INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet PROTHEO

Contraintes, Déduction automatique et Preuves de Propriétés de Logiciels

Nancy

THÈME 2A

Table des matières

PROTHEO est un projet du LORIA (UMR 7503) commun au CNRS, à l'INRIA, à l'Université Henri Poincaré *Nancy 1, à l'Université Nancy 2 et à l'Institut National Polytechnique de Lorraine.*

1 Composition de l'équipe

Responsable scientifique

Hélène Kirchner [DR CNRS]

Responsable permanent

Michaël Rusinowitch [DR INRIA]

Secrétaire

Christelle Bergeret [TR INRIA]

Personnel INRIA

Adel Bouhoula [CR]
Olivier Bournez [CR, à partir du 1/10/99]
Isabelle Gnaedig-Antoine [CR]
Florent Jacquemard [CR]
Claude Kirchner [DR]
Christophe Ringeissen [CR]

Personnel CNRS

Eric Domenjoud [CR]
Nicolas Hermann [CR, en détachement Maître de Conférences, Université Henri Poincaré, jusqu'au 31/8/99]

Personnel Université

Alexander Bockmayr [Professeur, Université Henri Poincaré, Nancy 1]
Arnaud Durand [ATER, Université Nancy 2, jusqu'au 31/1/99]
Pierre-Etienne Moreau [ATER, Université Nancy 2, jusqu'au 31/8/99]
Christelle Scharff [ATER, Université Henri Poincaré, Nancy 1, jusqu'au 31/8/99]
Laurent Vigneron [Maître de Conférences, Université Nancy 2]

Ingénieurs experts

Gilles Défourneaux [jusqu'au 1/7/99]

Chercheurs post-doctorants

Alessandro Armando [Post-doc MAE jusqu'au 30/9/99]

Pierre-Etienne Moreau [Post-doc industriel INRIA à partir du 1/9/99]

Jianyang Zhou [Post-doc INRIA, co-financé Région Lorraine jusqu'au 1/10/99, puis Post-doc création d'entreprise]

Collaborateur extérieur

Jianny Zhou [depuis le 1/11/99]

Chercheurs invités

Abdessamad Imine [du 1/11/99 au 31/12/99]

Sofiene Tahar [du 9/8/99 au 3/9/99]

Chercheur extérieur

Denis Lugiez [Professeur, Université de Marseille I]

Chercheurs doctorants

Horatiu Cirstea [boursier INRIA, puis ATER à partir du 1/9/99]

Eric Deplagne [allocataire MENESR]

Hubert Dubois [allocataire MENESR]

Laurent Juban [allocataire MENESR, puis ATER à partir du 1/9/99]

Julien Musset [boursier ENS à partir du 1/9/99]

Quang-Huy Nguyen [boursier INRIA à partir du 1/9/99]

Sorin Stratulat [boursier INRIA, co-financé Région Lorraine, puis ATER à partir du 1/9/99]

Stagiaires

Ramzi Azaiez [DEA]

Nozha Chennoufi [4 mois]

Azzouz Ezzaiem [4 mois]

Ahmed Jebali [DEA]

Hassen Sallay [4 mois]

Sana Sfar [4 mois]

2 Présentation et objectifs généraux

L'objectif du projet PROTHEO est la conception et la réalisation d'outils pour la spécification et la vérification de logiciels. Nous utilisons des langages déclaratifs et exécutables à base de règles, de contraintes et de stratégies. Nous développons un environnement de prototypage et des techniques de preuve adaptés pour vérifier des propriétés de ces programmes.

Nos recherches s'appuient sur trois domaines scientifiques : contraintes, réécriture et démonstration automatique. Ces trois thèmes se fertilisent mutuellement dans le projet, car, par exemple, nous utilisons les contraintes et les techniques de réécriture pour améliorer l'efficacité des démonstrateurs et nous formalisons les résolveurs de contraintes et les démonstrateurs à l'aide de règles contrôlées par des stratégies.

En résolution de contraintes, nous nous intéressons à des techniques de propagation de contraintes que nous cherchons à composer avec la programmation entière et la déduction symbolique, ainsi qu'à la collaboration de résolveurs de contraintes se combinant sur des domaines différents ou coopérant entre eux sur un même domaine. Les domaines d'applications privilégiés sont la planification, l'ordonnancement et l'optimisation combinatoire.

Nous développons un environnement de spécification et prototypage fondé sur des règles et des stratégies. Les règles permettent de décrire des calculs, des résolutions de contraintes, des déductions, des transformations de programmes, des transitions d'états, des évolutions d'objets... La programmation par règles se prête bien à l'expression de la concurrence et du non-déterminisme. Afin de traiter le non-déterminisme ou de guider l'application de règles de déduction, des stratégies sont nécessaires. Avoir un langage déclaratif au niveau des stratégies permet de les prototyper facilement et de raisonner sur le contrôle. Le souci d'efficacité nous conduit à proposer des techniques de compilation de la réécriture et des stratégies.

La programmation par règles se prête aussi à la vérification de propriétés. Nous développons des techniques pour prouver certaines propriétés, telles que : le programme termine (pour certaines valeurs), il donne un résultat unique ou des résultats d'un certain type, il est bien défini pour toutes les données possibles, il vérifie une certaine assertion exprimant par exemple sa correction. La spécification et la vérification d'applications liées aux télécommunications est un domaine d'application privilégié.

Nos recherches en démonstration automatique portent d'une part sur les preuves par récurrence et les preuves de propriétés observables, essentielles dans le domaine de la vérification, d'autre part sur l'intégration dans les démonstrateurs de contraintes permettant de restreindre l'espace de recherche et de procédures de décision efficaces sur des domaines interprétés comme l'arithmétique.

Les problématiques de la déduction automatique et de la résolution de contraintes alimentent également des recherches sur la complexité des calculs, les problèmes de décision, les problèmes de comptage des solutions.

Les logiciels développés dans le projet nous servent à valider nos idées, à présenter nos travaux à la communauté scientifique et à transférer nos connaissances vers des domaines d'applications.

3 Fondements scientifiques

3.1 Contraintes

Mots clés : contraintes, résolution, satisfaisabilité, propagation, programmation entière, combinaison, collaboration de solveurs..

Résumé : *Nous étudions la satisfaisabilité et la résolution de systèmes de contraintes, aussi bien sur des domaines symboliques, comme les termes, que sur des domaines numériques, tels que les entiers naturels ou les réels. Nous cherchons à combiner des techniques de propagation de contraintes, de programmation entière, de déduction symbolique et à faire collaborer des résolveurs de contraintes ainsi que la combinaison de telles contraintes. Les procédures que nous obtenons sont fondamentales pour les processus de déduction avec contraintes développés dans le projet.*

La notion de contraintes a montré son intérêt dans la modélisation de problèmes dans divers domaines allant de la mécanique à la logique, en passant par la gestion des activités humaines. Les propriétés à satisfaire sont alors décrites par un ensemble de contraintes dont il importe de déterminer la satisfaisabilité (c'est-à-dire l'existence de solutions) ou l'ensemble des solutions. Si l'on considère, par exemple, la gestion des emplois du temps d'un groupe de personnes, on souhaite savoir dans un premier temps s'il est possible d'ajouter une réunion (problème de la satisfaisabilité) et, dans une seconde étape, obtenir soit une, soit toutes les possibilités d'horaire (c'est-à-dire une ou toutes les solutions).

Dans le cadre des processus de déduction et de la démonstration de théorèmes, apparaissent les contraintes dites symboliques, sur le domaine des termes. Ainsi l'unification, c'est-à-dire la résolution d'équations sur les termes, est à la base de langages de programmation comme Prolog et des démonstrateurs fondés sur la résolution. Le problème se généralise à la résolution d'équations dans des théories équationnelles, par exemple l'unification et le filtrage modulo des symboles ayant des propriétés d'associativité et de commutativité[JK91]. D'autres systèmes de contraintes symboliques utiles dans ce cadre font intervenir des prédicats d'ordre ou d'appartenance, pour ne citer que les plus communs.

Les contraintes numériques jouent un rôle important non seulement en déduction automatique (par exemple, l'unification associative-commutative nécessite de résoudre des équations Diophantiennes linéaires), mais aussi dans de nombreux autres domaines de l'intelligence artificielle ou de la recherche opérationnelle. Un nouveau défi du domaine est de savoir intégrer et combiner des techniques de transformation symbolique pour faire des déductions sur les ensembles de contraintes, avec des méthodes de consistance locale et de propagation de contraintes, et des techniques plus classiques de programmation linéaire en nombres entiers ou de génération de plans de coupe.

Dans ce contexte, nous nous intéressons à la combinaison de contraintes, c'est-à-dire à la résolution de problèmes faisant intervenir des types de contraintes différents. Les outils de

[JK91] J.-P. Jouannaud, C. Kirchner, « Solving equations in abstract algebras: a rule-based survey of unification », *in : Computational Logic. Essays in honor of Alan Robinson*, J.-L. Lassez et G. Plotkin (éditeurs), The MIT press, Cambridge (MA, USA), 1991, ch. 8, p. 257–321.

réécriture et de preuve développés dans le projet sont mis à profit pour spécifier et prouver les procédures de résolution, de propagation et de simplification sur les domaines symboliques et numériques.

3.2 Réécriture et stratégies

Mots clés : réécriture, programmation fonctionnelle, programmation par règles, stratégie..

Résumé : *La réécriture est largement utilisée au sein du projet, d'une part comme technique essentielle dans les démonstrateurs et les résolveurs de contraintes que nous développons, d'autre part comme cadre logique pour spécifier et prototyper les outils que nous proposons. Dans ce type d'applications, la formalisation et l'étude des stratégies jouent un rôle important.*

Les techniques de réécriture ont été développées depuis les années 1970 et appliquées en particulier au prototypage des spécifications formelles algébriques et à la démonstration de propriétés liées à la vérification de programme [20, 11].

A l'origine, le but était de trouver un système de réécriture canonique qui permette de prouver la validité d'un théorème équationnel, en réécrivant chaque membre de l'égalité en un même terme. A partir d'une théorie équationnelle, la procédure de complétion de Knuth et Bendix [KB70] a été conçue pour engendrer, quand cela est possible, un système de réécriture confluent et terminant. Ces deux propriétés assurent la complétude de cette méthode pour décider la validité d'un théorème équationnel. Les techniques de réécriture ont ensuite été appliquées à la preuve par récurrence, à la preuve de cohérence et complétude des spécifications équationnelles ou conditionnelles, à la preuve en logique du premier ordre, à la résolution d'équations dans les théories équationnelles ou conditionnelles, et à des domaines plus spécifiques comme les preuves en géométrie ou les preuves de circuits. Les techniques de réécriture se sont avérées extrêmement utiles en démonstration automatique pour simplifier les espaces de recherche, ou pour inclure des procédures de décision de l'égalité dans des démonstrateurs plus généraux. Les démonstrateurs que nous développons tels SPIKE et daTac utilisent largement ces techniques.

Par ailleurs, la réécriture joue un rôle fondamental dans l'évaluation des langages de programmation fonctionnelle. Dans ce domaine, nos travaux ont porté sur le typage, l'introduction de fonctions partielles, la modularité et la paramétrisation des spécifications. La logique de réécriture a été introduite plus récemment [Mes92] comme une logique dans laquelle la déduction correspond à la réécriture concurrente, c'est-à-dire à l'application en une étape de règles de réécriture à différentes positions disjointes dans le terme. Cette logique fournit aussi un cadre permettant de coder d'autres logiques et a été le point de départ de nos travaux sur le système ELAN.

[KB70] D. E. Knuth, P. B. Bendix, « Simple word problems in universal algebras », *in : Computational Problems in Abstract Algebra*, J. Leech (éditeur), Pergamon Press, Oxford, 1970, p. 263–297.

[Mes92] J. Meseguer, « Conditional rewriting logic as a unified model of concurrency », *TCS 96*, 1, 1992, p. 73–155.

Un autre point commun à l'évaluation des langages fonctionnels et aux démonstrateurs de théorèmes (y compris les assistants de preuves) est l'étude des stratégies. Ces dernières permettent de restreindre l'espace de recherche en sélectionnant certaines branches, de guider les calculs et les déductions en spécifiant quelle règle doit être appliquée à quelle position dans le terme. En programmation fonctionnelle, on peut citer comme exemple la stratégie d'appel par nécessité ou celle d'évaluation paresseuse. En démonstration automatique, il est intéressant de décomposer règles d'inférence et stratégies exprimant le contrôle, car les preuves de correction et de complétude en sont facilitées. Par ailleurs, il est nécessaire d'avoir un langage de stratégies suffisamment puissant pour pouvoir exprimer en particulier l'itération, le raisonnement par cas, les choix déterministes et non déterministes. Nous étudions les stratégies du point de vue de leurs spécifications et de leurs propriétés. Nous les utilisons pour formaliser les preuves dans les démonstrateurs que nous développons.

3.3 Démonstration automatique

Mots clés : déduction, réécriture, récurrence, contraintes, paramodulation, résolution.

Résumé : *Nous développons des méthodes et des systèmes de déduction automatique fondés sur la réécriture et la résolution de contraintes. Ces méthodes sont appliquées aux preuves par induction et aux preuves équationnelles.*

L'élaboration de méthodes et d'outils de vérification de logiciels est l'un de nos objectifs majeurs. Pour le réaliser, nous développons des techniques et des systèmes de déduction automatique fondés sur la réécriture et la résolution de contraintes. La vérification de spécification sur des structures de données récursives fait fréquemment appel à des raisonnements par récurrence, ou à la manipulation d'équations, et exploite des propriétés d'opérateurs comme l'associativité ou la commutativité.

La réécriture, qui permet de simplifier les expressions et les formules, est désormais un ingrédient essentiel pour l'efficacité des systèmes de preuve automatisés. De plus, une relation de réécriture bien fondée peut s'utiliser naturellement pour implanter des raisonnements par récurrence. C'est la base de notre approche dans le système SPIKE qui permet en outre de combiner diverses techniques de simplification et de détecter les conjectures qui sont fausses. Dans le même cadre, nous pouvons coder des preuves de propriétés observables des programmes, qui sont motivées par les spécifications orientées objets.

Les contraintes permettent de différer la résolution de problèmes symboliques complexes pour les résoudre de manière opportuniste. Elles permettent également d'augmenter l'expressivité du langage de spécification et d'affiner les stratégies de preuves. Le traitement des contraintes d'unification ou d'orientation en présence d'opérateurs interprétés (par exemple associatifs-commutatifs) laisse espérer des preuves automatisée radicalement plus courtes. Une implantation de ces idées a d'ailleurs permis à W. McCune [Col96] de résoudre un problème mathématique ouvert. La combinaison des contraintes avec les simplifications par réécriture

[Col96] G. Colata, « With Major Math Proof, Brute Computers Show Flash of Reasoning Power », *The New York Times*, 1996, Tuesday December 10.

pose des problèmes complexes à la fois théoriques, sur la complétude des stratégies, et pratiques, pour une implantation performante. Nous explorons ces techniques d'un point de vue conceptuel mais aussi expérimental, par exemple dans le système daTac.

4 Domaines d'applications

Mots clés : modélisation, prototypage, vérification, logiciels de télécommunication, logiciels de planification.

Les recherches menées dans PROTHEO s'appliquent à la modélisation, au prototypage et à la vérification de composants logiciels. Pour modéliser des systèmes complexes, nous utilisons des langages déclaratifs fondés sur des règles, des contraintes et des stratégies qui permettent de prototyper rapidement une application. Nous offrons des outils de preuve adaptés pour vérifier des propriétés de ces systèmes ou plus précisément de leur modélisation. Les domaines d'application visés sont la spécification et la vérification de logiciels de télécommunication (protocoles et services) dans le cadre d'un contrat avec le CNET (voir module 7.2) et du projet RNRT Calife (voir module 7.3) et les logiciels de planification, comme ROUTEUR (voir module 7.1). A plus long terme, nous voulons appliquer nos techniques à la modélisation de systèmes réactifs et de systèmes physiques hybrides, c'est-à-dire ayant des comportements discrets et continus.

5 Logiciels

5.1 daTac

Participants : Michaël Rusinowitch, Laurent Vigneron.

Le système daTac [Vig94] (*Déduction Automatique dans des Théories Associatives-Commutatives*) est un logiciel de preuve de théorèmes et de complétion dans des théories associatives et commutatives. Les techniques de déduction implantées font intervenir des stratégies de sélection, des étapes de déduction, d'élimination d'informations redondantes et de traitement de contraintes symboliques. daTac a pu démontrer des problèmes assez difficiles (comme le Lemme de SAM) et a été utilisé avec succès pour étudier des algèbres modélisant des logiques non classiques pour la fouille de données.

Initialement développé en Caml Light©, langage fonctionnel de la famille ML, il a été converti cette année en Objective Caml©. Cette transformation, combinée avec des améliorations diverses, a permis de diviser les temps d'exécution par plus de dix, et d'avoir une meilleure gestion de la mémoire.
Le logiciel est muni d'une interface graphique réalisée en TCL/TK (X11 Toolkit) facilitant son utilisation.

daTac [1] est documenté, maintenu, diffusé par ftp et accessible sur le Web. Le correspondant

1. http://www.loria.fr/equipes/protheo/SOFTWARES/DATAC/

[Vig94] L. Vigneron, *Déduction automatique avec contraintes symboliques dans les théories équationnelles*, Th. univ., Université Henri Poincaré - Nancy 1, novembre 1994.

au sein du projet est Laurent Vigneron.

5.2 ELAN

Participants : Horatiu Cirstea, Hubert Dubois, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, Quang-Huy Nguyen, Christophe Ringeissen.

ELAN [BKK+98] est un langage de spécification et de prototypage fondé sur les notions de règles de réécriture et de stratégies qui contrôlent leur application. Il offre un cadre logique simple et naturel pour combiner les paradigmes de calcul et de déduction [33]. Ses originalités principales sont d'implanter des techniques de filtrage et de réduction de termes intégrant des opérations associatives et commutatives, compilées de façon très efficace [13] ; le traitement de calculs non-déterministes, i.e. pouvant retourner plusieurs résultats, dont l'énumération est géré par des stratégies ; un langage de stratégies dont les primitives, en particulier les opérateurs de choix, permettent une gestion fine de l'exploration de l'arbre de recherche,; ; la possibilité donnée à l'utilisateur de définir ses propres stratégies [16]. ELAN a été utilisé pour prototyper des démonstrateurs de théorèmes, des langages logiques de programmation, des résolveurs de contraintes et des procédures de décision. Il offre un cadre modulaire pour étudier leur combinaison.

La version distribuée du système ELAN inclut un interpréteur et un compilateur développés en C++ et Java, une bibliothèque de programmes standard, des exemples d'applications et un manuel d'utilisation. La distribution est exécutable sur les architectures DEC-ALPHA, SUN4 et Intel-PC.

ELAN[2] est documenté, maintenu, diffusé par ftp et accessible sur le Web. Le correspondant au sein du projet est Claude Kirchner.

5.3 SPIKE

Participants : Adel Bouhoula, Gilles Défourneaux, Michaël Rusinowitch, Sorin Stratulat.

Le système SPIKE [BR95] est un démonstrateur automatique de théorèmes dans les théories équationnelles et conditionnelles. SPIKE est développé en Caml Light©, langage fonctionnel de la famille ML. Le logiciel est muni d'une interface graphique réalisée en TCL/TK (X11 Toolkit), qui permet une interaction souple par menus.

Le système SPIKE s'inscrit dans le cadre des outils de vérification de programmes. Ses principales fonctionnalités sont les preuves par récurrence, le test de cohérence des spécifications et le test de complétude des définitions de fonctions. SPIKE dispose d'heuristiques pour la

2. http://www.loria.fr/ELAN/

[BKK+98] P. Borovansky, C. Kirchner, H. Kirchner, P.-E. Moreau, C. Ringeissen, « An Overview of ELAN », *in : Second Workshop on Rewriting Logic and its Applications WRLA'98, Pont-à-Mousson, France*, C. Kirchner, H. Kirchner (éditeurs), *Electronic Notes in Theoretical Computer Science, 15*, Elsevier Science B. V., 1998. URL : http ://www.elsevier.nl/locate/entcs/volume15.html.

[BR95] A. Bouhoula, M. Rusinowitch, « Implicit Induction in Conditional Theories », *Journal of Automated Reasoning 14*, 2, 1995, p. 189–235.

sélection des variables de récurrence et pour la génération automatique de lemmes. Il permet aussi l'interruption d'une preuve et l'ajout de lemmes. Par opposition à la plupart des systèmes actuels de preuve, qui construisent les démonstrations pas à pas et nécessitent de fréquentes interventions de l'utilisateur, voire une forte expertise de la part de celui-ci, SPIKE s'attache à réduire le nombre d'interactions par l'automatisation des tâches routinières.

Un certain nombre de problèmes difficiles ou inaccessibles aux autres systèmes ont pu être démontrés automatiquement par SPIKE (correction de tris, invariants dans le calcul des situations, etc.) ou avec une interaction plus faible qu'avec d'autres systèmes de preuves automatiques comme NQTHM, RRL, LP et PVS (tour de carte de Gilbreath, théorème de Ramsey, théorème de binôme, correction de circuits digitaux, par exemple). SPIKE est également utilisé pour l'enseignement des spécifications formelles algébriques et la vérification de leurs propriétés de complétude et de cohérence, notamment à l'Université Henri Poincaré – Nancy 1 (DESS, ESIAL 3ème année).

SPIKE[3] est documenté, maintenu, diffusé par ftp et accessible sur le Web. Le correspondant au sein du projet est Adel Bouhoula.

6 Résultats nouveaux

6.1 Réécriture et stratégies

Mots clés : Réécriture, stratégie, compilation, analyse syntaxique, règles de production, protocoles d'authentification..

Résumé : *Nous avons étudié les fondements sémantiques d'*ELAN *et de son langage de stratégies, amélioré l'environnement en travaillant sur le compilateur et l'analyseur syntaxique, et modélisé le contrôle de systèmes de règles de production.*

En collaboration avec P. Narendran, Michaël Rusinowitch a édité les actes de la *10th International Conference on Rewriting Techniques and Applications* [36].

6.1.1 Sémantique d'ELAN

Participants : Hélène Kirchner, Claude Kirchner, Christophe Ringeissen.

Dans [16] nous proposons une sémantique purement fonctionnelle pour les règles de réécriture et les stratégies d'ELAN. Le point de départ est de considérer une règle de réécriture comme une fonction qui s'applique à un terme via un opérateur d'application explicite. Des fonctions plus élaborées décrivant d'abord des dérivations puis des ensembles de dérivations peuvent ensuite être construites. Ces fonctions que nous appelons stratégies peuvent à leur tour être définies par des règles de réécriture et la construction peut être itérée pour exprimer des stratégies d'ordre supérieur. De plus, l'opération d'application elle-même est définie par un système de réécriture dont les propriétés sont étudiées. L'implantation de ce calcul en

3. http://www.loria.fr/equipes/protheo/SOFTWARES/SPIKE/

ELAN motive et valide cette approche. L'expressivité d'ELAN est illustrée par des exemples de fonctions et stratégies polymorphes.

Cette vision fonctionnelle des règles et des stratégies se prolonge de façon naturelle par la définition et l'étude du ρ-calcul.

6.1.2 Le ρ-calcul - calcul de réécriture

Participants : Horatiu Cirstea, Claude Kirchner.

Le travail précédent a mis en évidence que le contrôle de la réécriture doit être explicite et peut être décrit naturellement en utilisant la réécriture. Cette observation nous a amenés à un nouveau concept généralisant le λ-calcul et la réécriture que nous avons appelé le ρ-calcul [24].

Les objets de base du ρ-calcul sont construits à partir d'une signature, de l'opérateur d'abstraction "$\rightarrow$" et de l'opérateur d'application "[ ]( )". Le filtrage est utilisé pour lier les variables à leurs valeurs actuelles et est un paramètre fondamental du calcul. Le cas général considère des termes d'ordre supérieur et des théories équationnelles arbitraires, mais comme le filtrage y est en général indécidable, nous nous restreignons actuellement à des théories équationnelles du premier ordre. Un terme du ρ-calcul contient toutes les règles de réécriture nécessaires pour son évaluation. C'est également le cas pour le λ-calcul, mais pas pour la réécriture où le système de règles est implicite. Une autre caractéristique du calcul est la possibilité d'exprimer le non-déterminisme et la partialité. Cette caractéristique est obtenue en utilisant des ensembles de résultats pour la réduction et l'ensemble vide qui représente l'échec de l'application d'une règle. Pour faciliter la spécification et l'exécution des systèmes de réécriture non-déterministes, nous avons ajouté au calcul des opérateurs exprimant la composition des règles et la sélection d'un sous-ensemble de résultats. Ce type d'opérateurs permet à l'utilisateur de définir des stratégies plus élaborées.

Nous avons démontré que le λ-calcul et la réécriture sont des cas particuliers du ρ-calcul, dans le sens où la syntaxe et les règles d'inférence du ρ-calcul peuvent être restreintes afin d'obtenir les deux autres calculs. Nous avons prouvé que le ρ-calcul est confluent sous l'hypothèse que les règles d'inférence du calcul soient guidées par des stratégies appropriées. Nous avons aussi développé le $\rho\sigma$-calcul [24] qui utilise un système de substitution explicite et qui généralise le $\lambda\sigma$-calcul, et prouvé la confluence du calcul avec substitutions explicites.

Ce calcul permet de donner à ELAN une autre sémantique qui a l'avantage de s'étendre à l'ordre supérieur et peut ainsi prendre en compte des extensions du langage.

6.1.3 Compilation de règles de réécriture et de stratégies non-déterministes

Participants : Hélène Kirchner, Pierre-Etienne Moreau.

ELAN est un système qui permet de spécifier et d'exécuter des résolveurs de contraintes, des démonstrateurs et plus généralement tout processus décrit par des règles de transformation. Il possède des opérateurs associatifs-commutatifs (AC) et un langage de stratégies qui permettent une gestion fine de l'exploration d'un arbre de recherche et une manipulation aisée d'opérateurs mathématiques tels que l'union ensembliste, les connecteurs booléens ou les opérations arithmétiques. Ces deux notions améliorent grandement l'expressivité du langage

mais introduisent un double non-déterminisme lié à la possibilité d'appliquer plusieurs règles, de différentes façons, sur un terme donné. Cela rend difficile et généralement peu efficace leur implantation.

Nous avons étudié des techniques de compilation qui améliorent l'efficacité de ce type de langages. Nous avons proposé un nouvel algorithme, à base d'automates déterministes, pour compiler efficacement le filtrage syntaxique. Nous avons défini ensuite différentes classes de règles pour lesquelles nous proposons un algorithme efficace de filtrage. Cet algorithme utilise une structure de données compacte et les automates définis précédemment, ce qui améliore considérablement les performances du processus de normalisation dans son ensemble.

L'étude du langage de stratégies a conduit à implanter des primitives originales de gestion du *backtracking* et à définir un algorithme d'analyse du déterminisme permettant d'améliorer encore les performances, tout en réduisant l'espace mémoire nécessaire. Enfin, l'implantation des méthodes proposées a donné lieu à l'élaboration de nombreuses optimisations théoriques et techniques qui peuvent être largement réutilisées pour implanter d'autres langages de programmation par réécriture. La thèse de Pierre-Etienne Moreau [13] présente les algorithmes et leur évaluation, l'architecture et le fonctionnement du compilateur, ainsi qu'une proposition d'environnement de spécification, fondée sur l'utilisation d'un format de données partagé par les différentes composantes de l'environnement: analyseur syntaxique, interprétation, compilateur.

6.1.4 Analyse syntaxique et format d'échange

Participants : Claude Kirchner, Hélène Kirchner, Christophe Ringeissen.

Dans la perspective d'une nouvelle architecture plus ouverte pour ELAN, nous avons travaillé à un nouveau format d'échange correspondant à une syntaxe abstraite pour les programmes ELAN. Cette syntaxe abstraite, conçue en collaboration avec Mark van den Brand du groupe ASF+SDF (CWI, Amsterdam) s'apparente à celle utilisée dans le nouvel environnement ASF+SDF (Asfix) et à celle choisie pour CASL (Casfix), le langage de spécification algébrique défini par le groupe de travail CoFI[4]. Les syntaxes abstraites AsFix (ASF+SDF), Casfix (CASL) et maintenant Efix pour ELAN sont des instances d'une même structure de termes, les `ATerms`, développée dans le groupe ASF+SDF et diffusée sous la forme d'une librairie (`C` et `Java`).

Nous avons entrepris la réalisation d'un nouvel analyseur syntaxique pour ELAN, en raison des difficultés à maintenir l'analyseur utilisé actuellement, vieux de cinq ans et encore très fortement connecté à l'interpréteur. Nous collaborons avec Mark van den Brand pour réutiliser la technologie d'analyse syntaxique développée au sein du groupe ASF+SDF. Le résultat de ce nouvel analyseur est un terme (plus précisément un `ATerm`) dans le format Efix.

Nous avons réalisé par ailleurs un outil de conversion de l'Efix vers le format REF qui est actuellement le format directement exécutable avec l'interpréteur et le compilateur. Cet outil, écrit en `C` à l'aide de la librairie des `ATerms`, permet déjà d'exécuter une certaine classe de programmes écrits dans la syntaxe abstraite Efix produit par le nouvel analyseur.

4. `http://www.brics.dk/Projects/CoFI/`

6.1.5 ELAN pour CASL

Participants : Hélène Kirchner, Christophe Ringeissen.

Le projet CoFI (Common Framework Initiative for Algebraic Specifications) [Mos97] rassemble une grande partie de la communauté européenne travaillant sur les spécifications algébriques. Il en émane la proposition du langage CASL synthétisant les principales caractéristiques des langages algébriques. CASL se spécialise par restriction en une famille de langages ayant tous une syntaxe et une sémantique consistantes. L'état courant de l'environnement CoFI et des outils supportant le langage CASL[5] a été présenté au workshop WADT'99. Un des objectifs du projet est de réutiliser les outils existants dans la communauté pour exécuter les programmes et vérifier leur propriétés.

Dans ce cadre, nous avons montré comment exécuter une large classe de spécifications CASL en utilisant le système ELAN. Pour cela, nous avons réalisé un outil de conversion du Casfix vers l'Efix, toujours avec la librairie des `ATerms`, qui permet, lorsqu'il est utilisé conjointement à l'outil précédent, de rendre exécutable une spécification équationnelle écrite en CASL grâce à l'utilisation du moteur de réécriture de ELAN (interpréteur ou compilateur). Le système ELAN sera donc très prochainement mis à la disposition de la communauté CoFI comme outil d'exécution d'une certaine classe de programmes CASL [44].

6.1.6 Règles de production et en ELAN

Participants : Hubert Dubois, Hélène Kirchner.

Nous étudions comment modéliser en ELAN des règles de production et des problèmes d'aide à la planification, au travers des concepts de règles, de contraintes et de stratégies.

Dans [41], nous avons mis en évidence la nécessité d'une extension du langage ELAN permettant de définir des règles manipulant des objets structurés et des contraintes, toujours contrôlées à l'aide de stratégies. Cette extension permet de définir des classes d'objets, avec les attributs associés à chacune de ces classes. Les règles manipulant ces objets structurés s'appliquent sur une structure de multi-ensemble d'objets et la font évoluer en modifiant certains objets, en en créant de nouveaux ou en en supprimant d'autres. Cette méthodologie est bien adaptée aux problèmes de planification où des tâches sont décomposées en plusieurs autres jusqu'à arriver à un plan d'actions élémentaires. En nommant les règles, on peut utiliser le langage de stratégies d'ELAN et ainsi guider leur application et éviter une recherche en aveugle.

Dans les problèmes de planification et d'ordonnancement, apparaissent naturellement des contraintes concernant les temps de début et de fin des tâches et actions élémentaires. Pour les prendre en compte, nous avons enrichi la syntaxe des règles manipulant les objets structurés en y ajoutant la possibilité de traiter des contraintes locales à la règle. L'ensemble des contraintes évolue au fur et à mesure de l'application des règles. Les contraintes que nous manipulons sont des égalités et inégalités sur des domaines finis.

5. `http://www.loria.fr/~ hkirchne/CoFI/Tools/`

[Mos97] P. Mosses, « CoFI: The Common Framework Initiative for Algebraic Specification and Development », *in : Proceedings of TAPSOFT '97: Theory and Practice of Software Development, Lecture Notes in Computer Science, 1214*, Springer Verlag, p. 115–137, 1997.

Les stratégies sont encore utilisées à deux autres niveaux : d'une part, pour interagir entre les règles manipulant objets structurés et contraintes et le résolveur de contraintes, afin de déclencher un test de satisfaisabilité ou bien de générer une ou plusieurs solutions ; d'autre part dans le résolveur de contraintes utilisé, COLETTE [Cas98], qui traite la résolution de contraintes à l'aide de règles et de stratégies. Ces travaux ont été présentés à un Workshop ERCIM [28].

6.2 Contraintes

Mots clés : Combinaison, extension de solveur, propagation, programmation entière, optimisation combinatoire..

Résumé : *Nous avons poursuivi les travaux des années précédentes sur la combinaison et l'extension de solveurs. Nous étudions la modélisation par règles et la synthèse de solveurs à base de propagation sur des domaines finis, les techniques de programmation entière et d'optimisation combinatoire pour des problèmes de planification et d'ordonnancement.*

En collaboration avec Hubert Comon, Mehmet Dincbas et Jean-Pierre Jouannaud, Claude Kirchner a publié dans [19] une présentation systématique de la résolution de contraintes.

6.2.1 Combinaison de solveurs

Participant : Christophe Ringeissen.

Nous avons poursuivi notre étude du problème de la satisfaisabilité dans une union (ou mélange) de théories non-nécessairement équationnelles. Il s'agit de déterminer si, étant données une contrainte et une théorie, il existe un modèle de la théorie dans lequel la contrainte puisse être satisfaite. Le cas de théories formées sur des signatures disjointes avait été résolu au début des années 80 par Nelson et Oppen. Nous nous sommes intéressés au cas où les théories partagent des symboles de fonctions ou de prédicats. Nos résultats reposent sur une construction algébrique, appelée fusion, qui permet d'obtenir, sous certaines conditions, un modèle d'un mélange de théories à partir des modèles des théories composant le mélange. Nous nous sommes plus particulièrement intéressé cette année, à la fusion d'algèbres libres ainsi qu'à la fusion d'algèbres initiales. Nous avons ainsi fait une étude comparative de notre algorithme de combinaison avec celui conçu par F. Baader et K. Schulz pour le problème de disunification dans une amalgamation libre d'algèbres libres, une construction algébrique plus complexe que notre notion de fusion et qui n'est valable que pour des signatures disjointes. Nous avons également montré que notre algorithme de combinaison peut être appliqué pour résoudre des problèmes de filtrage lorsque l'on considère une fusion d'algèbres initiales.

L'ensemble de ces résultats, obtenus en collaboration avec Cesare Tinelli (Université de l'Illinois à Urbana-Champaign puis Université de Iowa), fait l'objet d'un article soumis à publication.

[Cas98] C. Castro, *Une approche déductive de la résolution de problèmes de satisfaction de contraintes*, Thèse d'université, LORIA, 1998.

6.2.2 Extension de solveurs

Participant : Christophe Ringeissen.

Dans [21], en collaboration avec Eric Monfroy (CWI, Amsterdam), nous présentons un cadre général pour étendre un solveur en lui permettant de traiter des contraintes hétérogènes faisant intervenir de nouveaux symboles de fonctions. Solex implante actuellement un ensemble de règles de transformation qui effectuent un pré-traitement et un post-traitement des formules, afin de rendre admissibles pour un solveur des contraintes qu'il ne pouvait pas traiter auparavant. Les règles agissent par ajout d'informations relatives aux nouvelles fonctions, par complément du domaine de calcul et par abstraction. Chacune de ces règles peut être vue comme un solveur élémentaire, et Solex comme une collaboration de solveurs. Il est donc possible de rendre ce schéma directement exécutable grâce au système Bali développé par Eric Monfroy durant sa thèse pour décrire et réaliser des collaborations de solveurs.

6.2.3 Règles de propagation de contraintes pour les domaines finis

Participant : Christophe Ringeissen.

Les algorithmes de résolution de contraintes sont de nos jours souvent fondés sur des mécanismes de propagation de contraintes, qu'on peut voir de façon abstraite comme une forme de déduction s'exprimant par des règles de transformation de contraintes. On peut ainsi imaginer mettre en œuvre de la propagation de contraintes simplement par l'application de règles dans un langage adapté. Nous avons travaillé avec Eric Monfroy (CWI, Amsterdam) dans le prolongement d'une étude commencée par Apt et Monfroy sur la création automatique de solveurs à base de règles pour la réduction de problèmes de satisfaction de contraintes. Nous avons amélioré le schéma général de production de règles grâce à l'utilisation d'un algorithme d'unification dans les algèbres finies et de la structure associée de graphes acycliques dirigés n-aires. Cette approche permet de traiter une nouvelle forme de règles de propagation avec paramètres, d'avoir une méthode plus déductive dans la construction de chaque règle, et de mieux contrôler la production de l'ensemble des règles.

6.2.4 Techniques de programmation entière

Participant : Alexander Bockmayr.

Programmation entière pour la planification Une approche récente pour résoudre des problèmes de planification en intelligence artificielle consiste à les traduire en problèmes de satisfaisabilité en logique propositionnelle. Bien que cette méthode ait des succès remarquables dans de nombreux domaines de planification, elle ne permet pas d'exprimer facilement des contraintes numériques, ou d'optimiser par rapport à une fonction objectif complexe. De plus, la représentation en logique propositionnelle impose des restrictions sur les connaissances du domaine qui peuvent être formulées.

Pour aborder ces difficultés, nous avons proposé dans [22] de passer de la logique propositionnelle à un langage de représentation plus riche qui est la programmation linéaire en

nombres entiers. Cela nous a permis de modéliser facilement des contraintes de capacité. Plus généralement, des connaissances riches sur le domaine peuvent être représentées de manière compacte et exploitées par les solveurs de programmation entière. Finalement la relaxation linéaire donne lieu à une nouvelle stratégie de recherche qui a permis de résoudre de manière approximative des problèmes de planification qui ne sont pas à la portée des méthodes exactes.

Théorie des plans de coupe en programmation entière La procédure de Gomory-Chvátal est une méthode fondée sur des plans de coupe qui permet de calculer l'enveloppe convexe P_I des points entiers contenus dans un polyèdre P. Le nombre d'itérations nécessaires pour obtenir P_I est appelé le *rang de Chvátal* de P. Cette notion fut introduite par Chvátal en 1973. Elle donne une mesure pour la difficulté de résoudre un programme linéaire en nombres entiers sur P. Le rang de Chvátal est toujours fini, mais il n'existe pas de borne supérieure pour des polyèdres quelconques même en dimension 2. Dans [15], nous avons démontré que le rang de Chvátal d'un polyèdre P contenu dans le cube $[0,1]^n$ est de l'ordre $O(n^3 \log n)$.

6.2.5 Applications à la résolution des problèmes combinatoires

Participants : Jianyang Zhou, Jiany Zhou.

Dans le cadre d'un projet de création d'entreprise, nous avons mené une étude sur des problèmes combinatoires, de planification et d'ordonnancement : planification d'interviews, planification d'itinéraires, ordonnancement d'atelier, et planification de média. Des modèles mathématiques de ces problèmes ont été établis et comparés, des expérimentations ont été menées pour les évaluer.

Nous avons utilisé le langage de contraintes NCL pour programmer deux applications particulières: une planification d'interviews et une planification de média.

- Nous avons développé un modèle « timetabling » pour planifier les interviews entreprises/étudiants pour un forum franco-allemand qui s'est tenu à Metz. Ce programme résout en 10 minutes le cas de 48 entreprises et 468 étudiants.
- Nous avons effectué une étude sur le problème de détermination des gagnants dans des enchères combinatoires. Nous avons testé un ensemble d'heuristiques pour prétraiter les offres afin de simplifier le problème. Des solutions aux problèmes comportant 200 articles et 340 acheteurs (170 articles et 273 acheteurs après simplification) peuvent être trouvées en quelques secondes. Après 14 heures de calcul, la solution optimale est trouvée sur un Pentium 300.

6.3 Déduction automatique

Mots clés : Déduction modulo, procédures de décision, théories contraintes..

Résumé : *Nous travaillons sur l'intégration de la déduction (au premier ordre ou à l'ordre supérieur) avec des calculs, des procédures de décision, des résolutions de contraintes.*

6.3.1 Déduction modulo

Participants : Eric Deplagne, Claude Kirchner.

En collaboration avec Gilles Dowek (projet COQ) et Thérèse Hardin (LIP6 et projet Para), nous avons introduit en 98 une nouvelle présentation de la logique du premier ordre appelée déduction modulo qui met en valeur l'aspect déduction modulo une congruence. L'une des originalités de l'approche est que cette congruence est définie non seulement sur les termes mais aussi sur les propositions.

Dans ce cadre, nous avons défini un calcul des séquents modulo une telle congruence. Il est souvent naturel d'interpréter cette congruence par un système de réécriture équationnel confluent. A une telle présentation de la logique du premier ordre, on peut associer une méthode de preuve appelée ENAR et fondée sur une extension de la résolution avec contrainte avec une règle de surréduction appropriée. Nous avons montré que cette méthode de preuve est correcte et complète à partir du moment où l'on sait éliminer les coupures dans le calcul des séquents modulo. Des conditions suffisantes sur la congruence ont été données par Gilles Dowek et Benjamin Werner pour assurer l'élimination de coupures. Ces résultats sont en cours de publication.

Parmi les théories considérées, la présentation au premier ordre de la logique d'ordre supérieur en utilisant le calcul des substitutions explicites satisfait l'élimination des coupures. Cette présentation est intentionnellement équivalente à la présentation traditionnelle de la logique d'ordre supérieur utilisant le λ-calcul, au sens où une proposition peut être démontrée sans les axiomes d'extensionnalité dans un système si et seulement si elle peut l'être dans l'autre. En appliquant la méthode de preuve ENAR, nous avons pu obtenir une simulation pas à pas de la résolution d'ordre supérieur. Ainsi, exprimer la logique d'ordre supérieur comme une théorie du premier ordre et appliquer une méthode de recherche de démonstration au premier ordre est au moins aussi efficace qu'une implantation dédiée. De plus rester dans le formalisme de la logique du premier ordre permet de bénéficier des optimisations connues dans ce cadre. Enfin, cela a permis d'étendre simplement la méthode à la résolution équationnelle d'ordre supérieur. Ces résultats ont été publiés à RTA'99 [27].

Par ailleurs, une reformulation des travaux de Patrick Viry sur l'expression du calcul des séquents standard dans le formalisme de la déduction modulo a conduit Eric Deplagne à donner une présentation complète du calcul des séquents standard maximisant le système confluent modulo lequel la déduction est appliquée.

6.3.2 Spécifications algébriques, types d'ordre supérieur et modèles ensemblistes

Participant : Hélène Kirchner.

Dans les spécifications formelles de type algébriques, le système de types est restreint à des sortes, sous-sortes et à des types fonctionnels du premier ordre. Ce n'est pas le cas des spécifications fondées sur les modèles qui autorisent des types d'ordres supérieurs interprétés de façon ensembliste comme des produits cartésiens, des espaces fonctionnels et des parties d'ensembles.

Dans des travaux précédents, nous avions proposé d'enrichir les spécifications algébriques

avec des types d'ordre supérieur, tout en restant dans le cadre de la logique des clauses de Horn avec égalité. Ce travail avec Peter Mosses (BRICS, Université de Aarhus) a été poursuivi pour simplifier notre première approche. Nous avons fait le lien avec les modèles ensemblistes classiques et travaillé sur l'existence de modèles initiaux dans différentes classes de modèles [32].

6.3.3 Déduction avec contraintes et simplification dans les théories équationnelles

Participants : Claude Kirchner, Christelle Scharff.

Dans le cadre des méthodes de recherche de preuve et de déduction en logique du premier ordre avec égalité, le souci de réduire l'espace de recherche et la volonté de résoudre de plus en plus de problèmes ont suggéré l'utilisation du parallélisme, de contraintes et de stratégies de contraction. L'objectif de la thèse de Christelle Scharff [14] était l'exploration de ces trois voies et de leurs interactions dans le cadre d'une procédure de complétion.

Nous avons proposé une nouvelle procédure de complétion close de grain fin fondée sur l'utilisation des graphes SOUR. Chaque nœud du graphe est un processus représentant un terme et les arcs sont des canaux de communication. La complétion est réalisée de façon complètement asynchrone par coopération entre les processus sans mémoire globale ni contrôle global et utilise une stratégie de contraction, ce qui est l'un des principaux avantages. Notre méthode est implantée en C et fait appel à PVM.

La complétion basique est rarement complète en combinaison avec des stratégies de contraction. Elle était connue complète seulement pour une forme de simplification particulière dite basique. Nous avons développé une nouvelle stratégie de simplification, la simplification E-cycle, fondée sur l'utilisation d'un graphe schématisant les dépendances entre égalités. Elle permet plus de simplifications et est plus simple à comprendre que la simplification basique. Elle a été implantée en ELAN.

En pratique, la simplification basique n'est pas efficace pour la complétion basique modulo. Nous avons proposé un nouveau système de règles d'inférence qui ne requiert pas de résoudre les contraintes et réduit l'écart entre la théorie et la pratique. Pour la complétude de notre système, il est nécessaire de faire des inférences dans les contraintes, ce qui est original par rapport aux approches actuelles de la recherche de preuve et de la déduction avec contraintes.

6.3.4 Preuve automatique par récurrence sur des théories contraintes

Participants : Adel Bouhoula, Florent Jacquemard.

Les structures de données telles qu'ensembles, listes ordonnées, *powerlists* peuvent être spécifiées par des théories équationnelles avec contraintes. Peu de travaux ont été consacrés à l'automatisation des preuves par récurrence dans de telles théories, car elles engendrent des schémas de récurrence très complexes. Nous avons proposé une nouvelle approche fondée sur la récurrence par ensembles test sur laquelle est bâti par exemple le démonstrateur SPIKE [BR95], et sur les automates d'arbres à contraintes.

[BR95] A. Bouhoula, M. Rusinowitch, « Implicit Induction in Conditional Theories », *Journal of Automated Reasoning 14*, 2, 1995, p. 189–235.

L'idée principale est la construction d'un automate d'arbres à contraintes qui caractérise le modèle initial de la spécification considérée. Nous avons généralisé cette construction, que l'on trouve dans la littérature pour un système de réécriture quelconque, aux systèmes de réécriture avec contraintes. L'automate obtenu est ensuite utilisé comme moteur de récurrence au cours d'une preuve, pour générer de nouveaux sous-buts lors d'une étape de récurrence. Ces sous-buts sont alors simplifiés, puis à leur tour prouvés par récurrence, ou invalidés dans le cas où un défaut de cohérence est détecté. Ce test se ramène à des procédures de décision du vide pour les automates d'arbres à contraintes.

Nous avons prouvé que cette procédure est correcte et réfutationellement complète, même en présence d'un système d'axiomes constructeurs contraints, pas nécessairement linéaire gauche, éventuellement non-terminant, mais confluent. Les axiomes non-constructeurs peuvent être quant à eux des équations conditionnelles et contraintes.

6.3.5 Intégration de procédures de décision

Participants : Alessandro Armando, Gilles Défourneaux, Michaël Rusinowitch, Sorin Stratulat.

Nous avons poursuivi notre étude sur l'intégration des procédures de décision dans un démonstrateur par réécriture du type de SPIKE. Nous avons défini des schémas de règles d'inférences assurant la correction du démonstrateur [39]. Il s'agit maintenant de terminer un prototype intégrant l'arithmétique linéaire et de le valider sur des exemples.

6.3.6 Clôture par congruence

Participant : Laurent Vigneron.

Des procédures de décision efficaces fondées sur la normalisation par réécriture sont indispensables pour développer des environnements de preuve. Les algorithmes de clôture par congruence s'appliquent dans ce cadre et permettent de construire une représentation compacte de la théorie équationnelle. En collaboration avec L. Bachmair, I. V. Ramakrishnan et A. Tiwari (Université de Stony Brook, NY), nous avons défini une notion de clôture abstraite par congruence. Ces travaux permettent de mieux comprendre de nombreux résultats dans ce domaine et de voir les liens entre eux (Nelson et Oppen, Downey, Sethi et Tarjan, Shostak). Une première partie de ces résultats a été présentée à la conférence RTA'99 [BRRT99].
Ces travaux ont été étendus pour permettre de considérer des opérateurs associatifs-commutatifs. La méthode utilisée est fondée sur la combinaison d'algorithmes de complétion pour des théories portant sur des signatures disjointes, afin d'engendrer un système convergent sur une signature étendue. Cette approche peut également être utilisée pour résoudre le problème du mot pour des théories AC closes, sans utiliser d'ordre de simplification AC.

[BRRT99] L. Bachmair, C. R. Ramakrishnan, I. V. Ramakrishnan, A. Tiwari, « Normalization via Rewrite Closures », *in : Proceedings of the 10th International Conference on Rewriting Techniques and Applications*, P. Narendran, M. Rusinowitch (éditeurs), Springer-Verlag, p. 190–204, Trento (Italy), juillet 1999.

Un article est actuellement soumis à une conférence, et un second en préparation pour un journal.

6.3.7 Preuves dans les ensembles approximants

Participant : Laurent Vigneron.

La collaboration avec Anita Wasilewska (Université de Stony Brook, NY) s'est poursuivie cette année pour démontrer des propriétés sur les ensembles approximants (rough sets) utilisés pour raisonner dans des logiques non classiques. En particulier, il a été mis en évidence qu'utiliser un démonstrateur comme daTac est indispensable, mais qu'il doit être combiné avec une démarche plus intuitive. C'est ce qui a été fait grâce à des diagrammes, appelés *diagrammes approximants* par analogie aux diagrammes de Venn. Ils ont permis de détecter les propriétés importantes à démontrer, mais ils ont aussi été utilisés pour trouver des contre-exemples. Un article décrivant cette approche automatique et intuitive a été présenté à la conférence NAFIPS-99 [38]. Il montre comment les diagrammes pourraient être utilisés comme interface entre l'utilisateur et le démonstrateur.

6.4 Preuve de propriétés de programmes

Mots clés : Preuve par récurrence, preuve équationnelle, procédure de complétion, terminaison, propriété observable.

Résumé : *Nous avons développé de nouvelles techniques de preuves de propriétés dans les théories équationnelles (terminaison, confluence, complétude), pour des propriétés observables, et des systèmes réactifs.*

6.4.1 Preuves de terminaison par induction

Participants : Isabelle Gnaedig, Hélène Kirchner.

Nous avons abordé le problème des preuves de terminaison de la réécriture avec une approche nouvelle utilisant l'induction explicite. Sur l'algèbre des termes clos, l'évaluation d'un terme t termine (brièvement t termine) si toutes les chaînes de réécriture partant de ce terme sont finies. L'idée de preuve, très simple, consiste à démontrer, dans le cas de la réécriture innermost, que tout terme termine si ses sous-termes terminent, en utilisant le fait que tout terme plus petit que le terme initial, pour un ordre noethérien, termine par hypothèse d'induction. Une autre nouveauté de notre approche est que l'ordre utilisé dans l'induction noethérienne n'est pas déterminé a priori, mais précisé au cours de la preuve par des contraintes qu'il doit satisfaire.

La méthode proposée utilise un mécanisme fondé sur deux étapes de base : une étape d'abstraction par des variables représentant les formes normales des sous-termes, et une étape de surréduction schématisant les étapes de réécriture innermost sur les termes clos. Toutes deux imposent de nouvelles contraintes sur l'ordre utilisé. Ce processus est itéré jusqu'à ce que les contraintes sur l'ordre d'induction soient insatisfaisables ou qu'on ne puisse plus surréduire.

Quand la procédure termine avec des contraintes satisfaisables, la preuve de terminaison est établie. Un ensemble de règles d'inférence décrivant ce mécanisme a été proposé, ainsi qu'une stratégie d'application. Nous avons prouvé la correction de l'ensemble vis-à-vis de la propriété de terminaison sur les termes clos [42].

Cette méthode, grâce à la force de l'induction, permet de prouver avec des ordres de simplification la terminaison de systèmes non simplifiants, donc récalcitrants pour les ordres de simplification utilisés avec l'approche classique.

6.4.2 Vérification simultanée de la complétude et de la confluence sur les termes clos

Participant : Adel Bouhoula.

Les méthodes existantes pour tester la confluence sur les termes clos utilisent des techniques de complétion ou bien vérifient que toutes les paires critiques entre les axiomes sont valides par rapport à un critère de confluence close. Ces techniques ne sont pas efficaces même si le nombre d'axiomes de la spécification est très réduit. En effet, les procédures de complétion divergent souvent et il existe en général un nombre très important de paires critiques entre les axiomes.

Nous avons développé une nouvelle procédure pour tester simultanément la complétude et la confluence sur les termes clos lorsque les fonctions sont définies à l'aide de constructeurs libres, lorsqu'il y a des relations entre les constructeurs et lorsque la spécification est paramétrée. Dans le cas où la spécification n'est pas complète ou bien n'est pas confluente sur les termes clos, notre procédure retourne l'ensemble des motifs pour lesquels les fonctions ne sont pas définies. Elle permet également d'identifier les règles qui empêchent la confluence.

Notre procédure est complète et termine toujours sous l'hypothèse qu'un oracle permet de décider des propriétés inductives. Par opposition aux techniques précédentes, on n'utilise pas de procédures de complétion et on n'a pas besoin de calculer l'ensemble des paires critiques des axiomes.

La méthode a été implanté dans le système SPIKE. Elle a permis de tester la complétude et la confluence sur les termes clos de plusieurs spécifications d'une manière totalement automatique alors que les autres techniques divergent ou engendrent des preuves très complexes.

6.4.3 Preuves de propriétés observables

Participants : Adel Bouhoula, Ahmed Jebali, Michaël Rusinowitch.

Les preuves de propriétés observables sont bien adaptées aux raisonnements sur les systèmes orientés objets où les états d'un objet sont observés en appliquant certaines méthodes à des attributs. Nous avons expérimenté notre technique de preuve observationnelle par contextes critiques sur des équivalences de comportements et sur la validation de raffinements de spécifications [23]. Les résultats ont montré des possibilités d'augmenter l'automatisation de ce type de preuves [43]. L'approche semble donc offrir une alternative prometteuse à la coinduction.

6.5 Vérification

Mots clés : Vérification de programmes, protocoles d'authentification, protocoles de télécommunication, systèmes réactifs..

Résumé : *Nous avons appliqué les techniques développées dans l'équipe à la certification de protocoles et à la vérification de systèmes réactifs.*

6.5.1 Analyse des protocoles d'authentification en ELAN

Participants : Horatiu Cirstea, Claude Kirchner.

La programmation avec règles de réécriture et stratégies a été utilisée pour décrire de nombreux processus de transitions. Dans [25], nous avons étudié comment spécifier et analyser en ELAN des protocoles d'authentification. Le protocole de Needham-Schroeder à clé publique a été utilisé pour réaliser une authentification mutuelle entre un initiateur et un répondeur qui communiquent par l'intermédiaire d'un réseau non sécurisé. Le comportement des agents et des intrus ainsi que les invariants de sécurité que le protocole doit vérifier sont naturellement décrits par des règles de réécriture conditionnelles. Leur application est contrôlée par des stratégies. Des attaques similaires à celles déjà décrites dans la littérature ont été redécouvertes. De plus, en rendant la formalisation exécutable, nous permettons l'utilisation de la spécification pour analyser le protocole ou pour rejouer des attaques ou des scénarios proposés par un tiers. Nous avons montré comment différentes stratégies utilisant le même ensemble de règles de réécriture peuvent améliorer l'efficacité de la détection des attaques. Nous avons comparé nos résultats aux approches existantes et nous avons obtenu des performances souvent meilleures que celles d'outils dédiés à la vérification des protocoles, en particulier Murphi.

6.5.2 Vérification de protocoles d'authentification avec daTac

Participants : Florent Jacquemard, Michaël Rusinowitch, Laurent Vigneron.

Nous avons proposé une sémantique opérationnelle pour les protocoles d'authentification fondée sur la surréduction associative et commutative. Plus précisément, nous avons conçu un algorithme qui compile une description standard d'un protocole en un système de réécriture qui peut ainsi servir à la simulation des exécutions du protocole à partir d'états initiaux. Cela permet d'une part de définir formellement la sémantique opérationnelle des protocoles et par exemple de vérifier qu'ils sont implantables. D'autre part, l'algorithme de compilation étant implanté en OCAML, les formules résultantes peuvent être envoyées presque telles quelles à des systèmes de preuves automatiques. Nous utilisons le démonstrateur daTac sur le résultat car il manipule une logique équationnelle bien adaptée. daTac cherche à résoudre un but qui représente l'existence d'une attaque. Nous pouvons également compiler les buts et le comportement de l'intrus.

Les expériences sur des protocoles classiques montrent que la méthode permet de retrouver des attaques connues. Par rapport à d'autres outils existants comme CASPER, nous ne sommes pas limités par la vérification finie du model checking et, grâce à l'unification, nous traitons

plus facilement la génération de nombres aléatoires (nonces et clés de session), la fraîcheur et les aspects temporels.

6.5.3 Vérification de protocoles ABR

Participants : Michaël Rusinowitch, Sorin Stratulat.

Le protocole ABR pour les réseaux ATM est bien adapté au transport de données car il assure un débit minimum et limite les pertes. Le protocole repose sur un contrat entre l'opérateur qui fournit le débit minimum et une application qui respecte le débit qui lui est alloué de manière dynamique, selon les ressources disponibles du réseau. Le Forum ATM a défini un algorithme pour contrôler la conformité du débit de la source. C. Rabadan et F. Klay (France Télécom CNET) ont proposé une version incrémentale plus efficace de cet algorithme. Nous avons obtenu une preuve automatique complète de l'équivalence entre les deux algorithmes en utilisant le système PVS. La preuve nécessite de nombreux lemmes et une analyse par cas complexe [37]. Des protocoles de contrôle de conformité ABR ont été vérifiés récemment par d'autres auteurs. Mais les protocoles concernés étaient seulement des approximations de celui que nous traitons.

6.5.4 Preuve de systèmes réactifs

Participants : Ramzi Azaiez, Adel Bouhoula, Julien Musset, Michaël Rusinowitch.

Nous avons étudié comment démontrer des propriétés de programmes écrits dans le langage synchrone LUSTRE à l'aide du système SPIKE. Dans un premier temps, nous avons proposé une traduction d'un programme LUSTRE vers une spécification utilisable par SPIKE ; un logiciel écrit en Caml permet de réaliser automatiquement cette tâche [40]. Nous avons ensuite étudié des exemples de programmes LUSTRE afin de valider la traduction proposée et d'améliorer les stratégies de preuves utilisées par SPIKE [45].

6.6 Complexité

Mots clés : Complexité du calcul, complexité de décision et de comptage, coNP-complet, PSPACE-complet, indécidabilité..

Résumé : *Nous avons obtenu des résultats sur le comptage et la reconnaissance de la base de Hilbert, la complexité de certains problèmes de filtrage associatif-commutatif et de l'accessibilité rigide.*

6.6.1 Base de Hilbert

Participants : Arnaud Durand, Nicolas Hermann, Laurent Juban.

Les travaux décrits dans cette section sont présentés dans la thèse de Laurent Juban [12].

Comptage de la base de Hilbert En collaboration avec Phokion G. Kolaitis (Université de Californie, Santa Cruz), nous avons étudié le problème de comptage de la base de Hilbert d'un système d'équations diophantiennes linéaires. La résolution de ce problème est importante pour déterminer la complexité de l'énumération de toutes les solutions des problèmes d'unification équationnelle dans les théories comme les semigroupes commutatifs et les monoïdes commutatifs, ainsi qu'en recherche opérationnelle, dans la théorie des variétés toriques, etc.

Nous avons obtenu des résultats pour le cas général, ainsi que pour des cas particuliers du problème de comptage de la base de Hilbert, qui expliquent la mauvaise performance des logiciels travaillant sur les théories équationnelles mentionnées. En particulier, le problème de calculer la cardinalité de la base de Hilbert d'un système diophantien linéaire homogène a été prouvé #P-difficile, donc intractable, et en même temps appartenant à la classe #NP. Étant donné que les classes de comptage ne sont pas closes par rapport à la réduction de Turing, utilisée dans le cadre de la complexité de comptage, ce résultat implique que ce problème est #NP-complet sous les réductions de Turing. Si le nombre d'occurrences de chaque variable est restreint à trois, la complexité du nouveau problème est équivalente au problème général. Par contre, si le nombre d'occurrences de chaque variable est restreint à deux, le problème entre dans la classe #P, mais en même temps la borne inférieure non-triviale est perdue. Or, si chaque variable n'a qu'une seule occurrence, le problème de comptage devient polynomial. De plus, même si la base de Hilbert est connue, le problème de calculer la cardinalité de l'ensemble minimal et complet d'unificateurs associatifs-commutatifs sous-jacents est #P-complet. Tous ces résultats ont été présentés lors de la conférence internationale LPAR'99 [30].

Circonscription propositionnelle Afin de déterminer les bornes supérieures des problèmes de comptage de la base de Hilbert, il est important d'étudier le problème de la circonscription propositionnelle qui permet de tester la minimalité d'une solution pour une formule propositionnelle. Nous avons obtenu une caractérisation syntaxique des relations logiques par rapport à la complexité du problème de circonscription, ainsi qu'une caractérisation syntaxique complète des relations logiques pour le problème de l'existence d'une solution unique. Ce dernier résultat constitue un théorème dichotomique pour le problème de satisfaisabilité unique d'une formule propositionnelle [31].

Reconnaissance de la base de Hilbert Le problème de reconnaissance de la base de Hilbert est étroitement lié à celui du comptage. La question est de savoir si un vecteur appartient à la base de Hilbert d'un système donné. C'est en général un problème coNP-complet. Nous avons prouvé que ce problème devient polynomial si l'entrée est donnée en notation unaire et que le nombre d'équations du système est borné par une constante. Par contre, il devient coNP-complet au sens fort, c.-à-d. coNP-complet même si l'entrée est en notation unaire, lorsque le nombre d'équations du système n'est pas borné.

De plus, la question de savoir si un ensemble de vecteurs constitue la base de Hilbert d'un système donné ou inconnu est un problème coNP-complet au sens fort. Ces résultats, présentés à la conférence internationale MFCS'99 [29], répondent à des problèmes ouverts posés respectivement par Henk et Weismantel en 1996, et par Edmonds et Giles en 1982.

6.6.2 Filtrage associatif-commutatif élémentaire simultané

Participant : Nicolas Hermann.

Les problèmes d'unification et de filtrage les plus étudiés en déduction automatique sont ceux des semigroupes commutatifs dont la théorie équationnelle est engendrée par les axiomes d'associativité et commutativité. D'où l'intérêt de connaître la frontière entre les cas tractables et intractables pour les problèmes de décision et de comptage en filtrage associatif-commutatif. En collaboration avec Phokion G. Kolaitis (Université de Californie, Santa Cruz), nous avons étudié les problèmes élémentaires simultanés, i.e. des conjonction de problèmes dont les termes sont formés d'un seul symbole associatif-commutatif et de constantes.

Dans un premier temps, nous avons regardé le cas où les problèmes contiennent au plus k équations et au plus m constantes. Si les paramètres k et m sont bornés par une constante, les problèmes de décision et de comptage sont polynomiaux, sinon le problème de décision est NP-complet et le problème de comptage est #P-complet.

Dans un deuxième temps, nous avons étudié le cas de problèmes qui contiennent au plus k équations, m constantes et où chaque variable apparaît au plus i-fois. Pour $i = 1$, ce sont les problèmes de filtrage AC sur les termes linéaires. Pour le problème de décision avec $i = 2$, deux constantes et un nombre non-borné d'équations, il existe un algorithme polynomial. Cet algorithme fait appel à l'algorithme de *b-matching* sur les multigraphes. Tous ces résultats ont été publiés dans l'article [18].

6.6.3 Complexité et décidabilité de l'accessibilité rigide

Participant : Florent Jacquemard.

En collaboration avec Harald Ganzinger et Margus Veanes (MPI, Saarbrücken), nous avons poursuivi l'étude de l'accessibilité rigide. Cette propriété, introduite dans un article des actes de la conférence ASIAN'98, généralise l'unification rigide au cas orienté de la réécriture. L'unification rigide a été introduite par J.H. Gallier, S. Raatz et W. Snyder (1987) dans le contexte de l'extension des méthodes de tableaux à la démonstration automatique en logique équationnelle. Si l'accessibilité rigide est indécidable en général, contrairement à l'unification rigide, il est possible de caractériser certains cas décidables et d'en donner la complexité. Dans [26], nous montrons que l'accessibilité rigide est PSPACE-complète dans le cas des signatures monadiques (tous les symboles ont une arité au plus 1), quand toutes les règles de réécriture considérées sont closes. Cela constituait dans le cas particulier de l'unification rigide un problème ouvert par Gurevich et Voronkov. Nous avons caractérisé dans [26] un fragment non-monadique décidable qui généralise ceux connus précédemment. Un article rassemblant les résultats de [26] a été accepté pour publication par le journal IJFCS [17].

Le problème essentiel restant ouvert est celui de la décidabilité de l'unification rigide dans le cas des signatures monadiques avec des règles de réécriture arbitraires, vers lequel on réduit facilement la résolution d'équations de Makanin.

6.6.4 Langages de motifs

Participants : Gregory Kucherov, Michael Rusinowitch.

Grégory Kucherov et Michael Rusinowitch ont écrit un survol sur des problèmes qui concernent les langages de motifs (arbres et mots) : recherche d'un sous-motif, inclusion de langages ... Ils ont montré qu'il n'existe pas d'algorithme pour décider si le complément d'un ensemble de motifs de mots est fini [35].

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Planification de transport porte-à-porte à la demande pour le GIHP

Participants : Eric Domenjoud, Claude Kirchner, Jianyang Zhou, Alexander Bockmayr.

Mots clés : propagation de contraintes, planification.

Résumé : *Dans le cadre d'un contrat avec le GIHP-Champagne, nous avons réalisé un moteur de calcul fondé sur la propagation de contraintes pour la planification d'un service de transport porte-à-porte à la demande.*

Le transport porte-à-porte à la demande est un service offert aux usagers qui indiquent un lieu de départ, un lieu d'arrivée et un horaire souhaité. Il s'agit alors d'affecter à chaque demande un véhicule et un chauffeur de manière à optimiser le coût du transport tout en garantissant la qualité du service et en respectant les horaires demandés ainsi que certaines contraintes liées aux personnes transportées. Un exemple de ce type de service est le transport des personnes handicapées par le Groupement pour l'Insertion des personnes Handicapées Physiques (GIHP).

Nous avons décrit pour le compte du GIHP-Champagne une modélisation complète du problème. Cette modélisation a été implantée dans un prototype de moteur de calcul qui résout le problème par propagation de contraintes et consistance locale avec des heuristiques pour l'optimisation locale. Ce prototype utilise la bibliothèque *NCL Scheduler* de contraintes sur des domaines finis réalisée durant la thèse de Jianyang Zhou [Jia97]. Au cours de l'année 1999, ce moteur de calcul a été intégré dans les interfaces développées par le GIHP-Champagne. De notre côté, nous avons optimisé le prototype initial afin d'augmenter sa capacité de traitement et nous l'avons étendu afin de prendre en compte de nouveaux types de prestations. La version actuelle effectue la planification de 300 transports en 2 minutes.

[Jia97] Z. Jianyang, *Calcul de plus petits produits cartésiens d'intervalles*, thèse de doctorat, Université d'Aix-Marseille II, LIM, 163, Avenue de Luminy, 13288 MARSEILLE, FRANCE, March 1997.

7.2 Vérification de services de télécommunications pour le CNET

Participants : Adel Bouhoula, Michaël Rusinowitch, Sorin Stratulat.

Mots clés : vérification, télécommunications..

Résumé : *Dans le cadre d'un contrat CTI avec le CNET, nous explorons par des études de cas l'applicabilité du démonstrateur* SPIKE *à la vérification de services de télécommunications.*

Dans [34], nous avons proposé une méthodologie en sept étapes pour spécifier, analyser et vérifier des interactions des services téléphoniques avec des systèmes de déduction, en particulier avec SPIKE. Elle permet la détection des interactions observables par l'utilisateur et aide à leur résolution.

7.3 CALIFE

Participants : Claude Kirchner, Hélène Kirchner, Quang-Huy Nguyen, Christophe Ringeissen.

Mots clés : spécification, vérification, démonstration automatique, télécommunications..

Résumé : *L'objectif du projet CALIFE, est de développer un environnement capable de supporter la spécification, la vérification formelle et la génération de tests de composants logiciels destinés à garantir la qualité des algorithmes critiques dans les réseaux de télécommunications. Les partenaires industriels impliqués dans ce projet RNRT sont la société CRIL Ingénierie, le CNET, la société Alcatel CIT.*

L'utilisation à grande échelle d'outils d'aide à la vérification formelle passe par la mise en œuvre de procédures efficaces permettant de démontrer automatiquement des propriétés valides dans certaines théories courantes, comme la logique des propositions, l'arithmétique de Presburger, les systèmes de transitions finis. Les applications visées se situent dans le cadre de la certification des logiciels de télécommunications. Notre objectif dans ce projet est de combiner l'efficacité de la démonstration automatique avec la puissance, la souplesse et la fiabilité des systèmes de preuve généralistes. Nous étudions dans ce cadre un premier prototype d'interface entre ELAN et l'assistant de preuves Coq. Ainsi, pour une procédure de décision donnée, ELAN se charge de découvrir si la propriété est vraie et rend de plus une trace décrivant comment parvenir au résultat. Cette trace est transformée en un script de preuve Coq qui permet de valider le théorème dans Coq. Mais les preuves ainsi construites peuvent être très grosses ce qui dégrade les performances du système. C'est pourquoi nous étudions la technique de preuve par réflection qui permet d'internaliser une procédure de décision dans Coq afin d'automatiser certaines preuves de manière sûre et efficace. Les preuves par réflection utilisent abondamment la possibilité de raisonner modulo un calcul, ce qui permet de soulager l'utilisateur en supprimant des preuves les arguments calculatoires. L'intégration des techniques de réécriture à la déduction et à la réduction du lambda-calcul est à l'étude dans ce cadre.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

Nous participons aux projets suivants du GDR/PRC ALP, sur les thèmes contraintes et déduction automatique : Collaboration de solveurs, Langages et Outils pour la DEduction sous Contraintes, Mélanges de Systèmes algébriques et logiques.

Nous coordonnons l'action de recherche coopérative INRIA Presysa : preuves par réécriture de systèmes synchrones et asynchrones.

8.2 Actions financées par la Commission Européenne

Nous participons au groupe de travail Basic Research Action d'ESPRIT qui, sous l'acronyme *CCL II* et le titre *Construction of Computational Logics*, fait intervenir les universités et les centres de recherche en informatique de Barcelone, Madrid, Munich, Orsay, Saarbrücken, ainsi que la société COSYTEC.

Nous participons au Groupe de Travail ESPRIT CoFI, *Common Framework Initiative for Algebraic Specification and Development*, et nous coordonnons le Tools Task Group.

8.3 Réseaux et groupes de travail internationaux

Nous sommes engagés dans le réseau d'excellence COMPULOG qui regroupe de nombreux sites européens travaillant sur la *programmation logique*.

Nous participons aussi aux groupes de travail *ERCIM Constraints* coordonné par K. Apt (CWI, Amsterdam) et *Programming Languages Technology* coordonné par N. Jones (Diku, Copenhague).

Nous participons au projet franco-autrichien Amadeus sur les *Contraintes Ensemblistes Récurrentes*. Les autres participants sont le LIFO (Orléans) et le LERIA (Angers) du côté français, et l'Université Technique à Vienne du côté autrichien.

8.4 Relations bilatérales internationales

8.4.1 Europe

Nous entretenons des relations avec l'université d'Amsterdam et le CWI dans le domaine de la réécriture et des spécifications formelles algébriques. Nous sommes en relation avec l'Université de Gènes et avec les universités de Kaiserslautern et de Saarbrücken, le DFKI et le MPI, dans le domaine de la démonstration automatique.

8.4.2 Méditerranée, Maghreb et Proche-Orient

Nous sommes en relation avec l'université de Tunis au travers d'enseignements et d'accueil d'étudiants.

8.5 Accueils de chercheurs étrangers

– Abdessamad Imine, Université d'Oran, a travaillé dans l'équipe sur la validation de spécifications TLA en SPIKE.
– Sofiene Thahar, Université Concordia (Montréal, Canada), a travaillé dans l'équipe sur la vérification du "Fairisle ATM Switch Fabric" avec SPIKE.

9 Diffusion de résultats

9.1 Animation de la Communauté scientifique

On trouvera ci-dessous la liste des responsabilités assumées par les membres du projet.

- Alexander Bockmayr : responsable de l'action Bioinformatique du LORIA.
  Membre du comité de programme de CP'99.
- Eric Domenjoud : membre de la commission de spécialiste (27e section) de l'Université de Metz.
- Nicolas Hermann : représentant du LORIA aux Doctoriales'99.
  Organisateur du workshop «Complexité en Déduction Automatique» du LORIA (29 juin 1999).
  Organisateur des journées du projet Amadeus à Orléans du 9 au 10 décembre 1999.
- Claude Kirchner : membre de la Commission d'évaluation de l'INRIA; président du Comité des Projets de l'INRIA-Lorraine et du LORIA; membre du Comité de Direction du LORIA; membre de la commission de spécialistes (27e section) de l'INPL, de l'Université de Metz et de Paris 6; membre du comité scientifique du LIP6 (Université de Paris 6)et du LIFO (Université d'Orléans); membre du comité scientifique du «Laboratório de Inteligência Artificial e Ciência de Computodores», laboratoire d'informatique de l'université de Porto (Portugal).
  Membre des comités éditoriaux des journaux : *Journal of Automated Reasoning*, *Journal of Symbolic Computation*, *Journal of Logic Programming* et *Journal of the Egyptian Mathematical Society* et *Journal of Information Science and Engineering*;
  Membre des comités de programmes des conférences «International Conference on Principles and Practice of Constraint Programming» : CP'99 et «XIX International Conference of the Chilean Computer Science Society» : SCCC'99
  Vice président du groupe de travail IFIP WG 1.6 sur la réécriture. Membre du conseil d'administration de CADE-inc (Board of trustees).
- Hélène Kirchner : membre du Comité de Direction du LORIA; membre du Comité des Projets de l'INRIA-Lorraine; membre nommé du Conseil de laboratoire du LORIA; membre de la commission de spécialistes (27e section) de Nancy 2; expertises au MENRT; co-responsable de l'action *Qualité et sûreté des logiciels et systèmes informatiques* du LORIA.
  Membre du Comité éditorial de la revue *Annals of Mathematics and Artificial Intelligence.*
  Co-chair du workshop CADE «Strategies in automated deduction»; membre du comité de programme de CADE-16; co-chair de FroCoS'2000 «Frontiers of Combining Systems»; membre du comité d'organisation de la conférence PPDP.
  Responsable du *Tools Task Group of CoFI* (*Common Framework Initiative for Algebraic Specifications*).

Membre des groupes IFIP WG 1.3 (*Foundations of Systems Specifications*) et IFIP WG 1.6 (*Term Rewriting*);

- Christophe Ringeissen : co-chair de FroCoS'2000; co-animateur avec Philippe Devienne du groupe du GDR ALP *Collaboration de Solveurs*.
- Michaël Rusinowitch : membre suppléant de la commission d'évaluation INRIA et des Commissions de spécialistes Nancy 2 et INPL; responsable-adjoint DEA Informatique Nancy.
  Co-chairman de RTA'99; membre du comité' MFCS'99; membre du comité d'organisation de la conférence RTA; membre du groupe IFIP WG 1.6; membre du comité d'organisation du workshop «Modélisation et Vérification» (Besançon, décembre 1999); responsable de l'ARC Presysa; membre du steering committee de International Workshop on First Order Theorem Proving (FTP); membre du comité de programme de FTP'2000.
- Laurent Vigneron : membre élu du conseil de laboratoire du LORIA; membre du comité de rédaction de «La lettre du LORIA»; représentant du projet PROTHEO à la COMIN.
  Secrétaire du groupe IFIP WG 1.6.

9.2 Enseignement universitaire

On trouvera ci-dessous la liste des enseignements universitaires effectués par les membres du projet.

– Adel Bouhoula a participé à l'enseignement du module *Spécification de Programmes Spécification de Logiciels* de l'ESIAL (3ème année) et du DESS d'informatique de l'Université Henri Poincaré – Nancy 1.
Il a enseigné la *Validation Automatique de Programmes* dans le cadre du module *génie logiciel* du DEA d'Informatique de l'Université de Tunis et dans le cadre de la deuxième année de l'école Sup'Com de Tunis.

– Isabelle Gnaedig a coordonné les enseignements du module *Spécification de Programmes Spécification de Logiciels* de l'ESIAL (3ème année) et du DESS d'informatique de l'Université Henri Poincaré – Nancy 1.
Elle a également organisé et encadré des travaux dirigés et travaux pratiques autour des spécifications algébriques, du langage LOTOS et de la sémantique des processus communicants dans le cadre de ce module.

– Nicolas Hermann a enseigné la théorie du codage en IUP2 GEII RNC à l'Université Henri Poincaré – Nancy 1.

– Alexander Bockmayr, Adel Bouhoula, Claude Kirchner, Hélène Kirchner, Michaël Rusinowitch ont enseigné des modules *Logique, Déduction et Programmation avec Contraintes, Démonstration assistée par ordinateur* dans le cadre du DEA d'Informatique de l'Université Henri Poincaré – Nancy 1.

– Hélène Kirchner a donné un cours de 6h à l'ENS Cachan *La réécriture: de la logique au calcul et à la preuve* au niveau du troisième cycle d'informatique.

Le projet comprend également des Moniteurs, ATERs, Maîtres de Conférences et Professeurs qui enseignent dans différentes universités de l'académie dans le cadre normal de leurs activités.

9.3 Participation à des colloques, séminaires, invitations

9.3.1 Colloques, tutoriels, conférences et séminaires invités

- Alexander Bockmayr : exposé à JIM'99 (Metz, 17-19 mai 1999) «Parallel Branch-and-Infer»; exposé à JFPLC/JNPC'99 (Lyon, 2-4 juin 1999) «Branch-and-Infer : A Unifying Framework for Integer Linear Programming and Finite Domain Constraint Programming»; exposé au LIFO (Orléans, 18 octobre 1999) «Integer Programs and Valid Inequalities for Planning Problems».
- Nicolas Hermann : exposés au séminaire du département d'informatique de l'Université Paris 12 (Créteil, mars 1999); séminaire *Kurt Gödel* de l'Université Technique de Vienne (Autriche, mai 1999) et au département d'informatique de l'Université de Bratislava (mai 1999).

  Il a donné un cours sur la programmation logique avec contraintes à l'Université Technique de Vienne (mai 1999).
- Claude Kirchner : conférence invitée au «third international CADE workshop on strategies in automated deduction» (5 Juillet 1999); séminaire au SRI International (29 Juillet 1999) sur la déduction modulo et son application à la mécanisation du raisonnement en logique d'ordre supérieur; séminaire à l'université Pontificale Catholique de Santiago (5 novembre) sur la déduction modulo; conférence invitée à la XIX «International Conference of the Chilean Computer Science Society» (11-12 novembre).

  Il a donné un cours sur la déduction modulo dans le cadre de l'école «School of logic and computation» (Edinbourg, 10-11 Avril), un cours sur la résolution de contraintes symboliques par les méthodes syntaxiques à l'école CCL Summer school (5-8 Septembre), et un cours d'introduction à la réécriture pour le calcul et le raisonnement à l'université technique Federica Santa Maria à Valparaiso.

  Il a présenté le calcul de réécriture et de la déduction modulo au Workshop «Automated Deduction» (Dagstuhl, 1-5 mars), le calcul de réécriture au GRD «Mélanges de systèmes algébriques et de systèmes logiques» (9 Avril) et le calcul de réécriture à Unif'99 (28-29 Juin).
- Hélène Kirchner : exposé à JFPL'99 (Lyon) sur le système ELAN; exposés au DFKI (Saarbruecken, 16 mars 1999) «Programming with rewrite rules and strategies in ELAN»; séminaire au SRI International (27 Juillet 1999) «Two semantics for ELAN»; séminaire aux journées BUG et Objectif Zéro Défaut (Lille, 29-30 novembre 1999) «L'environnement CoFI pour les spécifications algébriques».
- Michaël Rusinowitch a présenté les langages de motifs de mot et d'arbre au workshop «Complexité en Déduction Automatique» (28 juin).

Mis à part les congrès et colloques où furent présentés nos travaux, nous avons participé aux événements suivants : AMAST'98, ETAPS'99, WADT'99, FLOC'99, FM'99, JFPL'99, MFCS'99, FEMSYS'99, RECOMB'99, Workshop CCL et le Workshop on the Theory and Practice of Integer Programming in honor of Ralph E. Gomory on the Occasion of his 70th Birthday.

Dans le cadre des actions nationales, nous avons participé à des Journées du GDR/PRC ALP-AMI.

Dans le cadre de groupes de travail européens, nous avons participé à des réunions de travail CoFI, CCL et *ERCIM Working Group on Constraints*.

9.3.2 Séjours de chercheurs

– Nicolas Hermann a passé le mois de mai 1999 à l'Université Technique de Vienne (Autriche), département d'informatique, sur l'invitation du groupe du professeur Alexander Leitsch, où il a donné des cours sur la Programmation Logique avec Contraintes et a continué la collaboration scientifique avec Gernot Salzer.
– Nicolas Hermann a passé deux semaines au département d'informatique de l'Université de Californie à Santa Cruz, sur l'invitation du professeur Phokion G. Kolaitis, en août 1999.
– Claude Kirchner, Hélène Kirchner et Pierre-Etienne Moreau ont fait chacun un séjour de deux semaines en juillet 1999 au SRI (Californie) dans le cadre d'une coopération avec José Meseguer et le groupe Maude, autour de la logique de réécriture. Trois séminaires ont été présentés sur nos travaux sur ELAN et sur la déduction modulo.
– Claude Kirchner a visité l'Université technique Federica Santa Maria à Valparaiso du 2 au 15 Novembre 1999.

9.4 Jurys de thèses

Nous avons pris part aux jurys de thèses et d'habilitations suivants :

– Alexander Bockmayr : Eric Bourreau (Paris 13), Pierre-Etienne Moreau (Nancy).
– Nicolas Hermann : Laurent Juban (Nancy).
– Claude Kirchner : Christelle Scharff (Nancy), Ian Gamtini (Aix-Marseille).
– Hélène Kirchner : Jean-Yves Vion-Dury (Grenoble), Pierre-Etienne Moreau (Nancy), Vincent Schachter (Paris-Sud Orsay).
– Michael Rusinowitch : Mathieu Raffinot (Marne-la-Vallée), Nicolas Navet (Nancy), Philippe Balbiani (habilitation, Paris-Nord), Mohamed Tajine (habilitation, Strasbourg).

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] A. Bouhoula, M. Rusinowitch, « Implicit Induction in Conditional Theories », *Journal of Automated Reasoning 14*, 2, 1995, p. 189–235.

[2] A. Bouhoula, « Automated Theorem Proving by Test Set Induction », *Journal of Symbolic Computation 23*, 1, 1997, p. 47–77.

[3] C. Castro, « Building Constraint Satisfaction Problem Solvers Using Rewrite Rules and Strategies », *Fundamenta Informaticae 34*, 3, 1998, p. 263–293.

[4] N. Hermann, P.-G. Kolaitis, « The Complexity of Counting Problems in Equational Matching », *Journal of Symbolic Computation 20*, 3, septembre 1995, p. 343–362, L'article et le numéro de la revue sont parus effectivement en avril 1996 mais antidatés par l'éditeur.

[5] C. Hintermeier, C. Kirchner, H. Kirchner, « Dynamically-Typed Computations for Order-Sorted Equational Presentations », *Journal of Symbolic Computation 25*, 4, avril 1998, p. 455–526.

[6] C. Kirchner, H. Kirchner, M. Rusinowitch, « Deduction with Symbolic Constraints », *Revue d'Intelligence Artificielle 4*, 3, 1990, p. 9–52.

[7] C. Kirchner, H. Kirchner, M. Vittek, « Designing Constraint Logic Programming Languages using Computational Systems », *in : Principles and Practice of Constraint Programming.*

The Newport Papers, V. Saraswat et P. Van Hentenryck (éditeurs), The MIT Press, 1995, ch. 1, p. 131–158.

[8] C. Kirchner, C. Ringeissen, « Rule-Based Constraint Programming », *Fundamenta Informaticae 34*, 1998, p. 225–262.

[9] H. Kirchner, C. Ringeissen, « Combining Symbolic Constraint Solvers on Algebraic Domains », *J. Symbolic Computation 18*, 2, août 1994, p. 113–155.

[10] M. Rusinowitch, L. Vigneron, « Automated Deduction with Associative Commutative Operators », *Applicable Algebra in Engineering, Communication and Computing 6*, 1, janvier 1995, p. 23–56.

Livres et monographies

[11] M. Cerioli, M. Gogolla, H. Kirchner, B. Krieg-Brueckner, Z. Qian, M. Wolf, *Algebraic System Specification and Developement :Survey and Annotated Bibliography*, *Monographs of the Bremen Institute of Safe Systems*, *3*, Shaker Verlag, Bremen, Germany, août 1999.

Thèses et habilitations à diriger des recherches

[12] L. Juban, *Sur les problèmes de complexité en déduction automatique: base de Hilbert, modèles uniques et minimaux*, Thèse de doctorat d'université, Université Henri Poincaré – Nancy 1, Décembre 1999.

[13] P.-E. Moreau, *Compilation de règles de réécriture et de stratégies non-déterministes*, Thèse d'université, Nancy, juin 1999, `http://www.loria.fr/publications/1999/99-T-083/99-T-083.ps`.

[14] C. Scharff, *Déduction avec contraintes et simplification dans les théories équationnelles*, Thèse de doctorat d'université, Université Henri Poincaré – Nancy 1, Septembre 1999.

Articles et chapitres de livre

[15] A. Bockmayr, F. Eisenbrand, M. Hartmann, A. S. Schulz, « On the Chvátal Rank of Polytopes in the 0/1 Cube », *Discrete Applied Mathematics 98*, 1999, p. 21–27.

[16] P. Borovansky, C. Kirchner, H. Kirchner, C. Ringeissen, « Rewriting with strategies in ELAN : a functional semantics. », *International Journal of Foundations of Computer Science*, 1999.

[17] H. Ganzinger, F. Jacquemard, M. Veanes, « Rigid reachability, the non-symmetric form of rigid E-unification », *International Journal of Foundations of Computer Science , IJFCS*, 1999.

[18] M. Hermann, P. G. Kolaitis, « Computational Complexity of Simultaneous Elementary Matching Problems », *Journal of Automated Reasoning 23*, 2, août 1999, p. 107–136.

[19] C. Kirchner, H. Comon, M. Dincbas, J.-P. Jouannaud, « A Methodological View of Constraint Solving », *Constraints*, décembre 1999.

[20] H. Kirchner, « Term Rewriting », *in : Algebraic Foundations of Systems Specifications*, *IFIP State-of-the-Art Reports*, Springer, septembre 1999, ch. 9, p. 273–320.

[21] E. Monfroy, C. Ringeissen, « An Open Automated Framework for Constraint Solver Extension : the SoleX Approach », *Fundamenta Informaticae 39*, 1999, p. 167–187.

Communications à des congrès, colloques, etc.

[22] A. Bockmayr, Y. Dimopoulos, « Integer Programs and Valid Inequalities for Planning Problems », *in : 5th European Conference on Planning, ECP-99, Durham, United Kingdom*, *Lecture Notes in Computer Science*, Springer, 1999.

[23] A. Bouhoula, A. Jebali, M. Rusinowitch, « Reasoning about Object Behaviours with Rewriting. », *in : OOSDS'99 Workshop on Object-Oriented Specification Techniques for Distributed Systems and Behaviours (in conjunction with Principles, Logics, and Implementations of high-level programming languages , PLI99), Paris*, septembre 1999.

[24] H. Cirstea, C. Kirchner, « Combining Higher-Order and First-Order Computation Using Rho Calculus : Towards a Semantics of ELAN », *in : FroCoS'98, Amsterdam*, 1999.

[25] H. Cirstea, « Specifying Authentication Protocols Using ELAN », *in : Workshop on Modelling and Verification*, décembre 1999, `http://www.loria.fr/publications/1999/99-R-246/99-R-246.ps`.

[26] V. Cortier, H. Ganzinger, F. Jacquemard, M. Veanes, « Decidable fragments of simultaneous rigid reachability », *in : ICALP, Prague, Czech Republic*, *LNCS*, *1644*, Springer-Verlag, p. 250–260, juillet 1999.

[27] G. Dowek, T. Hardin, C. Kirchner, « HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic », *in : RTA'99*, juin 1999.

[28] H. Dubois, H. Kirchner, « Rule Based Programming with Constraints and Strategies », *in : Workshop of the ERCIM Working Group on Constraints, Paphos, Chypre*, octobre 1999, `http://www.loria.fr/publications/1999/99-R-084/99-R-084.ps`.

[29] A. Durand, M. Hermann, L. Juban, « On the complexity of recognizing the Hilbert basis of a linear Diophantine system », *in : 24th International Symposium on Mathematical Foundations of Computer Science (MFCS'99), Szklarska Poreba (Poland)*, *Lecture Notes in Computer Science*, Springer-Verlag, septembre 1999.

[30] M. Hermann, L. Juban, P. G. Kolaitis, « On the Complexity of Counting the Hilbert Basis of a Linear Diophantine System », *in : 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), Tbilisi ,Republic of Georgia*, *Lecture Notes in Computer Science (in Artificial Intelligence)*, *1705*, Springer-Verlag, p. 13–32, septembre 1999.

[31] L. Juban, « Dichotomy Theorem for the Generalized Unique Satisfiability Problem », *in : Fundamentals of Computation Theory (FCT'99), Iasi (Romania)*, *Lecture Notes in Computer Science*, *1684*, Springer-Verlag, p. 327–337, août 1999.

[32] H. Kirchner, P. D. Mosses, « Algebraic Specifications, Higher-Order Types, and Set-Theoretic Models », *in : 7th International Conference onAlgebraic Methodology and Software Technology - AMAST'98, Amazonia, Bresil*, *Lecture Notes in Computer Science*, *1548*, Springer, p. 373–388, janvier 1999.

[33] H. Kirchner, « ELAN », *in : JFPLC'99 Journées Francophones de Programmation Logique et programmation par Contraintes, Lyon, France*, F.Fages, HERMES Science Publications, p. 241–248, Paris, mai 1999.

[34] F. Klay, M. Rusinowitch, S. Stratulat, « Analysing Feature Interactions with Automated Deduction Systems », *in : 7th International Conference on Telecommunication Systems Modeling and Analysis, Nashville, Tennessee, USA*, Bezalel Gavish, mars 1999.

[35] G. Kucherov, M. Rusinowitch, « Patterns in words versus patterns in trees : a brief survey and new results », *in : International Conference "Perspectives of Systems Informatics", Novosibirsk*, *LNCS*, Springer-Verlag, juillet 1999.

[36] P. Narendran, M. Rusinowitch, « Rewriting Techniques and Applications. 10th International Conference, RTA'99 », *in : RTA'99, Trento, Italy*, *LNCS*, *1631*, Springer Verlag, p. 1–396, juillet 1999.

[37] M. Rusinowitch, S. Stratulat, F. Klay, « Mechanical Verification of a Generic Incremental ABR Conformance Algorithm », *in : Workshop on Modelling and Verification, Besancon*, décembre 1999.

[38] L. Vigneron, A. Wasilewska, « Rough Sets based Proofs Visualisation », *in : 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99), invited session on Granular Computing and Rough Sets, New York*, IEEE, p. 805–808, juin 1999, `http://www.loria.fr/publications/1999/99-R-056/99-R-056.ps`.

Rapports de recherche et publications internes

[39] A. Armando, G. Defourneaux, M. Rusinowitch, S. Stratulat, « Integrating decision procedures in Spike », *Rapport de recherche*, mai 1999.

[40] R. Azaiez, « Vérification des Systèmes Réactifs Dans le Modèle Synchrone », *Stage de dea*, LORIA, juin 1999.

[41] H. Dubois, H. Kirchner, « Modelling Planning Problems with Rules and Strategies », *Rapport de recherche*, mars 1999, `http://www.loria.fr/publications/1999/99-R-029/99-R-029.ps`.

[42] I. Gnaedig, H. Kirchner, T. Genet, « Induction for Termination », *Rapport de recherche*, décembre 1999.

[43] A. JEBALI, « Vérification Observationnelle », *Stage de dea*, Université Henri Poincaré Nancy I, juin 1999.

[44] H. Kirchner, C. Ringeissen, « Executing CASL Equational Specifications with the ELAN Rewrite Engine », *Rapport de recherche*, 1999, Note T-9 of CoFI, ESPRIT Working Group 29432.

[45] J. Musset, « Preuves de systèmes réactifs dans le modèle synchrone », *Stage de dea*, Dea d'Informatique de Lyon, juillet 1999.