Rapport INRIA 1994 — Programme 2

Contraintes, Déduction Automatique et Preuve de Propriétés de Logiciels.

PROJET PROTHEO

3 mai 1995

PROJET PROTHEO

Contraintes, Déduction Automatique et Preuve de Propriétés de Logiciels.

Localisation : *Nancy*

Mots-clés : concurrence (1), contrainte (1), déduction automatique (1), démonstration automatique (1), réécriture (1), spécification formelle (1).

1 Composition de l'équipe

Responsable scientifique

Claude Kirchner, directeur de recherche, INRIA

Responsable permanent

Hélène Kirchner, chargée de recherche, CNRS

Secrétariat

Christiane Guyot, INRIA

Personnel INRIA

Isabelle Gnaedig-Antoine, chargée de recherche
Michaël Rusinowitch, directeur de recherche (à compter du 1/10/94)
Adel Bouhoula, chargé de recherche (à compter du 1/10/94)

Personnel CNRS

Eric Domenjoud, chargé de recherche

Personnel Université

Denis Lugiez, maître de conférence (Université H. Poincaré, en détachement au CNRS à Grenoble à compter du 1/10/94)
Christophe Ringeissen, ATER (Université H. Poincaré)

Chercheurs doctorants

Farid Ajili, boursier du gouvernement tunisien et INRIA (à compter du 1/10/94)
Iliès Alouini, boursier du gouvernement tunisien et INRIA
Ali Amaniss, allocataire MESR
Narjès Berregeb, boursière CIES et INRIA
Régis Curien, ATER (ESSTIN, à compter du 1/10/94)
Thomas Genet, allocataire MESR (à compter du 1/10/94)
Claus Hintermeier, allocataire MESR et boursier INRIA
Maria Huber, allocataire MESR (jusqu'au 30/10/94)
Eric Monfroy, allocataire MESR
Jean-Luc Moysset, ATER (Nancy II, à compter du 1/10/94)
Pauline Strogova, boursière INRIA
Laurent Vigneron, allocataire MESR (jusqu'au 30/09/94)
Marian Vittek, boursier CNOUS (jusqu'au 30/09/94) puis INRIA (jusqu'au 30/11/94)

Chercheurs post-doctorants

Hendrik Lock, post-doc INRIA (Capital Humain et Mobilité)
Christopher Lynch, maître de conférence associé (Université H. Poincaré) puis post-doc INRIA
Toby Walsh, post-doc INRIA (jusqu'au 18/03/94)

2 Présentation du projet

L'objectif du projet est de contribuer à la conception et à la réalisation d'outils permettant de faire des preuves de propriétés de logiciels. Le projet comporte donc trois volets principaux dont l'intégration est l'un de nos objectifs majeurs.

D'une part nous étudions la résolution de contraintes symboliques (c'est-à-dire dans les algèbres de termes) et numériques (i.e. dans des domaines numériques tel que les réels ou les entiers naturels).

En déduction automatique, nos résultats vont dans deux directions complémentaires. D'une part l'utilisation de contraintes pour exprimer des stratégies de déduction complètes et d'autre part l'étude des preuves par récurrence, leurs applications étant tout particulièrement importantes pour la preuve de propriétés de programmes. Par ailleurs, nous continuons l'étude de la parallélisation de procédures de déduction et l'implantation de la réécriture parallèle dans le cadre de modèles de machines à mémoire distribuée.

Enfin en nous appuyant sur les résultats et techniques développés dans les deux volets précédents, nous abordons le problème de la preuve de propriétés de logiciels.

Parmi les points marquants de cette année 1994 mentionnons les suivants.

- Les résultats sur la combinaison de résolveurs de contraintes ont été étendus à une large classe de théories équationnelles qui peuvent partager des symboles.
- Le logiciel SPIKE permettant d'automatiser le raisonnement par récurrence est maintenant diffusé.
- Un cadre logique basé sur la logique de réécriture et permettant de spécifier et d'exécuter des procédures de déduction, de résolution et de manipulation de contraintes est maintenant opérationnel.
- L'utilisation d'automates d'arbres pour résoudre certain systèmes d'équations et de diséquations dans les théories associative et commutative ou dans le λ-calcul simplement typé, contraintes particulièrement utiles pour les preuves de complétude de définition.
- Notre connaissance de la déduction avec contraintes, en particulier associative et commutative a été complétée par la mise au point de nouvelles stratégies de paramodulation avec contraintes.

3 Actions de recherche

3.1 Contraintes

Notre travail sur la résolution et la satisfiabilité de contraintes fournit des outils pour la déduction, mais aussi utilise nos résultats en mécanisation du raisonnement. En parallèle aux travaux mentionnés ci après, nous avons mené une réflexion sur la restructuration complète

de la bibliothèque d'algorithmes d'unification UNIF pour aller vers une implantation modulaire tout particulièrement du point de vue de la combinaison des algorithmes.

3.1.1 Combinaison de résolveurs de contraintes

Participants : Eric Domenjoud, Hélène Kirchner, Christophe Ringeissen.

La notion de contrainte permet d'accroître l'expressivité des systèmes de déduction par l'intégration d'un domaine de calcul qui peut être suivant les cas, les termes du premier ordre, les entiers, les réels, etc... Mais il ne s'agit en général que d'un seul domaine à la fois. Il est donc primordial de développer des outils de résolution de contraintes hétérogènes faisant intervenir plusieurs domaines de calcul. Un cas particulièrement important, résolu par M. Schmidt-Schauß en 1987, est celui de l'unification dans le mélange de théories équationnelles à signatures disjointes.

Nous nous sommes consacrés à poursuivre l'étude des techniques de combinaison, introduites par F. Baader & K. Schulz en 1992, pour les appliquer à d'autres types de contraintes. Nous avons présenté [20] des algorithmes d'unification dans des mélanges de théories non disjointes où les symboles partagés sont supposés satisfaire une propriété adaptée de constructeurs. Cette condition permet de s'assurer que l'union est une extension conservative des théories qui la composent. Mais cette hypothèse seule n'est pas suffisante pour résoudre le problème de la combinaison des solutions issues de chaque algorithme puisqu'une variable peut désormais s'instancier simultanément dans chaque théorie composant l'union par un terme formé de symboles partagés. Ce problème n'existe pas dans le cas disjoint, où les seuls termes partagés sont les variables. Des solutions sont proposées pour n'avoir à considérer qu'un ensemble fini de termes partagés. C'est aussi pourquoi nous avons étudié des problèmes d'unification spécifiques comme le filtrage et le problème du mot, où la combinaison des solutions, sous certaines conditions, ne pose pas de difficulté. En effet, lorsque les membres gauche et droit des axiomes ont même ensemble de variables (théories régulières), alors les solutions d'un problème de filtrage sont closes et combiner les solutions se réduit à tester l'égalité entre termes. La combinaison des algorithmes de filtrage avait déjà été étudiée par T. Nipkow en 1989, mais uniquement pour l'union disjointe de théories régulières. Nous avons résolu le cas disjoint en général. Dans la classe optimale des théories partielle-

ment linéaires que nous introduisons [36], la résolution d'un problème hétérogène spécifique comme le filtrage s'effectue par résolution de sous-problèmes purs tout aussi spécifiques. Dans toutes les autres théories, nous montrons que le filtrage est insuffisant puisque l'unification est alors nécessaire pour résoudre les sous-problèmes purs ayant perdu en spécificité. L'approche suivie permet d'appliquer la plupart de nos algorithmes de combinaison de l'unification, du filtrage ou du problème du mot au problème de satisfaisabilité qui revêt une importance grandissante en programmation logique et en déduction automatique avec contraintes. Nous proposons par ailleurs dans [10] un cadre formel pour la combinaison de résolveurs de contraintes symboliques dans des structures engendrées par des termes. Le langage de contraintes et le résolveur qui en résultent sont utilisés pour l'intégration d'une structure prédéfinie au mécanisme général et incrémental de résolution par surréduction contrainte [29].

3.1.2 Contraintes d'ordre supérieur

Participants : Claude Kirchner, Denis Lugiez.

Les contraintes égalitaires d'ordre supérieur permettent d'exprimer des propriétés particulièrement intéressantes comme celle de la complétude d'un schéma (par exemple de récurrence) sur une structure de données. Les éléments de base sont alors les fonctions et les fonctions définies et étudiées sont les fonctionnelles. Il est alors utile d'étudier les contraintes égalitaires qui expriment des propriétés intéressantes de ces objets. Mise à part la question fondamentale de l'unification que nous étudions plus loin, le problème le plus significatif est le problème de compléments qu'on écrit:

$$\exists \bar{x} \forall \bar{y} : t \neq t_1 \wedge \ldots \wedge t \neq t_n$$

où $\bar{x}$ représente les variables de t et $\bar{y}$ les variables de $t_1, \ldots, t_n$.

Nous avons étudié ce type de problème dans le cadre plus général de la disunification qui a pour but de résoudre les formules générales construites sur le prédicat d'égalité dans le cadre d'une sémantique basée sur les termes clos. Cette question est trivialement indécidable, mais de plus, elle n'est même pas semi-décidable, voir la preuve (simple) de [33]. Nous avons par contre obtenu des résultats de décidabilité du problème de

complément d'ordre 2 dans le cas où celui-ci est top-linéaire, généralisation de la notion de linéarité classique. De même nous avons montré que les formules quelconques construites sur le prédicat d'égalité sont décidables lorsque tous les termes considérés sont des *patterns*, c.à.d. les arguments des variables libres d'un terme sont des variables liées distinctes. Ces résultats sont résumés dans [33] et forment une partie de l'habilitation de Denis Lugiez (soutenue en janvier 95).

Cas particulier des contraintes précédentes, l'unification dans le λ-calcul simplement typé est l'un des mécanismes fondamentaux de déduction dans les logiques d'ordre supérieur. Bien que le problème soit en général indécidable, il est important d'avoir des procédures qui se comportent sur les cas décidables le plus efficacement possible, et de comprendre en profondeur comment les implémenter. En particulier les rôles joués par l'opération de substitution, dans le cas du λ-calcul, pour réaliser soit la β-réduction soit l'unification, doivent être bien compris.

C'est dans ce contexte que nous avons étudié avec Gilles Dowek (Projet Coq) et Thérèse Hardin (Projet Para) comment utiliser une théorie de substitution explicite pour concevoir un nouvel algorithme d'unification pour le λ-calcul simplement typé. Ne nous intéressant pas à la théorie de substitution explicite elle même mais à son utilisation pour l'unification d'ordre supérieur, nous avons utilisée la théorie des substitutions explicites initialisée par Abadi, Cardelli, Curien et Lévy, appelée $\lambda\sigma$. Pour cette théorie équationnelle du premier ordre, nous avons décrit une procédure complète de résolution des problèmes d'unification sans variable de substitution. Cette procédure est décrite par des règles de transformation dont l'application, quand elle termine, fournit des formes résolues basées sur une notion très similaire à celle d'équation flexible-flexible. La résolution d'un problème d'unification entre λ-termes consiste alors à transformer ce problème en un problème d'unification du premier ordre modulo la théorie équationnelle $\lambda\sigma$. Ce problème étant résolu, nous avons montré comment traduire les solutions afin d'obtenir une description des solutions du problème initial. Cette approche a l'avantage de permettre une décomposition de l'algorithme de G. Huet en étapes élémentaires, toutes basées sur une théorie équationnelle du premier ordre, et qui peuvent être facilement implantées. Ce travail a été présenté à UNIF-94 [22, 41].

3.1.3 Contraintes associatives-commutatives

Participants : Denis Lugiez, Jean-Luc Moysset.

Nous avons continué l'étude des problèmes de compléments associatifs-commutatifs

$$\exists \bar{x} \forall \bar{y} : t \neq_{AC} t_1 \wedge \ldots \wedge t \neq_{AC} t_n$$

et développé particulièrement l'approche par automates d'arbres conditionnels. Nous avons montré que le problème de complément est NP-difficile en codant le problème 3-SAT. Ce codage n'utilise que des termes constructeurs linéaires, cas particulier pour lequel il existe un algorithme polynomial non-déterministe. Donc même ce cas très simple est NP-complet, alors que l'algorithme de filtrage correspondant est polynomial. Cela explique la complexité des algorithmes de compilation du filtrage et de vérification de complétude de définition présentés dans la thèse d'université de J.-L. Moysset qui sera soutenue début 95. Les résultats les plus intéressants sont ceux concernant le cas où les termes sont semi-linéaires (toutes les occurrences d'une variable non-linéaire sont sous le même noeud dans le terme aplati) pour lequel nous avons défini les automates conditionnels. Cette approche a permis d'écrire des preuves plus simples et plus modulaires ce qui permet de généraliser les résultats à des structures de données tels que les multi-ensembles et ensembles ainsi qu'au problème de la réductibilité inductive [12].

3.1.4 Contraintes Diophantiennes linéaires

Participants : Farid Ajili, Eric Domenjoud, Claude Kirchner, Hendrik Lock.

Les contraintes Diophantiennes sont constituées de combinaisons d'équations, d'inéquations et de diséquations linéaires homogènes ou non et dont chacun des membres est un polynôme à coefficients entiers. Nous étudions ces problèmes depuis plusieurs années au travers de l'étude du filtrage et de l'unification associative et commutative. L'idée générale que nous avons développée cette année est d'utiliser les techniques employées en démonstration automatique pour permettre de résoudre des problèmes rencontrés en programmation par contraintes, en particulier pour la planification de production. Cela nous a amené à mettre en oeuvre de nouveaux algorithmes qui, à leur tour, sem-

blent particulièrement attractifs pour les applications de démonstration automatique.

Dans ce cadre, Farid Ajili a fait dans son DEA [40] un survol et une présentation unifiée des différents algorithmes de résolution de systèmes d'équations Diophantiennes linéaires. Il a par ailleurs proposé un nouvel algorithme de résolution d'inéquations Diophantiennes, basé sur une méthode évitant d'introduire des variables auxiliaires [13]. Cette approche devrait permettre des gains importants en efficacité de résolution. Sa généralisation à des systèmes d'inéquations est en cours.

Par ailleurs Eric Domenjoud a proposé un algorithme original de résolution de contraintes Diophantiennes linéaires (équations, diséquations, inéquations) basé sur l'algorithme de Mac Mahon et Elliot [21]. Cet algorithme reste à tester et à comparer en pratique avec les algorithmes existants. De par sa simplicité conceptuelle, il offre des possibilités de parallélisation intéressantes.

Hendrik Lock a enrichi notre expérience issue des techniques de résolution de contraintes Diophantiennes pour la déduction automatique, par sa connaissance des techniques de propagation et d'implication de contraintes utilisées typiquement en programmation par contraintes [7, 19]. Il a montré comment coder un langage basé sur les formules booléennes en contraintes Diophantiennes linéaires sur des domaines finis. Cette transformation préserve la sémantique des formules considérées et permet d'utiliser les méthodes d'implication connues sur les domaines finis [32]. Par ailleurs, des stratégies de propagation de contraintes sont en cours d'étude pour permettre de meilleures performances, en particulier sur les algorithmes de planification de production.

3.1.5 Contraintes sur les termes schématisés

Participants : Ali Amaniss, Denis Lugiez.

Les méthodes de schématisation sont des outils qui sont particulièrement intéressants dans le cadre de la démonstration automatique, car ils permettent d'augmenter l'expressivité et autorisent à représenter de manière finie des ensembles infinis de termes. Par exemple, l'ensembles des entiers pairs peut s'écrire $s^{2n}(0)$ dans le formalisme dit des termes récurrents. Nous avons fait une étude comparative des méthodes de schématisation et nous avons étudié comment adapter la règle de résolution en utilisant les termes récurrents mentionnés ci-dessus. En-

suite nous avons montré que l'extension apportée par n'importe quelle schématisation dont l'unification entre les termes est décidable, laisse inchangée la correction et la complétude de la règle de résolution. Nous avons ensuite donné un algorithme d'*inclusion* qui est une sorte de filtrage qui nous a servi à définir un algorithme de généralisation entre les termes récurrents. Cet algorithme de généralisation est utilisé pour engendrer automatiquement des concepts généralisant plusieurs objets donnés, technique utile dans la manipulation de preuve (pour définir des schémas communs).

3.1.6 Contraintes réelles non linéaires

Participants : Eric Monfroy, Michaël Rusinowitch.

La résolution de systèmes d'équations polynomiales non linéaires au sein d'un langage de programmation logique à contraintes nécessite l'utilisation de méthodes algébriques telles que les Bases de Gröbner, ou l'élimination de quantificateur. Cependant, ces résolveurs étant très complexes, il n'est pas possible pratiquement de créer un système "universel" capable de traiter tous les problèmes faisant intervenir ce type de contraintes. C'est pourquoi nous travaillons à la réalisation d'un résolveur adapté à une classe de formules issue de la planification de trajectoires de robots mobiles (cette approche a été présentée aux Journées de Géométries Algorithmiques 94 [35]). Pour cela, il a fallu répertorier les déplacements sous forme de déplacements primaires et en extraire les contraintes qu'ils impliquent. La phase suivante consistera en la spécialisation et la coopération de résolveurs algébriques manipulant le système obtenu.

3.2 Déduction automatique

Notre travail sur le raisonnement par récurrence a donné lieu à des résultats particulièrement intéressants car nécessitant très peu d'interaction avec l'utilisateur. Le logiciel SPIKE qui implante ces travaux est maintenant distribué.

Nous avons également travaillé sur les preuves par analogie, la parallèlisation de la déduction et les procédures de preuves avec contraintes. Ces dernières ont donné lieu à des résultats très prometteurs, en particulier dans le cas associatif et commutatif, en permettant d'éviter l'instanciation des termes.

Enfin notre activité dans le domaine s'est également traduite par l'organisation de la douzième conférence CADE sur la déduction automatique à Nancy (voir la section 5.4).

3.2.1 Preuves par induction

Participants : Narjès Berregeb, Adel Bouhoula, Maria Huber, Michaël Rusinowitch, Toby Walsh.

Adel Bouhoula a développé dans sa thèse [3] plusieurs procédures de preuves par induction implicite [6, 15] avec l'objectif d'élaborer un système de preuve automatique, nommé SPIKE, nécessitant un minimum d'interaction avec l'utilisateur. Le principal avantage des méthodes proposées est qu'elles permettent l'utilisation de nombreuses techniques de simplification telles que la réécriture contextuelle et l'analyse par cas. Elles permettent aussi la manipulation des équations non orientables et la simplification des conjectures entre elles, qui simule la récurrence simultanée. Nous avons également généralisé la récurrence implicite pour les spécifications paramétrées [16, 18]. Ce cadre permet d'avoir des preuves courtes et structurées. D'autre part, la nouvelle procédure limitée aux spécifications booléennes non paramétrées, est complète par réfutation, même si les fonctions ne sont pas suffisamment complètes et s'il y a des relations entre les constructeurs.

Nous avons proposé une procédure de test de complétude suffisante opérationnelle pour les spécifications conditionnelles paramétrées et structurées [18]. La classe des spécifications étudiée est celle où d'une part, l'ensemble des symboles de fonctions non paramétrées est subdivisé en un ensemble de constructeurs et un ensemble de symboles de fonctions qui sont définies à l'aide de ces constructeurs, et d'autre part, le corps de la spécification doit être présenté sous la forme d'un système de réécriture structuré et décroissant. Cette procédure permet de vérifier qu'un symbole de fonction est opérationnellement suffisamment complet sur des constructeurs, en construisant un arbre de motifs et en testant la réductibilité par cas de toutes les feuilles de l'arbre. La méthode proposée n'interdit pas l'utilisation des relations entre les constructeurs pour assurer sa correction. A cela il faut ajouter que si l'on se restreint aux règles conditionnelles à préconditions booléennes, notre procédure apporte à l'utilisateur une aide à la construction de définitions complètes [17].

Un exposé invité a été donné par Adel Bouhoula au workshop international "Automation of Proof by Mathematical Induction" [47].

Narjès Berregeb et Michaël Rusinowitch ont étudié l'extension de SPIKE à des théories associatives et commutatives (AC). Ils ont étendu les méthodes de calcul des ensembles tests au cas de théories associatives et commutatives, et proposé quelques techniques d'optimisation. Ils ont également étendu les règles d'inférences de SPIKE de façon à pouvoir traiter les symboles AC. La correction et la complétude du système étendu sont en cours d'étude. Ces extensions ont été implantées dans SPIKE. Les expérimentations montrent que l'on obtient ainsi des preuves plus naturelles et avec moins de lemmes donc moins d'interactions.

Toby Walsh a développé une procédure pour reconnaître les causes de divergence des preuves par récurrence (explicite ou implicite) et proposer des lemmes. Cette procédure, fondée sur la notion de *difference matching* [1], a été implantée dans une version du système SPIKE par Iskander Kort, stagiaire de l'Ecole d'Ingénieur de Tunis [44]. Le programme a permis de prouver un certain nombre de théorèmes sans assistance [39] alors que le système de Boyer-Moore nécessite plusieurs interactions sur ces problèmes.

L'étude du langage des termes clos en forme normale par rapport à un système de réécriture est fondamentale pour montrer la complétude suffisante ou la réductibilité inductive dans le cadre des preuves par récurrence. Maria Huber en collaboration avec Dieter Hofbauer (Université Technique de Berlin) a publié dans [8] un algorithme de décision de la régularité du langage des termes clos en forme normale. Puis avec Dieter Hofbauer et Grégory Kucherov (projet EURECA), Maria Huber a poursuivi ce travail en étudiant sous quelles conditions le langage des termes clos réductibles par un système de réécriture est un langage d'arbres algébrique. Les résultats de cette étude ont été présenté à la conférence internationale CAAP à Edinburgh (Écosse)[27]. Ce travail a dégagé une nouvelle classe intéressante de langages algébriques appelés langages *coréguliers* ou *top-context-free*.

En liaison avec les problèmes de l'induction implicite, Michaël Rusinowitch s'est intéressé avec Grégory Kucherov (projet EURECA), aux *systèmes de réécriture de mots avec variables*. Ces systèmes sont liés à

[1] D. Basin, T. Walsh, Difference Unification, 13th International Joint Conference on Artificial Intelligence, Chambéry, (France)

la théorie des *langages de motifs* qui est actuellement développée par A. Salomaa et son équipe. Nous avons démontré que pour de tels systèmes la propriété de réductibilité inductive d'un terme est indécidable [11]. Cette propriété joue un rôle important dans la théorie de la réécriture et ses applications. Elle signifie que toutes les instances closes du terme sont réductibles par le système de réécriture. Le résultat généralise et renforce un résultat connu de D. Kapur, P. Narendran, D. J. Rosenkrantz, H. Zhang[2]. Dans [31] la complexité du problème est explorée dans le cas où les variables du système sont linéaires. L'article montre que le problème de réductibilité inductive d'un terme linéaire pour un système linéaire est *co-NP*-complet.

3.2.2 Complétion avec sortes ordonnées

Participants : Claus Hintermeier, Claude Kirchner, Hélène Kirchner.

Les présentations équationnelles avec sortes ordonnées permettent de spécifier des fonctions partiellement définies et des sous-types tout en restant dans un cadre algébrique. Leur étude se poursuit dans l'équipe depuis plusieurs années et constitue le sujet de thèse de C. Hintermeier. Dans [24], nous montrons comment calculer dans les algèbres avec sortes ordonnées définies axiomatiquement par des formules égalitaires, d'appartenance ou d'existence. Un système de déduction complet est donné et intègre l'interaction entre toutes ces formules. Afin de rendre ce calcul plus opérationnel, la notion de termes décorés est proposée pour mémoriser dans un terme l'information locale de type et son enrichissemnt au cours de la déduction, au moyen d'un processus de réécriture agissant à la fois sur les termes et sur leurs décorations. Une procédure de complétion pour les présentations équationnelles avec sortes ordonnées est décrite. Le système de règles complété permet de prouver non seulement des théorèmes égalitaires du type $(t = t')$, mais aussi des formules d'appartenance $(t : A)$ d'un terme t à une sorte A.

Dans ce cadre algébrique, les termes décorés et la réécriture fournissent donc une sémantique opérationnnelle de la déduction et des procédures de décision pour les formules précédentes. Dans [25], nous étudions une propriété indécidable, appelée héritage de sortes, qui est nécessaire pour faire de l'unification dans ces théories incluant des formules d'ap-

[2]D. Kapur, P. Narendran, D. J. Rosenkrantz, H. Zhang, *Sufficient-completeness, ground-reducibility and their complexity*, Acta Informatica, 28 (1991), p.311-350

partenance. Une condition syntaxique incluse dans une procédure de complétion de règles de réécriture permet de la tester. La complétion est plus ou moins complexe suivant le type de formules d'appartenance $(t : A)$ autorisé dans la présentation: par ordre croissant de difficulté, on résoud le cas où t est un terme plat et linéaire, puis le cas où t n'a de variables qu'à une profondeur au plus un, enfin le cas sans restriction sur t.

Une comparaison de l'expressivité des règles conditionnelles et des règles avec sortes ordonnées est proposée dans [26]. On y montre que la classe des systèmes de réécriture conditionnelle confluents sur les termes clos, décroissants et sans variable supplémentaire dans la condition (CTRSs), coincide avec la classe des systèmes de réécriture confluents sur les termes clos et terminants (TRSs), dans le sens suivant: toute fonction calculable par un CTRS de la première classe peut aussi être implantée par un TRS de la deuxième. La construction du TRS confluent sur les termes clos et terminant associé à un CTRS satisfaisant les propriétés mentionnées est complètement décrite et peut être automatisée. Cette construction suffit à prouver l'équivalence des deux classes. Le TRS obtenu est linéaire gauche et trivialement localement confluent. Cette transformation est extrêmement intéressante dans le cadre de la réécriture concurrente puisqu'elle permet de remplacer de la réécriture conditionnelle par de la réécriture standard, bien sûr au prix d'un bien plus grand nombre de règles.

3.2.3 Déduction avec contraintes

Participants : Hélène Kirchner, Christopher Lynch, Michaël Rusinowitch, Laurent Vigneron.

La déduction automatique dans les théories présentées par des systèmes d'équations est considérée comme un problème difficile. En effet le raisonnement par chaînage arrière pose des problèmes techniques de complétude. Le raisonnement par chaînage avant est source d'explosion combinatoire. Plotkin, à la suite des travaux de Robinson et Wos, a proposé de décomposer les inférences en deux composantes; une composante logique générale, traitée par résolution, paramodulation, et une composante relative à la théorie équationnelle considérée. Cette dernière s'exprime par des algorithmes spécifiques d'unification et de filtrage. La thèse de Laurent Vigneron [4] s'inscrit dans cette problématique.

Laurent Vigneron a proposé un système complet de déduction pour les théories équationnelles régulières [38]. Un avantage décisif de ce système est qu'il autorise la plupart des techniques connues de contrôle de l'espace de recherche : blocage des remplacements dans les variables, simplifications par réécriture, restrictions fondées sur un ordre ... Dans le cadre des théories associatives-commutatives la présentation des inférences par des contraintes permet de raffiner encore les déductions. En particulier Laurent Vigneron a montré qu'il est possible de remplacer l'unification associative-commutative, dont l'implantation est très coûteuse, par un test d'unifiabilité associative-commutative [37] ce qui réduit la complexité d'une exponentielle. Il a également caractérisé le blocage des variables dans les extensions. Les algorithmes issus de ces travaux sont implantés dans le système DATAC écrit en CAML Light. DATAC permet de nombreuses stratégies de déduction dans les théories équationnelles. En particulier, le logiciel implante les déductions basique et contrainte. C'est un outil d'expérimentation précieux qui devrait permettre la mise au point de stratégies efficaces. Des tests ont été effectués sur des problèmes liés aux algèbres de Kleene ou à des structures de treillis (lemme de SAM). Il semble que la stratégie basique donne les meilleurs résultats, mais la résolution de contrainte d'ordre n'est pas complètement achevée ce qui biaise un peu les résultats.

Les techniques de réécriture et de résolution de contraintes sont privilégiées dans la plupart de nos travaux en déduction automatique, déjà mentionnés ci-dessus. L'automatisation des preuves dans les théories incluant le prédicat d'égalité a motivé l'introduction de la réécriture et de plusieurs de ses extensions. Dans un article de synthèse [30] présenté en 93 à l'Ecole de Printemps d'Informatique Théorique consacrée à la réécriture, trois de ses extensions sont présentées: la réécriture ordonnée, la réécriture modulo une théorie équationnelle et la réécriture avec contraintes. L'évolution qui conduit de la généralisation des deux premières notions à la troisième y est mise en valeur.

Un autre exposé invité à l'Ecole de Printemps d'Informatique Théorique consacrée aux contraintes en 94 a été fait sur le thème de l'utilisation des contraintes en déduction automatique. Le papier correspondant [43] présente trois approches utilisant des contraintes en déduction automatique, chacune d'elles mettant en avant un intérêt différent de leur utilisation. L'expression de stratégies grâce à des contraintes est démontré sur l'exemple d'une procédure de complétion utilisant des stratégies

ordonnées et basiques. La schématisation, via les contraintes, de problèmes d'unification complexes est illustré par l'exemple d'un prouveur de théorèmes équationnels dans des théories associatives et commutatives. L'incorporation de théories prédéfinies dans un processus de déduction est faite dans le cadre d'une procédure de surréduction qui résoud des requêtes dans des théories définies par des règles de réécriture avec des contraintes prédéfinies. Les avantages des contraintes sont explicités et les problèmes ouverts mentionnés.

Dans [34], nous présentons une modification du système d'inférence basé sur la paramodulation dans laquelle l'égalité sémantique et les littéraux non-égalitaires sont stockés comme contraintes dans chaque clause. Ces contraintes sont créées quand de nouvelles clauses sont engendrées et elle sont héritées par tous les descendants de cette clause. Elles peuvent alors être utilisées pour faire de la démodulation et de la simplification par des clauses unitaires, si certaines conditions sont satisfaites. Ceci restreint l'espace de recherche de la procédure et la longueur des preuves obtenues. Nous montrons que ce processus est correct, complet et compatible avec les règles d'élimination de redondances. Nous montrons aussi la relation entre cette technique et l'élimination de modèles.

Dans [45], nous montrons que pour le système d'inférence basé sur la superposition, la sélection de n'importe quel littéral dans une clause est une stratégie complète dans le cas de clauses de Horn. Nous montrons quelles éliminations de redondances préservent la complétude. La sélection de littéraux positifs non-maximaux nécessite le développement d'une nouvelle technique de preuve appliquée dans ce papier. Un cas particulier de notre résultat est la règle de sélection qui choisit un littéral positif dans chaque clause qui en contient un. Ceci donne une stratégie complète pour les clauses de Horn équationnelles, qui se réduit à la SLD-résolution dans le cas non-équationnel. Elle est dirigée par le but dans le sens où les littéraux dans le corps d'une clause ne doivent être résolus que si la tête est utilisée dans une inférence avec le but. C'est donc une manière de combiner les techniques de réécriture avec l'orientation par le but. Ce résultat s'applique à la surréduction conditionnelle, car il donne une propriété de confluence pour laquelle la surréduction conditionnelle est complète, sans restriction sur les positions des variables dans les clauses. Il s'applique aussi à la surréduction conditionnelle basique.

3.2.4 Outils pour la preuve par analogie

Participants : Régis Curien, Denis Lugiez.

La vérification et la preuve de programmes sont des problèmes essentiels en programmation, mais les preuves sont en général difficiles et fastidieuses. Il importe donc de proposer des outils automatiques pour aider la réalisation de ces preuves. Une idée fondamentale est de chercher à réutiliser des preuves déjà effectuées lorsqu'on doit prouver une nouvelle propriété. On parle alors de preuve par analogie. Les travaux développés consistent à faire apparaître l'analogie directement entre une formule déjà prouvée et la formule à prouver, puis à construire mécaniquement la preuve cherchée en utilisant l'analogie. Ceci n'est possible qu'au terme d'un effort de formalisation du concept d'analogie. Pour cette formalisation, les preuves par expansion, développées par Miller et Pfenning, se sont avérées pertinentes, car elles préservent la structure des formules. Ainsi, il est possible, à partir de l'analyse de la similitude, ou de la différence entre les formules, de proposer des algorithmes transformant automatiquement la preuve prise comme référence en la preuve recherchée. Une étape fondamentale de ce travail est donc de définir formellement ce que sont une différence ou une similitude entre formules. Cette opération est basée sur l'utilisation d'un algorithme de filtrage d'ordre deux capable de tenir compte de certaines propriétés algèbriques des connecteurs logiques. En définitive, il a été montré qu'il est possible dans certains cas de produire des outils automatiques de transformation de preuves, correspondant à des notions formelles d'analogie. Ces résultats constituent la thèse d'université de Régis Curien qui sera soutenue en janvier 95.

3.2.5 Déduction concurrente

Participants : Iliès Alouini, Claude Kirchner, Christopher Lynch.

Notre travail sur la réécriture concurrente nous a amené cette année à conforter le modèle et l'implantation.

Compte tenu des spécificités du modèle de réécriture concurrente, et après étude des approches existantes, nous avons conçu un glanneur de cellules adapté à notre modèle. Ce glanneur a été intégré à l'implantation courante (écrite en C et utilisant l'environnement PVM) et permet des gains d'efficacité en occupation processeur et en temps

tout à fait appréciable. Le problème qu'il a fallu résoudre est le suivant: la réécriture concurrente réécrit une forêt de graphes acycliques en une autre telle forêt. Or l'information pertinente de cette forêt est un unique arbre distingué, descendant du terme à réduire initial. Les autres arbres, que l'on peut voir comme constitués de processus actifs qui s'autoréduisent, constituent les structures à glanner. Ce glannage est donc d'autant plus important que les cellules à ramasser étant actives, elles génèrent d'autres cellules inutiles. La solution que nous avons conçue consiste à exécuter des processus complètement asynchrones qui exécutent indépendamment un algorithme atomique et local de glannage [14].

Nous étudions maintenant comment intégrer la réécriture conditionnelle en traduisant systématiquement de tels systèmes en systèmes non conditionnels.

Par ailleurs, nous commençons à étudier le lien entre les systèmes de calcul et la réécriture concurrente, en particulier pour étudier la parallèlisation de procédures de déduction [48].

3.3 Validation de propriétés de logiciels

L'élaboration de méthodes et d'outils de preuve de propriétés de logiciels est l'un de nos objectifs majeurs. Nous nous appuyons dans cette section sur les résultats et techniques développés dans les deux sections précédentes. Dualement, les problèmes que nous rencontrons ici enrichissent notre problématique sur la résolution de contraintes et la déduction automatique.

Cette année nous avons continué notre travail sur un cadre logique basé sur la logique de réécriture et dont l'implantation est appelée ELAN. Nous avons en particulier expérimenté l'utilisation de ce cadre pour décrire des procédures de preuve par complétion de règles de réécriture. La réalisation de preuve de terminaison, en particulier en utilisant des contraintes de type a été poursuivie. Enfin le problème du routage dans des réseaux réguliers constitue une application intéressante de nos travaux.

3.3.1 ELAN

Participants : Claude Kirchner, Hélène Kirchner, Marian Vittek.

ELAN est un cadre logique qui permet de formaliser, sur une base uniforme, différentes logiques et différents résolveurs de contraintes. L'idée d'ELAN est basée sur le fait que la sémantique des langages de programmation logique dans le sens général du terme, la construction de preuves de théorèmes et la résolution de contraintes peuvent être décrites de façon uniforme en donnant la syntaxe des formules, l'ensemble des axiomes et l'ensemble des règles de déduction. Les trois processus peuvent être vus comme des instances du même schéma d'application de règles de réécriture (de déduction) sur des formules, suivant une stratégie jusqu'à ce qu'une forme spéciale soit obtenue. Pour formaliser ce point de vue, nous utilisons la logique de réécriture, introduite par J. Meseguer[3], qui offre les caractéristiques des cadres logiques. En effet, il est possible de représenter différentes logiques en logique de réécriture, en transformant une théorie de la logique objet en une théorie de la logique de réécriture, telle que les preuves des formules dans les deux formalismes se correspondent. D'un autre côté, il est aussi naturel de décrire des résolveurs de contraintes en termes de règles de réécriture, c'est-à-dire encore comme des théories de la logique de réécriture. L'application de ces règles calcule dans ce cas une forme résolue des contraintes. Plusieurs résolveurs de contraintes réalisés dans le projet sont décrits dans ce cadre, par exemple la résolution des contraintes d'équations, diséquations ou inéquations sur les termes comparés avec l'ordre de simplification.

La logique de réécriture est la base de la sémantique du mécanisme d'évaluation d'ELAN. Les règles peuvent être conditionnelles et sont enrichies par une construction d'affectations locales. Mais nous avons ajouté à ce formalisme une notion de *stratégie*. Un *système de calcul* est une théorie de la logique de réécriture qui est enrichie par un ensemble de stratégies décrivant des calculs et contrôlant l'application des règles de réécriture. Elles sont utilisées à la fois pour décrire le déroulement des preuves menant à un résultat, et pour restreindre l'espace de recherche. Les stratégies s'expriment dans un langage d'expressions construites à partir de noms de règles et de stratégies, par l'opérateur de concaténation, les opérateurs d'itération et les opérateurs de choix. Ces derniers

[3] J. Meseguer, *Conditional rewriting logic as a unified model of concurrency*, Theoretical Computer Science, 96 (1992), p.73-155

permettent une gestion relativement fine de l'exploration de l'arbre de recherche.

L'environnement ELAN permet de définir et exécuter les systèmes de calcul. Ceux-ci sont décrits dans des modules qui définissent chacun leur signature, leurs règles de réécriture et leurs stratégies. Les modules peuvent être paramétrés et peuvent importer d'autres modules construisant ainsi des systèmes de calcul plus complexes à partir des systèmes de calcul plus simples. La modularité d'ELAN a permis la création d'une bibliothèque de modules standards, souvent utilisés par des programmes différents, et regroupés dans une bibliothèque. Ces travaux ont fait l'objet de la thèse de Marian Vittek [5] et d'un chapitre de [9].

Dans [28, 46], nous avons montré comment prototyper la complétion avec contraintes en utilisant ELAN et les systèmes de calcul en logique de réécriture. Ce travail peut être vu comme un premier outil de validation de programmes équationnels implanté en ELAN. Dans un premier temps une procédure de complétion avec contraintes a été conçue en favorisant les simplifications entre formules. Cela nous a permis de clarifier la notion de redondance d'une égalité au cours de la complétion avec contraintes. Le processus a ensuite été décrit par des systèmes de calcul en mettant à profit le langage de stratégies d'ELAN pour expérimenter différentes stratégies de simplification.

3.3.2 Preuves de terminaison

Participants : Thomas Genet, Isabelle Gnaedig-Antoine.

L'étude de la terminaison de la réécriture par codage des termes sur des algèbres typées a été poursuivie. Le principe de la méthode consiste à interpréter des systèmes de réécriture, non typés au départ, sur des algèbres multi-sortées, puis à utiliser une extension de l'ordre LPO sur ces algèbres multi-sortées, pour tirer parti de la finesse de ces dernières.

Ce codage des systèmes de réécriture est très simple et facilement automatisable: il consiste à définir autant de sortes qu'il y a d'opérateurs dans un système, puis à étiqueter les opérateurs par les sortes de leurs arguments dans les termes. Les variables, quant à elles, peuvent recevoir toutes les sortes possibles. Les nouveaux systèmes ainsi définis induisent une relation de réécriture multi-sortée, dont on a prouvé que la terminaison impliquait la terminaison de la relation initiale.

Nous avons montré que l'extension du LPO sur ces algèbres multi-sortées restait un ordre de simplification, et que, moyennant une condition de précédence simple, il permettait la preuve de terminaison de notre réécriture multi-sortée [42].

D'autre part, les méthodes classiques de preuve de terminaison peuvent être vues comme des contraintes d'ordre sur les termes. Une extension de nos travaux sur la résolution de contraintes d'ordre sur les termes avec variables est en cours. Contrairement à la procédure que nous avions proposée précédemment, il ne s'agit plus de savoir s'il existe des solutions pour l'ordre LPO, mais s'il existe un ordre de simplification quel qu'il soit, ce qui est plus général.

3.3.3 Routage dans les réseaux

Participants : Claude Kirchner, Pauline Strogova.

Les outils et les méthodes que nous élaborons dans le projet doivent démontrer leur intérêt sur des problèmes qui ne sont pas directement inspirés de notre problématique. C'est dans ce contexte que nous travaillons avec Dominique Fortin et Pauline Strogova de l'action Archi à Rocquencourt à l'application des méthodes de déduction automatique à la recherche de chemins dans des réseaux réguliers.

Nous nous intéressons en fait à des réseaux réguliers particuliers basés sur des graphes ayant une structure algébrique de groupe fini. Ces graphes, dits de Cayley, sont symétriques et très denses, c'est-à-dire pour le même nombre de nœuds leur connexité est plus faible. La symétrie offre une possibilité de coder les données du problème de façon plus simple, et la densité permet d'économiser du matériel. Notre objectif étant de concevoir un algorithme permettant de trouver k chemins parallèles disjoints, nous cherchons a mettre en œuvre des techniques de réécriture dans les groupes. Les chemins à rechercher dans le réseau correspondant aux éléments du groupe, on cherche à passer d'une expression quelconque de ces éléments à celle en termes de générateurs uniquement. De plus, on cherche à minimiser la décomposition en générateurs, en "optimisant" les produits.

Nous avons étudié cette année le problème dans le cas $k = 1$. L'algorithme de routage que nous avons décrit, permet de trouver un chemin minimal entre deux points d'un graphe de Cayley. Il utilise une présentation minimale du groupe correspondant, connue a priori. Il est possible

de trouver cette présentation à partir des éléments générateurs du groupe en utilisant une méthode basée sur la réécriture. L'algorithme modifié, compte tenu de cette méthode, a été présenté au Workshop sur des Graphes de Cayley à Lyon, en décembre 1993. Nous l'avons testé sur de nombreux exemples à l'aide du logiciel RRL. Cependant, bien que nous avons démontré théoriquement la terminaison des systèmes de réécriture considérés, les groupes de taille supérieure à 10^4 éléments posent des problèmes cruciaux d'efficacité. Ceci s'explique par le nombre exponentiel de paires critiques calculées vis à vis de la taille du système de réécriture initial. Nous avons exposé ces résultats au Workshop ARIA [23].

4 Actions industrielles

Au sein de l'équipe DEMOTHEO du GDR Programmation, PROTHEO a participé à la mise en place d'une proposition d'action de soutien de programmes pour le développement d'outils de démonstration pour la vérification de programmes de type industriel. Le projet se propose de poursuivre le travail de conception d'assistants à la démonstration, dont SPIKE, d'assurer en particulier un effort de distribution et de documentation, de développer des interfaces et d'appliquer les outils élaborés à des problèmes réels de type industriel. La présence d'une équipe du CNET au sein de ce projet permet de fournir de tels problèmes. L'action se terminera en septembre 95 par une journée de présentation des résultats obtenus, largement ouverte aux industriels.

Par ailleurs, nos contacts avec ILOG permettront à Marian Vittek de faire un postdoc industriel entre cette entreprise et notre projet à partir d'avril prochain.

5 Actions nationales et internationales

5.1 Actions nationales

Nous sommes impliqués dans deux projets inter PRC du CNRS. D'une part le projet *Modèles de calculs* coordonné par Michel Parigot dont les sites principaux sont Lille, Marseille, Nancy, Orsay, Paris 7. D'autre part le projet *Mécanisation du raisonnement* coordonné par Jean-Pierre Jouannaud dont les sites sont Grenoble, Lyon, Nancy, Orsay, Rocquencourt, Toulouse.

Hélène Kirchner est responsable du projet groupé DEMOTHEO au GDR Programmation et Outils pour l'Intelligence Artificielle, sur le thème contraintes et déduction automatique. Ce projet regroupe les équipes PROTHEO, DEMONS du LRI et des chercheurs de Strasbourg et Rouen. Dans ce contexte, PROTHEO participe à une action du GDR Programmation sur le développement d'outils de démonstration pour la vérification de programmes de type industriel.

5.2 Actions internationales

Nous participons à un groupe de travail Basic Research Action d'ESPRIT, géré à Nancy par Hélène Kirchner qui, sous l'acronyme *COMPASS* et le titre *a COMPrehensive Algebraic approach to System Specification et development,* fait intervenir les universités d'Aarhus, Barcelone, Berlin, Braunschweig, Brême (Bernd Krieg-Brückner, coordinateur), Dresde, Edimbourg, Gênes, Lisbonne, Munich, Nancy, Orsay, Oslo, Oxford, Passau, ainsi que le LIENS, le Max-Planck Institut für Informatik à Sarrebrück, et l'Académie des sciences de Pologne.

Nous participons au groupe de travail Basic Research Action d'ESPRIT qui, sous l'acronyme *CCL* et le titre *Construction of Computational Logics,* fait intervenir les universités et les centres de recherche en informatique de Barcelone (Université : Fernando Orejas), Madrid (Université : Mario Rodriguez-Artalejo), Munich (Université : Tobias Nipkow), Nancy (INRIA-Lorraine : Claude Kirchner), Orsay (Université : Jean-Pierre Jouannaud), Sarrebrück (Université : Gert Smolka), Sarrebrück (Max Planck Institut : Harald Ganzinger) ainsi que la société COSYTEC (Mehmet Dincbas). Le groupe de travail *CCL* est à l'initiative de la conférence CCL-94 *Constraints in Computational Logic.*

Nous participons également au réseau d'excellence COMPULOG qui regroupe de nombreux sites européens travaillant sur la *programmation logique*.

Nous coordonnons le réseau Capital Humain et Mobilité *SOL*, créé en septembre 93, sur la résolution de contraintes en nombres entiers et sur les domaines finis. Ce réseau fait intervenir les universités et les centres de recherche en informatique de Lisbonne (Université : Pedro Barahona), Munich (ECRC : Alexander Herold), Nancy (INRIA-Lorraine : Claude Kirchner), Orsay (Université : Evelyne Contejean), Porto (Université : Miguel Filgueiras).

Nous faisons partie du réseau *IndusMind* sur les preuves par induction regroupant des équipes européennes et américaines et subventionné par Esprit du côté européen.

Nous entretenons des liens étroits avec les universités de Kaiserslautern et de Sarrebrück qui sont des universités proches ayant un grand laboratoire de recherche en informatique, ainsi qu'avec le DFKI et le MPI. Il se trouve que nous travaillons sur des sujets très voisins : la réécriture et la déduction concurrente à Kaiserslautern avec Jürgen Avenhaus et Klaus Madlener, les algèbres à sortes ordonnées et les contraintes associées avec Gert Smolka à Sarrebrück, la preuve automatique avec Harald Ganzinger à Sarrebrück (Max Planck Institut).

La coopération NSF-INRIA avec l'Université d'Urbana-Champaign (Illinois, USA) se poursuit sur le thème de la résolution de contraintes et ses applications. Elle implique du côté américain N. Dershowitz, C. Liu et E. Reingold.

La coopération NSF-CNRS avec l'Université d'Iowa (USA) se poursuit sur le thème de la démonstration automatique. Elle implique du côté américain H. Zhang.

Une coopération NSF-CNRS avec l'Université de l'Etat de New York à Stony Brook a débuté sur le thème de la déduction avec contraintes et la parallélisation des preuves. Elle implique L. Bachmair et I.V. Ramakrishnan du côté américain.

5.3 Séjours de chercheurs

Séjour d'Adel Bouhoula au Département d'Informatique de la Faculté des Sciences de Tunis (Tunisie, 29-3 Novembre 94).

Séjour d'un mois d'Eric Domenjoud à Porto (Portugal) dans le cadre du réseau HCM SOL.

Séjour de trois mois de Claus Hintermeier à Aarhus (Danemark, 14 Septembre - 13 Décembre 1994) dans le cadre du groupe de travail *COMPASS*.

Séjour de Claude Kirchner à Porto (Portugal, 24-28 Mai 94) dans le cadre du réseau HCM SOL.

Visite d'Hélène Kirchner à l'Université de Boston, et au Masachussets Institute of Technology (USA, 6-8 Janvier 94).

5.4 Organisation de colloques

Nous avons organisé du 26 Juin au 1 Juillet la douzième conférence CADE, qui est le principal forum de présentation de recherche en déduction automatique et dont la première édition a eu lieu au laboratoire national de recherche d'Argonne en 1974. Cette année un nombre record de 177 papiers ont été soumis, dont 76 acceptés. Le programme a comporté des présentations de contributions sélectionnées, des présentations invitées, des démonstrations de systèmes, des présentations de problèmes, des séances de travail, des tutoriels et une table ronde. Une innovation particulière, cette année, a été le grand nombre de workshops proposés, qui ont couvert la plupart des "thèmes chauds" en déduction automatique, et qui ont permis des discussions intenses entre spécialistes internationaux des domaines concernés. Une table ronde s'est focalisée sur la proposition QED. CADE-12 a attiré 251 participants, ce qui constitue le record absolu d'affluence à cette conférence et a été un succès scientifique important.

Denis Lugiez a organisé le workshop sur l'unification UNIF-8 au Val d'Ajol, du 25 au 27 juin 1994.

Michaël Rusinowitch et Toby Walsh ont co-organisé (avec A. Bundy) le troisième workshop sur les preuves par induction qui a eu lieu dans le cadre de CADE-12.

Hélène Kirchner a participé à l'organisation de la première conférence internationale CCL-94 *Constraints in Computational Logic*.

5.5 Participation à des comités de programmes et à des comités éditoriaux

Hélène Kirchner a été membre du comité de programme du *Third Symposium on Artificial Intelligence and Mathematics*, du comité de programme de CADE-12, du comité de sélection pour WADT-94 *Workshop on Abstracts Data Types*. Hélène Kirchner est membre du groupe IFIP WG 14.3 (Foundations of Systems Specifications).

Hélène Kirchner est membre du comité éditorial du journal *Annals of Mathematics and Artificial Intelligence*.

Claude Kirchner a été membre des comités de programmes des conférences LPAR-94 *Logic Programming and Automated Reasoning*, ALP-94

Algebraic and Logic Programming, CCL-94 *Constraints in Computational Logic*.

Claude Kirchner est membre des comités éditoriaux des revues *Journal of Symbolic Computation*, *Journal of Logic Programming* et *Journal of the Egyptian Mathematical Society*.

Michaël Rusinowitch a été membre du comité de programme des conférences CADE-12 et STACS-94 *Symposium on Theoretical Aspects of Computer Science*.

Claude Kirchner est éditeur invité d'un numéro spécial du journal *Theoretical Computer Science* consacré à la conférence RTA-93 [1].

Michaël Rusinowitch est co-éditeur avec J.-L. Rémy du projet EURECA d'un numéro spécial du *Journal of Symbolic Computation* sur les systèmes de réécriture conditionnels [2].

6 Diffusion des résultats

6.1 Actions d'enseignement

6.1.1 Enseignement universitaire

Nous sommes intervenus en particulier dans le cadre du DEA d'informatique de Nancy. Plus précisement nous avons réalisé les enseignements suivants :

Cours Environnements de preuves et réécritures: cours de techniques avancées du DEA d'informatique, Adel Bouhoula et Pierre Lescanne (projet EURECA).

Cours Déduction et programmation logique avec contraintes: cours de techniques et concepts spécialisés du DEA d'informatique, Claude et Hélène Kirchner.

Cours Logique I: cours de tronc commun du DEA d'informatique, Denis Lugiez.

Cours Logique II: cours de techniques et concepts spécialisés du DEA d'informatique, Michaël Rusinowitch.

Cours Théorie des langages et algorithmique avancée: cours de licence et maitrise d'informatique de l'Université Henri Poincaré, Denis Lugiez.

Autres enseignements: l'équipe comprend sept moniteurs et ATERs, qui participent aux activités d'enseignement des universités et écoles d'ingénieurs de Nancy.

6.1.2 Séminaires et formation permanente

Nous sommes intervenus dans les écoles d'informatique théorique sur les thèmes suivants :

Réécriture, école de printemps des Jeunes Chercheurs, GRECO de Programmation, Toulouse, Mars 94, Claude Kirchner.

Réécriture, résolution, preuve, 6ième école d'été européenne en logique, langage et information (ESSLI'94), Copenhagen, Août 94, Claude Kirchner.

On the use of constraints in automated deduction, école de printemps d'informatique théorique du LITP sur le thème des contraintes, Orsay, Mai 94, Hélène Kirchner.

Programmation logique, déduction automatique, école de printemps des Jeunes Chercheurs, GRECO de Programmation, Toulouse, Mars 94, Michaël Rusinowitch.

Par ailleurs, nous avons donné les séminaires suivants:

- Adel Bouhoula à la Faculté des Sciences de Tunis (Avril 1994), à l'Ecole normale supérieure de Lyon (Mai 1994), au FST & ENSI (Tunisie, Novembre 1994).
- Christophe Ringeissen au Laboratoire d'Informatique Fondamentale de Lille (Mars 1994).
- Michaël Rusinowitch à l'Université de Kaiserslautern (RFA, Novembre 93) et au SRI (Palo Alto, USA, Mars 94).

6.1.3 Jurys de thèses

Claude Kirchner, Hélène Kirchner et Michaël Rusinowitch ont été membres ou rapporteurs de 16 thèses d'université, d'état ou d'habilitation.

6.2 Participation aux manifestations

Outre les conférences et workshop internationaux CTRS, UNIF, ICLP, CADE, ICALP, ALP, CCL, STACS, CAAP où nous avons présenté nos travaux, nous avons participé aux évènements suivants:

Participation d'Adel Bouhoula au workshop "Abstract Data Types" et au Working Group Esprit COMPASS (Santa Margherita Ligure, Italie), au workshop "Automation of Proof by Mathematical Induction" (exposé invité à Nancy, Juin 1994), aux Journées du GDR Programmation (Lille, Septembre 1994).

Participation de Régis Curien aux Journées de l'opération inter-PRC Mécanisation du Raisonnement (Chamrousse, Novembre 94).

Participation de Claus Hintermeier au workshop "Abstract Data Types" (Santa Margherita Ligure, Italie), au "Nordic Workshop on Programming Theory" (Aarhus, Danemark).

Participation de Claude Kirchner aux Journées de l'opération inter-PRC Mécanisation du raisonnement (Toulouse, Février 94 et Chamrousse, Novembre 94), à la conférence PASCO'94 (conférence invitée à Linz, Autriche, Septembre 94), au workshop SOL (Paris, Novembre 94).

Participation d'Hélène Kirchner au "Third International Symposium on Artificial Intelligence and Mathematics" (Fort Lauderdale, Floride, 3-5 Janvier 94), au workshop "Abstract Data Types", au Working Group Esprit COMPASS (Santa Margherita Ligure, Italie, 30 Mai - 2 Juin 94), aux Journées du GDR Programmation (Lille, 21-23 Septembre 94).

Participation de Denis Lugiez aux Journées de l'opération inter-PRC Mécanisation du Raisonnement (Toulouse, Février 94).

Participation de Christophe Ringeissen aux Journées de l'opération inter-PRC Mécanisation du Raisonnement (Toulouse, Février 94).

Participation de Michaël Rusinowitch à la conférence SAC/ACM-SIGAPP (Phoenix, Arizona, Mars 94), au "Third International Workshop on Proofs by Induction" (Nancy, Juin 94).

Participation de Pauline Strogova au workshop "Automated Reasoning in Algebra" (Nancy, Juin 94) et au workshop "Cayley graphs in computer science" organisé par EBRA-group ASMICS (Lyon, Décembre 93).

Participation de Laurent Vigneron aux Journées du GDR Programmation (Lille, Septembre 94), aux Journées de l'opération inter-PRC Mécanisation du Raisonnement (Toulouse, Février 94 et Chamrousse, Novembre 94), au workshop "Theory Reasoning in Automated Deduction" (Nancy, Juin 94).

7 Publications

Livres et monographies

[1] C. KIRCHNER (réd.), *Special Issue on RTA-93, Theoretical Computer Science*, Elsevier Science Publishers B. V. (North-Holland), 1995.

[2] J.-L. RÉMY, M. RUSINOWITCH (réd.), *Journal of Symbolic Computation, Special Issue CTRS*, Academic Press, 1994.

Thèses

[3] A. BOUHOULA, *Preuves Automatiques par Récurrence dans les Théories Conditionnelles*, Thèse de Doctorat d'Université, Université de Nancy 1, mars 1994.

[4] L. VIGNERON, *Déduction automatique avec contraintes symboliques dans les théories équationnelles*, Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy 1, novembre 1994.

[5] M. VITTEK, *ELAN: Un cadre logique pour le prototypage de langages de programmation avec contraintes*, Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy 1, octobre 1994.

Articles et chapitres de livre

[6] A. BOUHOULA, M. RUSINOWITCH, «Implicit induction in conditional theories», *Journal of Automated Reasoning*, 1994, To appear.

[7] S. BREITINGER, H. LOCK, «Using Constraint Logic Programming For Industrial Scheduling Problems», *in : Logic Programming: Formal Methods and Practical Applications*, North Holland, 1994, to appear.

[8] D. HOFBAUER, M. HUBER, «Linearizing Term Rewriting Systems Using Test Set», *Journal of Symbolic Computation 17*, 1, 1994, p. 91–129.

[9] C. KIRCHNER, H. KIRCHNER, M. VITTEK, «Designing Constraint Logic Programming Languages using Computational Systems», *in : Principles and Practice of Constraint Programming. The Newport Papers.*, P. Van Hentenryck et V. Saraswat (réd.), MIT press, 1995, p. 131–158.

[10] H. KIRCHNER, C. RINGEISSEN, «Combining Symbolic Constraint Solvers on Algebraic Domains», *Journal of Symbolic Computation 18*, 2, 1994, p. 113–155.

[11] G. KUCHEROV, M. RUSINOWITCH, «On the ground reducibility problem for word rewriting systems with variables», *Information Processing Letters*, To appear.

[12] D. LUGIEZ, J.-L. MOYSSET, «Tree automata help one to solve equational formulae in AC-theories», *Journal of Symbolic Computation*, 1994, To appear.

Communications à des congrès, colloques, etc.

[13] F. AJILI, «Solving Diophantine Inequation without Adding a Slack Variable», *in : Proceedings of the second SOL Workshop*, E. Contejean (réd.), Orsay, novembre 1994.

[14] I. ALOUINI, «Garbage Collection for Concurrent Rewriting», *in : Proceedings of RTA'95*, 1995. To appear.

[15] A. BOUHOULA, M. RUSINOWITCH, «A General Format for Implicit Induction», *in : International Workshop on the Automation of Proof by Mathematical Induction*, Nancy (France), juin 1994.

[16] A. BOUHOULA, «Proofs in parameterized conditional specifications», *in : 10th ADT Workshop and 6th General Compass Meeting*, Genova (Italy), juin 1994.

[17] A. BOUHOULA, «SPIKE: a system for sufficient completeness and parameterized inductive proof.», *in : Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, A. Bundy (réd.), *Lecture Notes in Artificial Intelligence*, *814*, Springer-Verlag, p. 836–840, juin 1994.

[18] A. BOUHOULA, «Sufficient Completeness and Parameterized Proofs by Induction», *in : Proceedings Fourth International Conference on Algebraic and Logic Programming, Madrid (Spain), Lecture Notes in Computer Science*, *850*, Springer-Verlag, p. 23–40, septembre 1994.

[19] S. BREITINGER, H. LOCK, «Improving Search for Job Shop Scheduling with CLP(FD)», *in : Proceedings of Programming Language Implementation and Logic Programming, PLILP 94*, J. Penjam, M. Hermenegildo (réd.), LNCS 844, Springer, septembre 1994.

[20] E. DOMENJOUD, F. KLAY, C. RINGEISSEN, «Combination techniques for non-disjoint equational theories», *in : Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, A. Bundy (réd.), *Lecture Notes in Artificial Intelligence*, Springer-Verlag, p. 267–281, juin/juillet 1994.

[21] E. Domenjoud, «Extending Elliot/MacMahon's algorithm to arbitrary linear constraints», *in : Proceedings of the second SOL Workshop*, E. Contejean (réd.), Orsay, novembre 1994.

[22] G. Dowek, T. Hardin, C. Kirchner, «Higher-order unification via explicit substitutions», *in : Proceedings 8th International Workshop on Unification, Val d'Ajol (France)*, D. Lugiez (réd.), CRIN, juin 1994.

[23] D. Fortin, C. Kirchner, P. Strogova, «Routing in Regular Networks Using Rewriting», *in : Proceedings of the CADE international workshop on automated reasoning in algebra (ARIA)*, J. Slaney (réd.), p. 5–8, juin 1994.

[24] C. Hintermeier, C. Kirchner, H. Kirchner, «Dynamically-Typed Computations for Order-Sorted Equational Presentations –Extended Abstract–», *in : Proc. 21st International Colloquium on Automata, Languages, and Programming*, S. Abiteboul, E. Shamir (réd.), *Lecture Notes in Computer Science*, *820*, Springer-Verlag, p. 450–461, 1994.

[25] C. Hintermeier, C. Kirchner, H. Kirchner, «Sort Inheritance for Order-Sorted Equational Presentations», *in : To appear (95) in the LNCS Proceedings of the 10th WADT – 6th Compass Workshop*, 1994.

[26] C. Hintermeier, «A Transformation of Canonical Conditional TRS's into Equivalent Canonical TRS's», *in : Proceedings of the 4th International Workshop on Conditional Rewriting Systems, Jerusalem (Israel)*, juin 1994.

[27] D. Hofbauer, M. Huber, G. Kucherov, «Some Results on Top-context-free Tree Languages», *in : Proceedings 19th International Colloquium on Trees in Algebra and Programming, Edinburgh (U.K.)*, S. Tison (réd.), *Lecture Notes in Computer Science*, *787*, Springer-Verlag, p. 157–171, avril 1994.

[28] H. Kirchner, P.-E. Moreau, «Prototyping completion with constraints using computational systems», *in : Proceedings of RTA'95*, 1995. To appear.

[29] H. Kirchner, C. Ringeissen, «Constraint solving by narrowing in combined algebraic domains», *in : Proc. 11th International Conference on Logic Programming*, P. Van Hentenryck (réd.), The MIT press, p. 617–631, 1994.

[30] H. Kirchner, «Extensions of rewriting», *in : Proceedings of Ecole de Printemps d'Informatique Théorique sur la Réécriture (1993)*, *Lecture Notes in Computer Science*, Springer-Verlag, 1995. Research report CRIN 94-R-175.

[31] G. Kucherov, M. Rusinowitch, «The Complexity of Testing Ground Reducibility for Linear Word Rewriting Systems with Variables», *in :*

Proceedings 4th International Workshop on Conditional Term Rewriting Systems, Jerusalem (Israel), N. Dershowitz, N. Lindenstrauss (réd.), 1994.

[32] H. Lock, « From Problem Specifications to Diophantine Equations », *in : Proceedings of the second SOL Workshop*, E. Contejean (réd.), Orsay, novembre 1994.

[33] D. Lugiez, « Higher-order disunification: some decidable cases », *in : Constraints in Computational Logics*, J.-P. Jouannaud (réd.), *Lecture Notes in Computer Science*, *845*, Springer-Verlag, p. 121–135, septembre 1994.

[34] C. Lynch, « Local Simplification », *in : Constraints in Computational Logics*, J.-P. Jouannaud (réd.), *Lecture Notes in Computer Science*, *845*, Springer-Verlag, p. 3–18, septembre 1994.

[35] E. Monfroy, R. Schott, « Résolution de contraintes géométriques, applications en planification », *in : Actes Journées de Géométrie Algorithmique*, Val d'Ajol, mars 1994.

[36] C. Ringeissen, « Combination of matching algorithms », *in : Proceedings 11th Annual Symposium on Theoretical Aspects of Computer Science, Caen (France)*, P. Enjalbert, E. W. Mayr, K. W. Wagner (réd.), *Lecture Notes in Computer Science*, *775*, Springer-Verlag, p. 187–198, février 1994.

[37] L. Vigneron, « Associative-Commutative Deduction with Constraints », *in : Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, A. Bundy (réd.), *Lecture Notes in Artificial Intelligence*, *814*, Springer-Verlag, p. 530–544, juin 1994.

[38] L. Vigneron, « Proofs in Equational Theories », *in : Proceedings 8th International Workshop on Unification, Val d'Ajol (France)*, D. Lugiez (réd.), CRIN, juin 1994.

[39] T. Walsh, « A Divergence Critic », *in : Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, A. Bundy (réd.), *Lecture Notes in Artificial Intelligence*, *814*, Springer-Verlag, p. 14–28, juin/juillet 1994.

Rapports de recherche et publications internes

[40] F. Ajili, *Etude de la résolution de contraintes diophantiennes linéaires sur les entiers naturels*, Rapport de DEA, Université Henri Poincaré - Nancy 1, septembre 1994.

[41] G. Dowek, T. Hardin, C. Kirchner, « Higher-order unification via explicit substitutions », *rapport de recherche n° 94-R-243*, CRIN, décembre 1994.

[42] I. GNAEDIG, « Typing for termination », *rapport de recherche n° 94-R-219*, CRIN, 1994.

[43] H. KIRCHNER, « On the Use of Constraints in Automated Deduction », *rapport de recherche n° 94-R-176*, CRIN, 1994, Submitted to the Proceedings of Ecole de Printemps d'Informatique Théorique sur les Contraintes (1994), Springer-Verlag, LNCS.

[44] I. KORT, « Génération Automatique de Lemmes pour fixer la Divergence du Démonstrateur SPIKE », *Rapport de Stage de Fin d'Etudes*, CRIN, 1994.

[45] C. LYNCH, « Theorem Proving with Equational Horn Clauses: Any Selection Rule is Complete », *rapport de recherche n° 94-R-224*, CRIN, 1994, Submitted.

[46] P.-E. MOREAU, *Complétion avec contraintes en ELAN*, Rapport de DEA, Université Henri Poincaré - Nancy 1, septembre 1994.

Divers

[47] A. BOUHOULA, « The Challenge of Mutual Recursion and Mutual Simplification in Implicit Induction », Invited talk at the International Workshop on the Automation of Proof by Mathematical Induction, juin 1994.

[48] C. KIRCHNER, « Concurrent Theorem Proving: a Rewrite Based Approach », Invited presentation at PASCO-94, septembre 1994.

8 Abstract

The goals of the PROTHEO project are the design and implementation of tools for integrating program developments and proofs of properties, while taking advantage of the complementary nature of automated deduction, logics for programming and constraint solving.

We are interested in three research areas which strongly contribute to the previous objectives: first the problems of constraint satisfaction and solving, second the study of logics for programming or proving including the constraint paradigm, and third the proof of program properties. We are thus working on the following topics:

- Constraint solving,
- Automated deduction with constraints,
- Logic programming with constraints,
- Proofs by induction,

- Proofs of program properties,
- Parallelization of deduction processes.

We are developing several systems, in particular UNIF (a library of symbolic constraint solving algorithms), SPIKE (a theorem prover for first-order logic and inductive reasoning), ELAN (a logical framework based on the rewriting logic).

Table des matières