Rapport INRIA 1995 — Programme 2

Logique, Recherche de Preuves et Programmation

Action TYPES

5 janvier 1996

Action TYPES

Logique, Recherche de Preuves et Programmation

Localisation : *Nancy*[1]

Mots-clés :

1 Composition de l'équipe

Responsable scientifique

Didier Galmiche, Maître de conférences Habilité,
Université Henri Poincaré - Nancy 1

Secrétaire

Nathalie Pierre, Technicien de recherche Inria

Personnel Inria

Philippe de Groote, Chargé de recherche
Christian Retoré, Chargé de recherche

Personnel Universités et URA 262

Jean Yves Marion, Maître de conférences, Université
Nancy 2, depuis le 1/10/95
Guy Perrier, Maître de conférences, Université Nancy 2,

[1]Projet commun à l'Inria Lorraine et au Centre de Recherche en Informatique de Nancy (CRIN, laboratoire des universités de Nancy 1, Nancy 2 et de l'INPL, associé au CNRS URA 262)

IUT Epinal, depuis le 1/10/95
Odile Hermann, Professeur Certifié, IUFM Nancy-Metz

Chercheurs doctorant

Denis Bechet, Boursier AMN, jusqu'au 30/09/95, puis
ATER Université de Metz, depuis le 1/10/95
Eric Boudinet, Boursier MESR, jusqu'au 30/09/95
Dominique Larchey-Wendling, Elève ENS Lyon,
depuis le 1/10/95
Guy Perrier, Prof. agrégé du secondaire, jusqu'au 30/09/95
Catherine Pilière, Boursier MESR, jusqu'au 30/09/95

Chercheurs invités

François Lamarche, chercheur invité, depuis le 1/10/95
Jim Lipton, assistant professor, Wesleyan University (Etats Unis), du 1/06/95 au 30/06/95
John Mullins, assistant professor, Ottawa University (Canada), du 1/06/95 au 30/06/95

Stagiaires

Guillaume Bonfante, Ecole des Mines - DEA Informatique
Laurent Parise, MIM, ULP Strasbourg
Sylvain Pogodalla, Ecole des Mines - DEA Informatique

Collaborateur extérieur

Alain Lecomte, Maître de conférences Habilité,
Université Grenoble 2

2 Présentation du projet

Le projet scientifique de l'action TYPES s'inscrit dans le cadre général de la conception de systèmes informatiques en considérant que la modélisation des systèmes et des calculs doit se faire au niveau logique. Pour prendre en compte les différents points de la conception et de l'analyse d'un système au travers de la logique, on considérera comme fondamentale la relation entre *informatique* et *logique* (plus précisement la *théorie de la démonstration*) telle qu'elle apparaît au travers de deux paradigmes actuels de la programmation : *proofs-as-programs* (reposant

sur l'isomorphisme de Curry-Howard) et *proof-search as computation*. Notons que ces deux points de vue partagent à la fois des techniques concernant l'analyse structurelle des preuves (ou démonstrations) et la recherche automatisée de preuves. On va considérer ici des logiques ou des systèmes de preuves appropriés. Ils sont issus de la théorie des types et des logiques classique, intuitionniste, linéaire, dans lesquelles les preuves ou les réseaux de preuve (preuves sous forme de graphes particuliers) peuvent être utilisés comme modèles de calcul.
Cet aspect calculatoire (ou algorithmique) des preuves ne peut être appréhendé, analysé et mis en œuvre sans la compréhension et la maîtrise de la complexité des problèmes abordés et sans disposer de méthodes performantes de construction de preuves.
La problématique de notre travail sera définie selon des objectifs que nous pouvons résumer par

- l'étude de la prise en compte au niveau logique (formalismes logiques, théorie de la démonstration) de concepts de la programmation fondamentaux pour le développement de systèmes informatiques (non-déterminisme, contrôle, concurrence, complexité, séquentialité).

- l'étude et la conception de méthodes et d'outils de recherche et d'analyse de preuves. Nous souhaitons proposer des méthodes originales de recherche de preuves prenant en compte par exemple le contenu algorithmique des preuves ainsi que l'intérêt des fragments logiques considérés (issus des logiques linéaire, classique, etc..) pour la modélisation et la spécification de problèmes.

- l'application de ces méthodes et leur mise en œuvre (et implantation) dans des domaines précis qui peuvent nécessiter l'adaptation et la spécialisation des méthodes au domaine d'étude.

Ainsi, nos recherches se font selon une démarche qui repose sur l'analyse des aspects calculatoires de la théorie de la démonstration (principalement dans des logiques constructives), la proposition de méthodes de construction de preuves et l'interaction entre analyse et mise en œuvre des techniques et outils. Un souci constant est l'étude fine de l'automatisation et de la complexité de techniques proposées.

3 Actions de recherche

Nos actions de recherche en 1995 sont présentées ici suivant une partition entre deux axes complémentaires traitant de *la logique et la construction de preuves* et de *la logique et la modélisation de concepts de programmation*. L'interaction entre ces deux axes est une caractéristique de notre approche.

3.1 Logique et construction de preuves

Dans ce cadre, nos travaux ont porté sur la proposition de nouvelles méthodes de recherche de preuves, principalement dans des fragments de la logique linéaire, ainsi que sur l'implantation de ces méthodes en Quintus-Prolog et dans le système Isabelle. Ils ont porté également sur l'étude des réseaux de démonstration et de leurs critères de correction ainsi que sur les réseaux d'interaction. De plus, nous avons poursuivi l'étude du contenu algorithmique des preuves en logique intuitionniste du second ordre.

3.1.1 Construction de preuves et de réseaux en logique linéaire

Participants: Guillaume Bonfante, Eric Boudinet, Didier Galmiche, Philippe de Groote, Jean-Yves Marion, Sylvain Pogodalla

Dans un premier temps, nous avons appliqué notre méthode générale de conception de preuve canonique à la logique linéaire intuitionniste (ILL) et à la logique linéaire intuitionniste complète (FILL). Cette méthode repose sur l'étude des permutabilités et des mouvements d'inférence dans une preuve et conduit à la définition d'un ordre d'application des règles et ainsi à la définition de preuves canoniques, qui sont complètes par rapport au fragment [5].
Ainsi, en vue de l'étude du fragment FILL nous avons défini des preuves canoniques qui sont complètes pour le fragment et proposé des outils de démonstration automatique dans ce système logique [10].
Toutefois, même si on peut constater que l'approche précédente conduit à des méthodes et à des implantations relativement efficaces, la complexité des preuves nous amène à sortir du cadre logique et à aborder l'*aspect algorithmique de la recherche de preuve*. Nous avons obtenu, dans ce cadre, des résultats qui mettent en jeu d'une part des algorithmes parallèles basés sur des contractions d'arbres, d'autre part des techniques

d'optimisation.
Ainsi, nous avons proposé un algorithme parallèle de décision pour le fragment additif de la logique linéaire (ALL), montrant de plus que ce fragment est NC^1-complet. Cet algorithme est tel que le temps de calcul est logarithmique dans la taille de la formule à démontrer et utilise un nombre polynomial de processeurs[2]. Plus récemment, nous avons proposé un algorithme séquentiel de décision pour ALL en temps polynomial [12]. On obtient ainsi un nouveau système de preuves qui permet de proposer une méthode des tableaux qui réalise la recherche de preuves en temps polynômial. A la base de ce résultat, on trouve la possibilité de représenter des preuves dans ce fragment sous forme de graphe acyclique.
A partir de ces idées, nous avons proposé, au travers d'un nouveau système de preuves, une nouvelle méthode de recherche de preuves pour le fragment multiplicatif et additif de la logique linéaire (MALL), qui repose sur une relation de déduction manipulant à la fois des contextes multiplicatifs et additifs [11].
De plus, nous avons étudié les réseaux de preuves et la recherche automatique de preuves dans des théories linéaires conjonctives [23]. L'algorithme de construction automatique de réseaux dans le fragment multiplicatif de la logique linéaire (MLL) que nous avions proposé précédemment[3] a été analysé et implanté (en Quintus-Prolog). De plus, pour compléter ces études, nous avons étudié la recherche de preuves en logique linéaire au travers de la logique de l'Unité[4] (LU) et de certains de ses fragments [20].

Le fragment MALL a été implanté dans le système Isabelle [8]. Cette tâche présentait plusieurs difficultés a priori: la métalogique utilisée par Isabelle étant un fragment de la logique intuitionniste d'ordre supérieur les mécanismes de contrôle permettant d'implanter une logique sous-structurelle ne sont pas fournis par le système. De plus, la construction de démonstration en Isabelle se fait par chaînage arrière. Or, si l'on en

[2] J.-Y. Marion, P. Bradford and L. Moss, *The Additive fragment is NC^1-complete*, Logic Complexity and calculability, Indianapolis, USA, 13-16 Octobre 1994

[3] D. Galmiche and G. Perrier, *A procedure for automatic proof nets construction*, Int. Conf. LPAR'92, St Petersburg, LNAI 624, pp 42-53, 1992.

[4] J.Y. Girard, *On the Unity of Logic*, Annals of Pure and Applied Logic, 59:201-207, 1991

croit la littérature,[5] la recherche de preuves par chaînage arrière s'adapte mal au cas de la logique linéaire.
La solution au premier de ces problèmes a consisté à implanter la logique linéaire sous forme d'un calcul de séquents. Quant au deuxième problème, il a été résolu par le développement de critères permettant d'élaguer l'arbre de recherche de manière efficace. Parmi ces critères, il en est un qui s'est avéré particulièrement intéressant et qui concerne ce que l'on pourrait appeler, par analogie avec le théorème de séquentialisation de Girard,[6] la recherche heuristique d'un tenseur scindant. Un des problèmes, en effet, que l'on rencontre lorsque l'on désire démontrer un séquent linéaire S formé de plusieurs formules de la forme $A \otimes B$ est de déterminer laquelle de ces formules doit être attaquée en premier. Or, dans le cas où S est démontrable, la ou les formules à attaquer en premier correspondent précisément aux tenseurs scindants des réseaux de démonstration de S. Bien sûr, dans la mesure où S n'a pas encore été démontré, on n'en connait aucun réseau de démonstration. Aussi, l'heuristique construit-elle une approximation des éventuels réseaux de S telle que tout tenseur scindant pour l'approximation est scindant pour tout réseau de S.

3.1.2 Développement du système APRIL

Participants: Eric Boudinet, Laurent Parise

Concernant l'implantation (en Quintus-Prolog) des démonstrateurs dans différents fragments de LL (s'appuyant sur la notion de preuves canoniques spécialisées pour chaque fragment [5]), Eric Boudinet a continué à développer le système APRIL qui consiste en différents modules de recherche de preuves (automatique ou interactive) dans divers fragments de la logique linéaire (CLL, ILL et FILL), avec une interface conviviale et des possibilités de manipulation des preuves obtenues (stockage, affichage, réutilisation etc...).
Un module qui permet de produire sous forme d'arbre la preuve développée lors du processus de recherche a été implanté en vue d'améliorer

[5] T. Tammet. Proof search strategies in linear logic, Technical Report 70, Department of Computer Science, Chalmers University of Technology, Göteborg, Sweden, 1993.

[6] J.-Y. Girard. Quantifiers in linear logic II. Technical Report 19, Equipe de Logique Mathématique, Université de Paris VII, 1991.

la manipulation des preuves [21].
Des réalisations récentes sont à inclure sous forme de modules dans APRIL, comme un algorithme de construction automatisée de réseaux de preuves [23] ainsi qu'un algorithme de recherche de preuves dans un fragment de LU [20].

3.1.3 Réseaux de preuves, séquentialité et linguistique

Participants: Christian Retoré, Alain Lecomte

Une des applications informatiques de la théorie de la démonstration qui nous intéresse particulièrement est la linguistique informatique, et plus précisément la syntaxe des grammaires catégorielles. Le noyau des grammaires catégorielles est le calcul de Lambek qui n'est autre qu'un fragment de la logique linéaire non-commutative. L'application du calcul ordonné [18] à ce domaine ouvre de nouvelles perspectives en décrivant à l'intérieur de la logique des constructions linguistiques courantes (discontinuités, ordre relativement libre, clitiques,...) jusque là absentes des grammaires catégorielles. Notre but est donc d'étudier, en vue d'applications à la linguistique, les réseaux de preuves (ou de démonstration) en calcul ordonné et la séquentialité.
Concernant le travail sur les réseaux multiplicatifs, en utilisant une présentation duale d'un travail précédent[7], nous avons obtenu un nouveau critère pour le fragment multiplicatif, formulé en termes de couplages parfaits. Cela permet d'unifier diverses techniques de séquentialisation, et s'avère très utile à l'étude du calcul ordonné [7]. Concernant le calcul ordonné, nous avons obtenu un résultat partiel sur la séquentialisation du calcul ordonné[8]. La conjecture plus forte sur les graphes orientés, munis d'un couplage parfait, sans circuit alternant et saturés est vraie si toutefois les arcs ne faisant pas partie du couplage définissent une relation faiblement transitive. Cette conjecture plus forte en toute généralité a été réfutée par D. Bechet: les programmes qu'il a écrits ont engendré un contre exemple. Lors de l'étude du calcul d'Abrusci, un critère pour ses réseaux, plus simple que celui qu'il propose, a été obtenu. Il est formulé en termes de bon parenthésage et permet d'envisager l'étude de la complexité de ce calcul [24].

[7] C. Retoré, Graph theory from linear logic: aggregates, Prépublication 47, Equipe de Logique, Université Paris 7, October 1993

[8] C. Retoré, *Réseaux et Séquents Ordonnés*, Thèse de Doctorat, spécialité Mathématiques, Université Paris 7, février 1993

Concernant les applications linguistiques de cette étude des réseaux, le modèle linguistique basé sur le calcul ordonné développé avec A. Lecomte a été amélioré et a donné lieu à une publication [13].
De plus, nous avons proposé une caractérisation sémantique de la correction d'un réseau [17]. On sait que les réseaux (avec la règle mix) admettent une sémantique calculée comme un ensemble d'expériences. Ici, nous prouvons l'inverse c'est à dire: une structure de preuve est un réseau de preuve si son ensemble d'expériences est un objet sémantique (clique de l'espace cohérent correspondant).

3.1.4 Les réseaux d'interaction

Participants: Denis Bechet

L'étude de traductions du λ-calcul dans les systèmes de réduction de graphes (les réseaux de preuves, les réseaux d'interaction...) nous ont amenés à proposer des transformations de preuves en logique linéaire simplifiant les constructions de base. Ainsi, les additifs et les portes auxiliaires des opérateurs exponentiels disparaissent au cours de ces transformations. Ceci débouche sur une implantation presque direct du processus de normalisation de preuve dans les réseaux d'interaction. Les transformations introduites démontrent la puissance d'expression des quantificateurs du second ordre. Nous pensons généraliser leur utilisation dans les réseaux en relation avec les notions de modules et d'interfaces.
Yves Lafont ayant proposé un système universel de réseaux d'interaction (comme la notion de machine de Turing universelle) composé de trois cellules, nous avons réussi, en collaboration avec lui, à le réduire à deux cellules. Il reste à simplifier les résultats obtenus et à démontrer certaines conjectures dans ce domaine.
La notion d'interface dans les réseaux de preuves ou d'interaction semble importante car elle permet de réduire les possibilités de connexion de bouts de réseaux. Nous avons trouvé un système fini de générateurs d'interface dans les réseaux d'interaction. Cette notion peut être étendue à la logique linéaire et au calcul ordonné. Nous espérons que cette notion permettra de trouver une réponse au problème de la séquentialisation du calcul ordonné.

3.1.5 Preuves et contenu algorithmique

Participants : Didier Galmiche, Odile Hermann

Dans ce cadre, l'étude de la mécanisation du développement de preuves et de programmes dans le système AF_2 a été poursuivie avec un effort particulier de formalisation des travaux antérieurs [3]. Ainsi, la mise en œuvre de la recherche de preuves inductives dans ce cadre ainsi que sa mécanisation a nécessité la proposition d'un calcul des séquents, équivalent au système initial, mieux adapté à la recherche efficace de preuves. Elle nous a conduit à définir des tactiques et des stratégies spécialisées conduisant à des plans de preuves. Ces résultats sont issus d'une analyse fine de la structure des preuves ainsi que de leur sémantique.
Le deuxième point qui est la mise en œuvre de l'induction (en fait déterminer le type d'induction et sur quelles variables la réaliser) a conduit à certaines propositions. Tout d'abord nous avons restreint la classe de spécifications considérées en vue de résoudre le problème du choix de variable et de la position d'induction. Nous avons étudié une taxonomie des fonctions traitées et des relations entre ces classes de fonctions et l'échec de certaines formes d'induction. De plus, nous proposons une procédure de choix d'induction qui prend en charge à la fois la recherche des variables et des positions d'induction mais également le choix heuristique de schémas d'induction adaptés, évitant les échecs et considérant la forme du programme synthétisé lors de la recherche de preuves.
Ces résultats ont été implantés sous forme de plans de preuve dans le système SKIL[9] (développé en Quintus-Prolog).

3.2 Logique et concepts de programmation

Dans ce cadre, complémentaire des études précédentes, nous avons étudié plus précisement la modélisation et la prise en compte au niveau logique de concepts de programmation comme les opérateurs de contrôle (en logique classique), les concepts de concurrence, non-déterminisme et programmation logique (en logique linéaire), l'évaluation partielle et l'étude de classes de complexité.

[9] D. Galmiche and O. Hermann, *SKIL: a system for programming with proofs*, LPAR'93, International Conference on Logic Programming and Automated Reasoning, LNAI 698, pp 348-350. 1993

3.2.1 Logique et opérateurs de contrôle

Participants : Philippe de Groote, Catherine Pilière

Dans le cadre de l'isomorphisme de Curry-Howard, il existe des rapports étroits entre logique classique et opérateurs de contrôle séquentiel. Ceci a motivé une étude des mécanismes de traitement d'exceptions en ML standard et a donné lieu à la définition d'un λ-calcul de traitement d'exceptions [9]. Ce calcul ($\boldsymbol{\lambda}^{\rightarrow}_{exn}$) est muni d'une règle de typage correspondant à l'élimination du tiers exclus et sa sémantique opérationnelle, qui est conservative par rapport à une sémantique opérationnelle à la ML, assure que toute exception levée sera forcément traitée.

3.2.2 Aspects concurrents de la logique linéaire

Participants : Eric Boudinet, Didier Galmiche, Guy Perrier

Il existe différentes approches pour modéliser la programmation concurrente dans des fragments de la logique linéaire. Nous avons considéré différents points de vue (ou paradigmes): *proofs as computations* avec des relations entre des extensions non-déterministes de réseaux de Petri et (la prouvabilité et) la recherche de preuves dans des fragments de la logique intuitionniste linéaire (ILL); *proof-search as computations* en considérant la programmation logique concurrente au travers de calculs de processus.

Un premier travail a consisté à exploiter les résultats obtenus dans le domaine de la démonstration en logique linéaire pour proposer un modèle du calcul parallèle qui repose sur le paradigme *proof search as computation* [4]. Une analyse reposant sur la permutabilité d'inférences dans le calcul des séquents a permis de délimiter un fragment de la logique linéaire intuitionniste suffisamment restreint pour que la construction de démonstrations puisse y constituer un calcul réaliste mais en même temps suffisamment étendu pour pouvoir exprimer certains concepts fondamentaux du parallélisme [15, 16, 22].

Le modèle obtenu, dénommé CPL, a les caractéristiques d'un calcul de processus où la communication par messages s'effectue sous une forme asynchrone. L'intérêt d'un tel modèle est qu'il permet de représenter dans un même cadre logique les programmes parallèles et certaines de leurs propriétés. En particulier, il permet d'exprimer à un niveau logique la notion d'interface d'un système informatique et d'utiliser le potentiel de la théorie de la démonstration pour définir un calcul sur les interfaces. Le travail a également porté sur la confrontation avec d'autres modèles

du parallélisme (une traduction du π-calcul dans CPL est en cours) et sur la représentation de ce modèle à l'aide des réseaux de démonstration de la logique linéaire.
Un autre travail a consisté à considérer l'approche reposant sur les paradigmes *formulas-as-states* et *proofs-as-computations* où les liens entre des extensions de réseaux de Petri (qui permettent de prendre en compte différents non-déterminismes) et des fragments de LL ont été analysés. Nous avons étudié dans ce contexte un cadre logique particulier, FILL (Full Intuitionistic Linear Logic), qui, à la différence de ILL, inclut une multiplication disjonctive et est représenté avec un calcul des séquents à conclusions multiples. Il apparaît naturel d'analyser FILL comme modèle de la programmation concurrente. Ainsi nous avons défini de nouveaux réseaux non-deterministes en vue d'appréhender des cas de non-déterminismes comme celui de la position de certaines resources, Le problème d'accessibilité dans un tel réseau a été montré équivalent au problème de la prouvabilité dans le fragment FILL, ce qui nous permet d'utiliser, pour l'analyse des calculs, nos résultats sur la recherche de preuves dans ce fragment [10]. De plus, nous avons analysé l'intérêt du fragment FILL comme un cadre de mise en œuvre de la programmation concurrente suivant le paradigme *proof-search as computations*, ce en essayant d'étendre ce qui a été fait dans le système CPL. Cela conduit à un calcul concurrent, issu de FILL, où on peut traiter des non-déterminismes au niveau de l'expression des messages [10]. Une poursuite de ce travail consistera à étudier l'expression de mécanismes pour la programmation objet concurrente dans de tels cadres et à comparer avec d'autres approches (logiques ou non) de ce type de programmation.

3.2.3 Evaluation partielle

Participants : Denis Bechet

Nous avons proposé de nouveaux mécanismes d'identification des objets spécialisés. Cela découle d'une partition de la spécialisation de programme en deux phases. La première phase, dite de *clonage*, cherche de nouvelles définitions récursives du programme original en dupliquant les expressions et en tenant compte de la donnée fixée. La seconde phase optimise le programme cloné. Cette approche permet de dissocier l'utilisation de transformations optimisantes de la recherche de nouvelles définitions récursives. Ainsi, des méthodes originales pour ce deuxième problème peuvent être introduites. Nous proposons notamment une méthode d'identification basée sur la notion d'*événement* résumant les actions nécessaires pour créer un objet spécialisé [2, 19].

3.2.4 Classes de complexité

Participants : Jean-Yves Marion

Les caractérisations des classes de complexité, indépendantes du modèle de calcul, peuvent être réparties, tant bien que mal, suivant trois thèmes: la théorie des modèles finis, la théorie de la démonstration, et enfin *les schémas de fonctions*, thème dans lequel nous nous situons. En effet, à la différence des résultats antérieurs, les caractérisations des classes de complexité obtenues sont sans construction ad-hoc et reposent sur une notion de stratification des données d'un calcul.
L'espace polynomial, *Poly-Space*, contient, entre autres, les problèmes d'optimisation stochastique. Cette classe est aussi un point de jonction entre les algorithmes séquentiels utilisant un espace polynomial et les algorithmes parallèles dont le temps de calcul est polynomial. En collaboration avec D. Leivant, nous avons montré que les programmes calculables en espace polynomial sont représentés par une classe de fonctions engendrées par un schéma de récurrence avec substitution de paramètres dont les domaines sont stratifiés [14].

4 Actions nationales et internationales

4.1 Actions nationales

Des membres du projet ont participé aux actions nationales suivantes :

- Modèles Logiques de la Programmation, projet inter PRC du CNRS dirigé par M. Parigot, dont les principaux sites sont Paris 7 (J.L. Krivine, M. Parigot), Orsay (H. Comon), Lille (M. Dauchet), Marseille (J.Y. Girard, Y. Lafont), Sophia-Antipolis (G. Boudol), Chambéry (R.David) et Nancy (R. Amadio, A. Cichon, H. Kirchner), jusqu'en fin 1994 (participants: D. Galmiche et Ph. de Groote).

- PRC IA dans le projet *Méthodes sémantiques de démonstration pour des logiques non-standard* en particulier pour développer et évaluer des méthodes sémantiques de compilation logique pour des logiques sous-structurelles (participant: D. Galmiche).

4.2 Actions internationales

Des membres du projet ont participé aux actions internationales suivantes :

- Réseau MEDICIS : ce réseau Capital Humain et Mobilité de la CEE, sur les méthodes de développement de spécifications formelles, regroupe des équipes universitaires d'Allemagne, de Belgique, d'Espagne, de Grande-Bretagne, du Portugal et de Pologne, (participants : D. Galmiche et Ph. de Groote).

- Projet Esprit BRA 7032 Gentzen sur le thème *Common foundations of logical and functional languages*, et donc sur les fondements et modèles logiques de la programmation, avec Marseille (J.Y. Girard), Turin (M. Dezani), Rome (C. Böhm), Swansea (J.R. Hindley), Munich (Switzenberger), Zurich (Engeler), (participant: D. Galmiche).

4.3 Participation à des comités de programmes

D. Galmiche a participé au comité de programme du *5th Int. Workshop on Extensions of Logic Programming, ELP'96*, Leipzig, mars 1996 ainsi qu'au comité de programme du *5th Int. Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, Palerme, mai 1996.
Ph. de Groote est l'éditeur du volume 8 du Cahier du Centre de Logique de l'Université Catholique de Louvain intitulé *The Curry-Howard Isomorphism*, Academia, 1995.

4.4 Invitations

Le projet a accueilli, dans le cadre de séminaires, des chercheurs, dont, en particulier, Arnaud Fleury (Equipe de Logique Paris 7), Jacqueline Vauzeilles et Christophe Fouquere (Paris 13), Gilles Zemor (ENST Paris), Walter Py (Université de Chambéry), Sachio Hirokawa (Université de Swansea), Roy Dyckhoff (Université de St Andrews), Natacha Kurtonina (Université d'Utrecht), Aarne Ranta (Université de Tampere).

5 Diffusion des résultats

5.1 Enseignement

Nous sommes intervenus en particulier dans des cours de second et troisième cycle de l'Université Henri Poincaré et dans des écoles d'ingénieurs. On peut citer, entre autres,

- Cours de Maîtrise Informatique *Compilation et langages*, (Ph. de Groote),
- TD de 2ème année de l'Ecole des Mines, *Algorithmique stochastique*, (J.-Y. Marion),
- Cours de 2ème année ESSTIN, *Concepts d'informatique théorique*, (D. Galmiche),
- Cours de DEA d'Informatique, *Modèles Logiques de Calcul*, (D. Galmiche),
- Cours de DESS Informatique, *Logique et Programmation Logique*, (D. Galmiche),
- Cours de 3ème année SUPELEC, Campus de Metz, *Logique et Programmation : concepts fondamentaux*, (D. Galmiche).

5.1.1 Séminaires

Nous avons donné un certain nombre de séminaires, en particulier,

- *Morceaux choisis de complexité algorithmique et de programmation fonctionnelle*, Université Bordeaux I, janvier 1995, (J.Y. Marion),
- *Calculabilité sur les structures finies et langages fonctionnels*, Université de Lille I, février 1995, (J.Y. Marion),
- *Introduction à la logique linéaire*, Xerox research center, Meylan, février 1995, (Ph. de Groote),
- *Proof search in linear logic: an overview*, INESC, Université de Lisbonne, mars 1995, (D. Galmiche),
- *Comment caractériser les fonctions facilement calculables*, Université de Caen, mars 1995, (J.Y. Marion),
- *Calculabilité et modèles finis*, Séminaire du Groupe Logique, Nancy, avril 1995, (J.Y. Marion),

- *Proofs and Concurrency in linear logic*, Université de Tokyo, juin 1995, (D. Galmiche),
- *Concurrent Programming in Full Intuitionistic Linear Logic*, IRIT, Université de Toulouse, novembre 1995, (D. Galmiche).

De plus, Ph. de Groote, A. Lecomte et C. Retoré ont visité l'équipe de linguistique du Rank-Xerox Research Center de Meylan (en février 1995) ainsi que le projet CROAP lors de la venue à Sophia-Antipolis de A. Ranta (en mars 1995).

5.1.2 Jurys de thèse

En 1995, D. Galmiche a été membre du jury de 6 thèses d'Université (dont 2 en tant que rapporteur).

5.2 Participation à des conférences et colloques

- Conference CMU'95, Paris, mars 1995 (J-Y. Marion),
- Workshop BRA Gentzen, Swansea, mars 1995 (D. Galmiche),
- Conference TLCA'95, Edinburgh, avril 1995 (Ph. de Groote),
- Workshop on Logic Domains and Programming Languages, Darmstadt, Mai 1995 (C. Retoré),
- 4th Workshop on Theorem Proving with Analytic Tableaux and related Methods, St Goar am Rhein, May 1995 (D. Galmiche, Ph. de Groote),
- Workshop HCM Medicis, Varsovie, Juin 1995 (D. Galmiche, Ph. de Groote),
- Conference RenPar'7, Mons, Juin 1995 (G. Perrier),
- 2nd Workshop on Non-Standard Logics and Logical Aspects of Computer Science, Irkutsk, Juin 1995 (J-Y. Marion),
- Workshop on Object-Based Parallel and Distributed Computation, OBPDC'95, Tokyo, Juin 1995 (D. Galmiche),
- Summer School ESSLI and Workshop on Formal Grammar, Barcelone, Août 1995 (A. Lecomte, C. Retoré),
- Conference on Computer Science Logic, CSL'95, Paderborn, Septembre 1995 (G. Perrier, C. Retoré).

5.3 Organisation de colloques et de cours

A. Lecomte et C. Retoré ont organisé à Nancy, les 18 et 19 septembre 1995, une rencontre internationale dont le thème était *Logical Aspects of Computational Linguistics*.

6 Publications

Livres et monographies

[1] P. de Groote (réd.), *The Curry-Howard Isomorphism*, volume 8 of Cahier du centre de Logique, Université Catholique de Louvain, Academia, 1995.

Thèses

[2] D. Bechet, *Les valeurs alternatives et la notion d'événement dans l'évaluation partielle*, thèse de doctorat, Université Paris 7 – Denis Diderot, Paris, octobre 1995.

[3] O. Hermann, *Mécanisation de la recherche de preuves et de programmes en Arithmétique Fonctionnelledu Second Ordre*, thèse de doctorat, Université Henri Poincaré, Nancy, Mai 1995.

[4] G. Perrier, *De la construction de preuves à la programmation parallèle en logique linéaire*, thèse de doctorat, Université Henri Poincaré, Nancy, Janvier 1995.

Articles et chapitres de livre

[5] D. Galmiche, G. Perrier, « On Proof Normalization in Linear Logic », *Theoretical Computer Science 135*, 1, December 1994, p. 67–110.

[6] D. Galmiche, « Proof Theory and Programming », *ERCIM News 23*, October 1995, p. 16–17, Special number on Software Quality.

[7] C. Retoré, « Des réseaux de démonstration pour la linguistique: une introduction à la logique linéaire », *Traitement Automatique des Langues*, 1996, to appear.

Communications à des congrès, colloques, etc.

[8] P. de Groote, « Linear logic with Isabelle: pruning the proof search tree », *in: 4th Workshop on Theorem Proving with Analytic Tableaux and*

Related Methods, LNCS 918, Springer Verlag, p. 263–277, St Goar am Rhein, Germany, 1995.

[9] P. de Groote, « A simple calculus of exception handling », *in : Second International Conference on Typed Lambda Calculi and Applications, TLCA'95, LNCS 902*, Springer Verlag, p. 201–215, Edimburgh, 1995.

[10] D. Galmiche, E. Boudinet, « Nets, Proofs and Concurrent Object Programming in Full Intuitionistic Linear Logic », *in : Object-based Parallel and Distributed Computation, OBPDC'95*, Tokyo, Japan, June 1995.

[11] D. Galmiche, J.-Y. Marion, « Dealing with additives in MALL proof search », *in : 2nd Workshop on Non-standard Logics and Logical Aspects of Computer Science*, Irkutsk, Russia, June 1995.

[12] D. Galmiche, J.-Y. Marion, « Semantic Proof Search Methods for ALL - a first approach - », *in : 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, St Goar am Rhein, Germany, May 1995.

[13] A. Lecomte, C. Retoré, « Pomset logic as an alternative categorial grammar », *in : DYANA workshop on Proof theory, linear logic and categorial grammar*, M. M. M. Abrusci, C. Casadio (réd.), 1995.

[14] D. Leivant, J.-Y. Marion, « Ramified recurrence and computational complexity II: substitution and poly-space, LNCS 933 », *in : Proceedings of CSL 94*, L. Pacholski, J. Tiuryn (réd.), Springer Verlag, p. pp 486–500, 1995.

[15] G. Perrier, « CPL : un modèle du calcul parallèle », *in : Septièmes Rencontres Francophones du Parallélisme, RenPar'7*, J.-L. Dekeyser, G. Libert, P. Manneback (réd.), PIP-FPMs, Mons, Belgique, juin 1995.

[16] G. Perrier, « A model of concurrency based on Linear Logic », *in : Conference on Computer Science Logic, CSL'95*, Paderborn, Germany, 1995.

[17] C. Retoré, « On the relation between coherence semantics and multiplicative proof nets », *in : Workshop on Logic Domains and Programming Languages*, Darmstadt, May 1995.

[18] C. Retoré, « Pomset logic: a non-commutative extension of classical linear logic », *in : Conference on Computer Science Logic, CSL'95*, Paderborn, Germany, 1995.

Rapports de recherche et publications internes

[19] D. BECHET, «Removing Value Encoding using Alternative Values in Partial Evaluation of Strongly-Typed Languages», *technical report n96-R-005*, CRIN-CNRS, 1996, Accepted at ESOP'96.

[20] G. BONFANTE, *Les fragments de la Logique de l'Unité : preuves et applications*, Mémoire, DEA Informatique, Université Henri Poincaré, Nancy I, 1995.

[21] L. PARISE, *Etude de la représentation des preuves dans le système APRIL*, Mémoire, Magistère Mathématiques, ULP Strasbourg, Septembre 1995.

[22] G. PERRIER, «Concurrent Programming in Linear Logic», *technical report n95-R-052*, CRIN-CNRS, Nancy, March 1995.

[23] S. POGODALLA, *Réseaux de preuves, actions et concurrence*, Mémoire, DEA Informatique, Université Henri Poincaré, Nancy I, 1995.

Divers

[24] C. RETORÉ, «A simpler criterion for Abrusci proof nets», Draft, August 1995.

7 Abstract

The scientific project of TYPES is involved in the general approach to the design of correct systems through the links between logic and programming. We consider the relationship between programs and proofs as it appears in two programming paradigms: *proofs-as-programs* (Curry-Howard isomorphism) and *proof-search-as-computation* as fundamental. Thus, we modelize concepts and calculi through logic (proof theory, semantics) with methods based on the analysis of proofs and on the automatized proof construction (search) in constructive logics.

The main research topics are the following:

- Analysis and representation of programming concepts into logic (mainly proof theory), treatment of non-determinism and control in sequential and concurrent systems.

- Methods for the analysis and the construction of proofs in adequate frameworks for concepts design and systems specification (type theory,

linear logic and typed λ-calculi).

- Proof environments based on these methods and tools (for verification, synthesis) and applications.

Table des matières