Keywords: Automated deduction, Proof by induction, Noetherian induction, Induction by rewriting, Deduction modulo, Rewriting

Résumé

La preuve par récurrence est une méthode de preuve fondamentale, tant pour les nombreuses preuves du domaine mathématique qui l'utilisent que parce que les propriétés que l'on souhaite prouver au sujet d'un programme ou de sa spécification sont souvent prouvables par récurrence. C'est également le cas pour les propriétés sur les protocoles, comme l'absence de possibilité d'intrusion, dont la preuve nécessite également la récurrence, sur la longueur des messages notamment.

Les méthodes de preuve par récurrence, et donc plus encore les systèmes qui les implémentent, sont très diverses. La méthode de preuve par récurrence noethérienne est générale et ses fondements sont clairs. Cependant son automatisation est difficile, ce qui a conduit à la conception d'assistants de preuve qui sont fortement voire complètement guidés par l'utilisateur. La méthode de preuve par récurrence par réécriture est au contraire par nature très automatique.

Le lien entre récurrence noethérienne et récurrence par réécriture est classiquement établi de façon sémantique. Cette thèse établit le lien au niveau des preuves à l'aide du formalisme de la déduction modulo. La preuve par récurrence par réécriture est ainsi vue comme le résultat de l'internalisation en déduction modulo des hypothèses de récurrence.

Pour parvenir à établir ce lien, le formalisme de la déduction modulo est étendu au traitement de congruences exprimées à l'aide de règles et d'équations conditionnelles. L'évaluation de ces conditions nous a également amené à prendre en compte le contexte dans lequel est appliquée la congruence. Enfin, l'ordre de récurrence ne pouvant pas être compatible avec la congruence, il a fallu le définir comme protecteur, c'est-à-dire bloquant l'application de la congruence.

Les résultats obtenus sur l'internalisation des hypothèses de récurrence permettent d'expliquer certains comportements de la méthode de récurrence par réécriture : l'utilisation d'un but comme hypothèse de récurrence pour la preuve d'un autre but et la possibilité de ne pas vérifier explicitement la condition de récurrence.

Le lien établi dans cette thèse, parce qu'il se situe au niveau des preuves, permettra aussi la coopération des systèmes dans un mode sceptique où la réponse n'est pas de type "oui" ou "non" mais un terme de preuve à partir duquel la correction de la preuve peut être vérifiée grâce à l'isomorphisme de Curry-Howard.

Mots-clés: Déduction automatique, Preuve par récurrence, Récurrence noethérienne, Récurrence par réécriture, Déduction modulo, Réécriture.

Monsieur **DEPLAGNE Eric**

DOCTORAT de l'UNIVERSITE HENRI POINCARE, NANCY 1

en INFORMATIQUE

VU, APPROUVÉ ET PERMIS D'IMPRIMER

n°750

Nancy, le 5 décembre 2002

Le Président de l'Université

Cl. BURGET

UNIVERSITE HENRI POINCARÉ, NANCY 1 * LE PRESIDENT

Université Henri Poincaré, Nancy I
24-30 rue Lionnois - B.P. 3069 - 54013 NANCY Cédex
Tél. : 03 83 68 20 00 - Fax : 03 83 68 21 00