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 I, 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]

Assistante de projet

Christelle Bergeret-Etienne [TR INRIA]

Personnel INRIA

Adel Bouhoula [CR, en disponibilité à partir du 1/12/2000]

Olivier Bournez [CR]

Isabelle Gnaedig-Antoine [CR]

Florent Jacquemard [CR, en disponibilité à partir du 1/10/2000]

Claude Kirchner [DR]

Christophe Ringeissen [CR]

Pierre-Etienne Moreau [CR, à partir du 1/09/2000]

Personnel CNRS

Eric Domenjoud [CR]

Nicolas Hermann [CR]

Personnel Université

Alexander Bockmayr [Professeur, Université Henri Poincaré - Nancy I]

Laurent Vigneron [Maître de Conférences, Université Nancy 2]

Acceuil Jeune

Hassen Kacem [INRIA, à partir du 1/09/2000]

Chercheurs doctorants

Yannick Chevalier [ATER, à partir du 1/09/2000]

Horatiu Cirstea [ATER, jusqu'au 31/08/2000]

Eric Deplagne [allocataire MNERT]

Hubert Dubois [allocataire MNERT]

Damien Eveillard [boursier UHP, à partir du 1/12/2000]

Olivier Fissore [boursier BDI CNRS — Région Lorraine, à partir du 1/09/2000]

Laurent Juban [ATER, jusqu'au 31/08/2000]

Julien Musset [boursier ENS]

Quang-Huy Nguyen [boursier INRIA]

Sorin Stratulat [ATER, jusqu'au 31/08/2000]

Mathieu Turuani [allocataire MNERT, à partir du 1/09/2000]

Chercheurs post-doctorants

Pierre-Etienne Moreau [Post-doc industriel INRIA jusqu'au 31/08/2000]

Zhebin Qian [Post-doc UHP, à partir du 1/10/2000]

Jürgen Stuber [Post-doc INRIA, à partir du 1/06/2000]

Jianyang Zhou [Post-doc création d'entreprise, jusqu'au 31/03/2000]

Conseillers extérieurs

Horatiu Cirstea [du 1/09/2000 au 31/12/2000]

Sorin Stratulat [du 1/09/2000 au 30/11/2000]

Collaborateur extérieur

Jiany Zhou [jusqu'au 31/02/2000]

Chercheurs invités

Vladimir Lounine [du 1/06/2000 au 30/09/2000]

Silvio Ranise [du 15/05/2000 au 16/06/2000]

Marian Vittek [du 3/07/2000 au 15/08/2000 et du 29/11/2000 au 13/12/2000]

Contractuels

Nicolaï Pisaruk [Chercheur, du 15/03/2000 au 15/03/2001]

Stéphane Riviere [Ingénieur d'étude, du 1/10/2000 au 30/09/2001]

Stagiaires

Tarek Abbes [4 mois]

Noual Anibar [4 mois]

Emmanuel Beffara [ENS Lyon, 1 mois]

Arnaud Carayol [ENS Lyon, 1 mois]

Mounir Chtioui [4 mois]

Fakhri Elmekki [4 mois]

Hassen Kacem [4 mois]

Julien Narboux [ENS Lyon, 1 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 le raisonnement 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, l'optimisation combinatoire et la bioinformatique.

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, etc. 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 mécanisation du raisonnement 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 : contrainte, résolution, satisfaisabilité, propagation, programmation entière, combinaison, collaboration de résolveurs.

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 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.*

[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.

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.

À 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.

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.

[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.

3.3 Démonstration automatique

Mots clés : déduction, réécriture, récurrence, contrainte, 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,McC97] de résoudre un problème mathématique ouvert. La combinaison des contraintes avec les simplifications par réécriture 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, logiciel de télécommunication, logiciel 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

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

[McC97] W. McCune, «Solution of the Robbins Problem», *JAR 19*, 3, 1997, p. 263–276.

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 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 biologiques, de systèmes réactifs et de systèmes physiques hybrides, c'est-à-dire ayant des comportements discrets et continus.

5 Logiciels

5.1 CASRUL

Mots clés : Protocoles cryptographiques, vérification automatique.

Participants : Florent Jacquemard, Michaël Rusinowitch [correspondant], Laurent Vigneron.

CASRUL est un système de vérification automatique de protocoles cryptographiques. Cet outil, écrit en Objective Caml© avec l'aide de Julien Narboux, a pour but de traduire un protocole donné dans une syntaxe abstraite classique, en un système de réécriture. Ce système peut alors être utilisé par tout démonstrateur en logique équationnelle pour détecter automatiquement des attaques.

Nous avons utilisé le démonstrateur daTac pour étudier de nombreux protocoles : EKE, NSPK, Otway Rees, SRA, TMN, etc. Cela nous a permis de retrouver de nombreuses attaques dans ces protocoles, ce qui est très encourageant pour la suite.

CASRUL [1] n'est pas encore diffusé, mais une page web décrit son contenu et donne la liste des protocoles que nous avons étudiés, ainsi que les attaques trouvées.

5.2 daTac

Mots clés : Déduction automatique, logique du premier ordre avec égalité, théories associatives-commutatives, contraintes symboliques.

Participants : Michaël Rusinowitch, Laurent Vigneron [correspondant].

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.

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

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

Cette année, les principales évolutions de ce logiciel sont sa traduction en Objective Caml© et son enregistrement à l'APP (Agence pour la Protection des Programmes).

daTac[2] est documenté, maintenu, diffusé par ftp et accessible sur le Web (version 0.91).

5.3 ELAN

Mots clés : règles, stratégies, spécification, prototypage, calcul, déduction.

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

ELAN [BKK+98] est un environnement de spécification et de prototypage fondé sur l'application de règles de réécriture contrôlée au moyen de stratégies. Il offre un cadre logique simple et naturel pour combiner les paradigmes de calcul et de déduction. Ses originalités principales sont d'implanter des techniques de filtrage et de réduction de termes modulo l'associativité-commutativité, compilées de façon très efficace ; le traitement de calculs non-déterministes, i.e. pouvant retourner plusieurs résultats, dont l'énumération est gérée 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. ELAN est utilisé pour prototyper des démonstrateurs de théorèmes, des langages 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 respectivement 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 Linux-Intel. Nous distribuons actuellement la version 3.4 [56].

ELAN[3] est documenté, maintenu, diffusé par ftp et accessible sur le Web.

5.4 SPIKE

Mots clés : preuves par récurrence, cohérence de spécifications, complétude de définition de fonctions.

Participants : Adel Bouhoula [correspondant], Michaël Rusinowitch, Sorin Stratulat.

Le système SPIKE [BR95] est un démonstrateur automatique de théorèmes dans les théories conditionnelles. Une version de SPIKE datant de 1995 est écrite en Caml Light©, langage

2. http://www.loria.fr/equipes/protheo/SOFTWARES/DATAC/
3. http://elan.loria.fr

[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, `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.

fonctionnel de la famille ML. Le logiciel est muni d'une interface graphique TCL/TK (X11 Toolkit).

Le système SPIKE s'inscrit dans le cadre des outils de vérification de programmes. Ses 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 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. SPIKE s'attache à réduire le nombre d'interactions par l'automatisation des tâches routinières de preuves.

Un certain nombre de problèmes ont pu être traités automatiquement par SPIKE avec une interaction plus faible qu'avec d'autres systèmes de preuves automatiques comme NQTHM, RRL, LP et PVS (tour de Gilbreath, théorème de Ramsey, théorème de binôme, correction de circuits digitaux, par exemple).

Un nouveau prototype (en Objective Caml©) intégrant des stratégies et des procédures de décision est en cours de développement.

SPIKE[4] est documenté, 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, règle de production.

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.*

6.1.1 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

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

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. D'un point de vue pratique, la gestion du double non-déterminisme est traité de manière unique et repose sur l'utilisation de la bibliothèque de gestion du *backtracking*. 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. Ces travaux ont été présentés à la conférence *Rewriting Techniques and Applications* [44] et seront publiés dans le *Journal of Functional Programming* [22].

6.1.2 Syntaxes abstraites comme formats d'échange

Participants : Hélène Kirchner, Christophe Ringeissen.

Nous avons poursuivi nos travaux sur la conception d'une nouvelle syntaxe abstraite pour ELAN, utilisée comme format d'échange permettant d'une part de concevoir une architecture modulaire pour le système ELAN et d'autre part d'utiliser les outils de réécriture d'ELAN en connexion avec d'autres systèmes et langages.

Cette syntaxe abstraite, appelée Efix et conçue en collaboration avec Mark van den Brand du groupe ASF+SDF (CWI, Amsterdam), s'apparente à celle utilisée dans le nouvel environnement de programmation ASF+SDF (Asfix) et à celle choisie pour CASL (Casfix), le langage de spécification algébrique émanant du projet CoFI (Common Framework Initiative for Algebraic Specifications) [Mos97] qui rassemble une grande partie de la communauté européenne travaillant sur les spécifications algébriques. Toutes ces syntaxes abstraites sont des instances d'une même structure de termes, les `ATerms`, diffusée sous la forme d'une librairie (`C` et `Java`) par le groupe ASF+SDF.

Nous avons développé des outils pour engendrer, manipuler et transformer un terme représenté dans la syntaxe abstraite Efix.

En particulier, nous avons réalisé, en collaboration avec Mark van den Brand, un analyseur syntaxique dont le rôle consiste à engendrer le terme Efix correspondant à tout programme ELAN (avec règles et stratégies). Cette réalisation a permis de revoir et clarifier la syntaxe du langage ELAN en réutilisant les outils d'analyse syntaxique développés au sein du groupe ASF+SDF [50].

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

[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.

Par ailleurs, nous avons également travaillé sur un autre outil de conversion dont le rôle est de convertir une syntaxe abstraite de CASL en la syntaxe abstraite Efix. Utilisé conjointement à l'outil précédent, il rend exécutable une classe importante de spécifications CASL grâce à l'utilisation du moteur de réécriture d'ELAN (interpréteur ou compilateur). Les spécifications CASL considérées peuvent désormais être structurées et contenir des opérateurs associatifs-commutatifs ainsi que des prédicats. Le système ELAN fournit ainsi des outils d'exécution pour CASL qui sont mis à la disposition de la communauté CoFI. Un article décrivant le langage CASL et le projet CoFI [14] a été accepté pour un numéro spécial de TCS consacré aux *Current trends in Algebraic Development Techniques.*

6.1.3 Le ρ-calcul - calcul de réécriture

Participants : Horatiu Cirstea, Claude Kirchner.

Nos travaux précédents sur la logique de réécriture et ELAN ont mis en évidence que le contrôle de la réécriture doit être explicite et peut être décrit naturellement en utilisant la réécriture [17]. Cette observation nous a amenés à un nouveau concept généralisant le λ-calcul et la réécriture que nous avons appelé le ρ-calcul et qui est en particulier développé dans la thèse d'Horatiu Cirstea [11].

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 [11] qui utilise un système de substitution explicite et qui généralise le $\lambda\sigma$-calcul, et prouvé la confluence du calcul avec substitutions explicites. Une partie de ces résultats ont été présentés en particulier dans [35] et sont synthétisés dans l'article [59].

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. Il permet également de donner une expression unifiée des calculs objets classiques tels que le *Lambda Calculus of Objects* de Fisher, Honsell et Mitchell, ainsi que le *Object Calculus* d'Abadi et

Cardelli. Nous avons développé ces résultats dans le cadre d'une vision simplifiée du calcul développée en collaboration avec Luigi Liquori du projet ECOO [58].

Par ailleurs nous avons étudié une version simplement typé du calcul de réécriture et montré que pour ce système de type le calcul est fortement normalisant et préserve les types [36].

Dans ce cadre et compte tenu de la puissance d'expression de l'abstracteur "$\rightarrow$", il était naturel de considérer un système de type plus élaboré où l'abstraction sur les types est également décrite par réécriture. Ceci nous a amené à définir, avec Luigi Liquori, un système de type très général basé sur l'abstraction "$\rightarrow$" et à généraliser le λ-cube de Barendregt dans un ρ-cube. Ce travail, qui en est encore à ses débuts, est décrit dans [57].

6.1.4 Réécriture Objets

Participants : Hubert Dubois, Hélène Kirchner.

Une extension du langage ELAN par les concepts objets a été proposé dans [38, 60]. Cette extension est motivée par la nécessité de représenter des données structurées et des états, et de pouvoir les combiner avec des règles et des stratégies qui décrivent leur évolution. Le langage offre la possibilité de définir des classes d'objets avec attributs et méthodes. Les classes sont elles-mêmes des objects avec des méthodes particulières, en particulier la méthode *new* qui crée un nouvel objet de la classe. Les objects ont des attributs dont les valeurs sont accessibles et transformables, et les méthodes peuvent être cachées ou révélées en modifiant les possibilités d'y accéder dans l'objet auxquel elles appartiennent. Sémantiquement, cette extension est en fait une instance particulière du ρ-calcul. En combinant les concepts d'objets, de règles et de stratégies, on obtient un langage particulièrement expressif qui est fondé sur une sémantique claire basée sur le $\rho-$calcul [58].

6.1.5 Règles de production 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 [39], 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 à 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 par exemple 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.

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.

6.2 Contraintes

Mots clés : résolution de contraintes, optimisation discrète, programmation par contraintes, programmation entière, combinaison de résolveurs, recherche opérationnelle, bioinformatique.

Résumé : *Nous avons poursuivi les travaux des années précédentes sur l'extension et la combinaison de résolveurs. Nous étudions la modélisation par règles et la synthèse de résolveurs à base de propagation sur des domaines finis. Nous travaillons sur des techniques de plans de coupe et l'intégration de la programmation entière avec la programmation par contraintes. Les principaux domaines d'application sont la recherche opérationnelle et la bioinformatique.*

6.2.1 Génération de règles pour la résolution de contraintes dans les domaines finis

Participant : Christophe Ringeissen.

Nous avons poursuivi nos travaux sur l'utilisation de règles pour modéliser la résolution de contraintes dans les domaines finis. En collaboration avec Eric Monfroy (CWI, Amsterdam), nous avons étudié le problème de la création automatique de résolveurs à base de règles pour la réduction de problèmes de satisfaction de contraintes. Nous avons ainsi amélioré le schéma général de production de règles proposé initialement par K. Apt et E. Monfroy 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 décrite dans [46] 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.

En réutilisant le même algorithme d'unification, nous avons par ailleurs montré [47] comment représenter les contraintes et engendrer efficacement leurs solutions dans un langage comme ELAN qui permet de spécifier algèbriquement les contraintes par réécriture avec stratégie. Cette approche consiste à construire une règle, qui lorsqu'elle est appliquée de façon non-déterministe, calcule l'ensemble des solutions d'une contrainte. Ce calcul est effectué au

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

moyen de systèmes de réécriture confluents et terminants pour spécifier les fonctions représentées par les graphes acycliques dirigés obtenus par l'algorithme d'unification. Les applications de cette représentation se situent dans le cadre de la transformation de spécifications ELAN. Par cette méthode, on est capable de transformer une spécification (par réécriture) d'une contrainte en une autre qui aura l'avantage de pouvoir s'exécuter plus efficacement.

6.2.2 Théorie des plans de coupe en programmation entière

Participants : Alexander Bockmayr, Nicolai Pisaruk.

Les plans de coupe ont été inventés en 1958 par R. Gomory afin de résoudre des problèmes de programmation linéaire en nombres entiers. Depuis, ils ont été étudiés non seulement en programmation mathématique, mais aussi en logique et en théorie de complexité. Dans [33] nous présentons une synthèse de plusieurs résultats récents sur les plans de coupe, qui sont situés à l'interface de la logique et de la théorie d'optimisation. En particulier, nous discutons la longueur et le rang de preuves par des plans de coupe basés sur le principe de Gomory-Chvátal.

Dans [54] nous étudions la complexité de la clôture élémentaire P' d'un polyèdre P, c.-à.-d. l'intersection de P avec tous les plans de coupe de Gomory-Chvátal. Nous montrons qu'il existe - si l'on fixe la dimension - une borne polynomiale sur le nombre d'inégalités nécessaires pour définir P'. Basé sur ce résultat nous développons en dimension variable un algorithme polynomial pour calculer des plans de coupe de profondeur maximale.

Dans [55] nous caractérisons l'enveloppe convexe de l'ensemble de solutions d'une disjonction de certains systèmes d'inégalités linéaires. En particulier, nous traitons des disjonctions de polymatroïdes. Ce travail est actuellement poursuivi en collaboration avec E. Balas (CMU, Pittsburgh, USA) et L. Wolsey (Core, Louvain-la-Neuve, Belgique).

6.2.3 Application à l'optimisation de la chaîne logistique globale

Participants : Alexander Bockmayr, Nicolai Pisaruk.

Dans le cadre du projet européen LISCOS (Large Scale Integrated Supply Chain Optimisation Software) nous travaillons sur l'optimisation de la chaîne logistique globale d'une entreprise par une combinaison de techniques de programmation entière (branch-and-cut) et programmation par contraintes [52]. Nous avons développé une nouvelle contrainte symbolique `flow` pour modéliser et résoudre des problèmes de flot dans un réseau. Dans [53] nous appliquons cette contrainte pour modéliser un problème générique de planification. En collaboration avec COSYTEC nous avons étudié trois problèmes logistiques de taille industrielle proposés par le partenaire PSA [51].

6.2.4 Applications à la bioinformatique

Participants : Alexander Bockmayr, Damien Eveillard, Vladimir Lounine, Stéphane Rivière.

La bioinformatique est un nouveau domaine d'application des techniques de résolution de

contraintes et d'optimisation discrète développées dans notre équipe. Nous travaillons actuellement sur deux problèmes :

1. Détermination et analyse des enveloppes macromoléculaires : à partir de données expérimentales de cristallographie, nous cherchons à déterminer la distribution de la densité électronique d'une macromolécule sur une grille tridimensionnelle (problème de phasage)

2. Prédiction de la structure secondaire des ARN : à partir de la séquence des nucléotides d'une molécule ARN nous essayons de faire des prédictions sur sa structure secondaire en prenant en compte à la fois des données thermodynamiques, phylogénétiques et expérimentales.

Ces travaux sont faits en collaboration avec le Laboratoire de cristallographie LCM3B (A. Ourjoumtsev) et le Laboratoire de maturation des ARN et enzymologie moléculaire (C. Branlant) de l'Université Henri Poincaré - Nancy I.

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 1998 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 (projet COQ) pour assurer l'élimination de coupures.

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 et les résultats que nous avions obtenus pour l'unification d'ordre supérieur [18], 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 publiés à RTA'99 vont paraitre dans [19].

En formalisant l'axiome d'induction dans le cadre de $HOL_{\lambda\sigma}$(exprimant la logique d'ordre supérieur au premier ordre), nous avons pu montrer que la preuve par induction d'une propriété équationnelle peut être vue comme un processus de déduction modulo, dans lequel on enrichit la congruence avec les différentes hypothèses d'induction. Pour cela il a fallu étendre la déduction modulo à des règles et équations conditionnelles, ainsi qu'à des symboles «protecteurs» limitant l'action de la congruence. On obtient ainsi une nouvelle vision des mécanismes d'induction au premier ordre dits «induction sans induction». Ceux qui procèdent par réécriture correspondent à la réduction du but par la congruence et ceux qui procèdent par complétion correspondent à la simplification de la congruence. On obtient donc la possibilité de faire coopérer dans un cadre formel unifié des techniques de démonstration automatique avec des étapes de démonstration assistée. Nous avons commencé à traduire ces résultats dans un système de preuve permettant leur implantation en ELAN.

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 [37].

Jürgen Stuber a démontré que ENAR est complète par une autre méthode basée sur la construction d'un modèle de Herbrand, suite à son travail sur l'intégration des théories dans les preuves par superposition. Par cette méthode on gagne des notions fortes de redondance pour les clauses et les inférences, ce qui est important pour trouver des preuves en pratique. Mais on est obligé de construire un ordre bien-fondé sur les propositions qui est compatible avec la réécriture et possède d'autres propriétes plus techniques. Nous avons démontré qu'un tel ordre existe pour un petit fragment de la théorie des ensembles [49].

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.

Ces travaux ont été présentés au cours d'un exposé invité au *Winter Workshop in Logics, Types and Rewriting*, à Edinburgh en février 2000 et sont soumis à un journal.

6.3.3 Preuve automatique par récurrence : ensemble couvrant contextuel

Participant : Sorin Stratulat.

Dans sa thèse [12] Sorin Stratulat introduit le concept d'*ensemble couvrant contextuel* (ECC), qui généralise celui d'ensemble couvrant. Le principe de preuve par récurrence avec ECC est décrit par un système d'inférence abstrait. Ce système généralise la plupart des procédures de récurrence basées sur les ensembles couvrants. La thèse présente également des conditions d'applications des ECC dans la construction de règles d'inférence. La thèse aborde le problème de l'intégration *correcte* des techniques de raisonnement dans le cadre des ECC. Les ECC élémentaires sont engendrés par des modules, représentant des implantations de techniques de raisonnement. Grâce à leurs propriétés de composition, on peut engendrer de nouveaux ECC. Comme étude de cas, un ensemble de modules de raisonnement est défini à partir d'un ensemble de techniques de réécriture conditionnelle, comme la réécriture inductive et la réécriture par cas. Puis sur cette base le système d'inférence du démonstrateur SPIKE a pu être spécifié comme une instance du système abstrait. Ceci permet une extension modulaire et incrémentale de SPIKE. Sorin Stratulat montre comment SPIKE, étendu avec la technique de subsomption sémantique inductive et intégrant une procédure de semi-décision pour l'arithmétique linéaire, prouve la correction de l'algorithme MJRTY [BM91] en utilisant une combinaison de raisonnement inductif et arithmétique.

Ces travaux sont également décrit dans l'article [23].

6.3.4 Intégration de procédures de décision

Participants : 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 [27]. Un prototype intégrant l'arithmétique linéaire et la théorie de l'égalité a été développé et validé sur des exemples.

6.3.5 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

[BM91] R. S. Boyer, J. S. Moore, « MJRTY - A Fast Majority Vote Algorithm », *in: Automated Reasoning: Essays in Honor of Woody Bledsoe*, R. S. Boyer (éditeur), *Automated Reasoning, 1*, Kluwer Academic Publishers, 1991, p. 105–117.

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, et sur les automates d'arbres à contraintes.

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.

6.3.6 Clôture par congruence

Participants : Laurent Vigneron, Silvio Ranise, Michael Rusinowitch.

La clôture par congruence est une procédure dont le but est de construire une représentation compacte d'une théorie équationnelle. En collaboration avec L. Bachmair, I. V. Ramakrishnan et A. Tiwari (Université de Stony Brook, NY), nous avons utilisé cette technique pour définir des procédures de décision efficaces fondées sur la normalisation par réécriture, cadres indispensables pour développer des environnements de preuve. 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] ainsi qu'au workshop FreGe'00 [30].
Ces résultats 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.
Ce résultat a été présenté à la conférence FroCoS'2000 [29], et un article pour un journal est en préparation.

Nous étudions également comment obtenir de manière uniforme des procédures de clôture par congruence pour des théories utilisées en vérification en spécialisant des systèmes de preuves par réécriture.

[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.

6.3.7 Contraintes d'ordre

Participant : Michael Rusinowitch.

Les contraintes d'ordres permettent d'écarter les instances closes redondantes dans les preuves par résolution ou paramodulation. Elles permettent aussi d'augmenter le champ d'application des ordres tels que RPO, LPO, KBO. Nous avons montré dans [45] que la théorie de l'ordre RPO est décidable dans le cas d'une signature monadique totalement ordonnée.

6.3.8 Coopération de techniques de preuve

Participants : Hassen Kacem, Claude Kirchner, Hélène Kirchner, Quang-Huy Nguyen.

Dans le cadre d'un projet avec France Telecom R&D, nous nous intéressons à la coopération de différentes techniques de preuve pour vérifier la conformité à des normes dans les réseaux téléphoniques. Pour faire ce genre de vérification, il faut pouvoir combiner l'efficacité de la démonstration automatique avec la souplesse et la fiabilité d'un assistant de preuve. Dans ce projet, nous étudions comment faire coopérer deux systèmes : l'environnement ELAN développé dans le projet PROTHEO, qui permet d'effectuer automatiquement et très efficacement des preuves par réécriture, et l'assistant de preuve COQ développé à l'INRIA-Rocquencourt. ELAN est utilisé pour trouver rapidement des preuves particulières en utilisant la réécriture. Ces preuves sont ensuite importées et internalisées par COQ. Cette coopération est décrite dans [26] présenté au workshop *Logical Frameworks and Metalanguages* affilié à la Conférence LICS (Logic in Computer Science) en juillet 2000. Le stage de 4 mois de Hassen Kacem dans le projet PROTHEO [63] a consisté à étudier cette démarche sur un algorithme particulier. Il a programmé en ELAN un test de satisfiabilité de formules propositionnelles, très utilisé dans le domaine de la vérification de circuits, l'algorithme de Stallmarck, ainsi qu'une variante. Le compilateur ELAN génère une preuve et en transmet la trace à CoQ.

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 : Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner.

L'étude de la terminaison de la réécriture sur les termes clos a été poursuivie. La quasi-totalité des techniques de preuve de terminaison de la réécriture existantes sont des techniques globales, assurant la terminaison des chaînes de calcul partant de tous les termes d'une algèbre

donnée. Qui plus est, les algèbres considérées sont des algèbres libres. Pour assurer la terminaison des programmes dans des langages à base de règles, les besoins sont autres : il est utile de travailler sur des algèbres de termes clos, et de pouvoir caractériser quels ensembles de termes aboutissent à des calculs finis, dans le cas où il n'y a pas terminaison globale. Il peut être aussi très utile de disposer de techniques de preuve pour des stratégies de réécriture particulières.

Dans ce contexte, nous avons proposé une méthode de preuve de terminaison de la réécriture *innermost*, par induction explicite sur la propriété à prouver : on établit qu'un terme termine en supposant que les termes plus petits que lui pour un ordre noethérien terminent. L'ordre n'est pas donné a priori mais contraint au fur et à mesure de la preuve. La méthode proposée repose sur un double mécanisme : une étape d'abstraction des sous-termes d'un terme donné par des variables représentant leurs formes normales, une étape de surréduction du terme obtenu, schématisant les étapes de réécriture *innermost* sur les termes clos. La procédure peut soit diverger, avec une succession infinie d'étapes d'abstraction et de *narrowing*, soit s'arrêter sur un échec si on ne peut comparer le terme de départ et les sous-termes à abstraire, soit s'arrêter uniquement sur des termes auxquels on peut appliquer directement l'hypothèse d'induction, ou sur des termes qu'on ne peut plus surréduire. Dans ce dernier cas, la terminaison du terme donné est établie.

Nous avons, cette année, proposé deux adaptations à notre méthode de preuve. La première consiste à remplacer l'induction noethérienne sur les termes par une induction sur les symboles d'opérateurs. La comparaison des sous-termes du terme courant avec le terme de départ est remplacée par une application récursive de la procédure sur les sous-termes considérés. Cet algorithme permet de conclure dans bon nombre de cas lorsque la méthode de base s'arrête en échec.

La seconde adaptation étend l'algorithme de base de sorte à pouvoir conclure lorsque, partant d'un terme donné, la procédure s'arrête avec succès pour certains des termes dérivés seulement. On caractérise ainsi, via des contraintes égalitaires sur les termes dont les solutions sont des instances closes des termes, un ensemble de termes qui terminent [62].

Une implantation de notre méthode a également été réalisée en ELAN, pour l'algorithme de base [61]. Contrairement aux autres méthodes de preuve de terminaison, cette approche est prometteuse pour d'autres stratégies, car la relation de réécriture est exprimée explicitement dans le mécanisme de preuve. Une extension est actuellement en cours pour la stratégie *outermost* et des stratégies locales exprimées sur les opérateurs.

6.4.2 Preuves de terminaison

Participants : Isabelle Gnaedig, Hélène Kirchner.

Un survol de plusieurs méthodes permettant de prouver la terminaison de programmes par réécriture ou de montrer des propriétés des ensembles de formes normales est présenté dans [43]. On s'y concentre sur les techniques qui ont été implantées en ELAN et qui sont utiles pour prouver la terminaison de programmes ELAN. Les problèmes abordés sont les suivants : la terminaison générale d'un programme par réécriture, la terminaison *innermost* d'un ensemble de termes clos, le calcul ou l'approximation des ensembles de formes normales, la terminaison d'un programme par réécriture construit de façon modulaire, l'estimation du nombre de formes

normales pour un programme non-déterministe avec stratégies. Pour chaque problème, nous présentons brièvement une méthode et son domaine d'application, donnons des références à d'autres approches existantes, avant de décrire l'implantation qui en est faite en ELAN. L'article conclut sur quelques questions ouvertes. Il a donné lieu à un exposé invité au Workshop on Rewriting Logic and its Applications, qui s'est tenu à Kanazawa en septembre 2000.

6.4.3 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 implantée 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.

6.4.4 Preuves de propriétés observables

Participants : Adel Bouhoula, 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 simplifié notre technique de preuve observationnelle par contextes critiques [BBR98] dans un article soumis. Les possibilités d'augmenter l'automatisation de ce type de preuves sont importantes et l'approche semble offrir une alternative prometteuse à la coinduction.

[BBR98] N. Berregeb, A. Bouhoula, M. Rusinowitch, «Observational Proofs with Critical Contexts», *in : Fundamental Approaches to Software Engineering - ETAPS'98, Lisboa, Portugal, Lecture Notes in Computer Science, 1382*, Springer-Verlag, p. 38–53, avril 1998.

6.5 Vérification

Mots clés : vérification de programme, protocole d'authentification, protocole de télécommunication, système réactif.

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

6.5.1 Structures de données pour la vérification

Participant : Olivier Bournez.

Les algorithmes ou les méthodes de vérification pour les systèmes hybrides sont très souvent basés sur des techniques qui utilisent des calculs de points fixes sur des suites de polyèdres. Il semble que la manipulation de polyèdres non-convexes et de dimension quelconque soit l'un des facteurs importants qui expliquent leur relative «lenteur» et inefficacité sur des exemples de grande taille.

En particulier, la vérification de propriétés des automates temporisés nécessite de manipuler des polyèdres particuliers appelés polyèdres temporisés. En collaboration avec Oded Maler (VERIMAG, Grenoble), nous avons étudié et proposé de nouvelles représentations des polyèdres temporisés adaptées à la vérification des automates temporisés [34].

Ces représentations, basées sur les sommets, étendent celles que nous avions proposées pour les polyèdres orthogonaux. Les algorithmes que nous avions programmés par le passé pour les polyèdres orthogonaux ont donné naissance au laboratoire VERIMAG à un outil expérimental "d/dt" [28], qui permet la vérification de propriété de systèmes hybrides, mais aussi de résoudre des problèmes plus généraux comme le problème de la synthèse de systèmes hybrides [13].

6.5.2 Vérification de protocoles d'authentification

Participants : Yannick Chevalier, Florent Jacquemard, Michaël Rusinowitch, Mathieu Turuani, 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. Nous avons implanté l'algorithme de compilation (appelé CASRUL) en Objective Caml©, et les formules résultantes peuvent être envoyées presque telles quelles à des systèmes de preuves automatiques. Nous utilisons ainsi un mini traducteur (CASDAT) qui convertit le résultat de CASRUL en données directement utilisables par le démonstrateur daTac, car ce dernier 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. Les résultats ont été présentés au Workshop FreGe'00 [42] et sont publiés dans les actes de la conférence LPAR [41].

Nous étudions actuellement comment modifier notre modélisation pour éviter l'unification équationnelle qui est coûteuse. D'autre part nous développons des stratégies pour l'intrus de manière à réduire l'espace de recherche d'une faille. Par exemple seuls les messages acceptables par un récepteur sont engendrés par l'intrus.

Ces travaux vont se poursuivre dans le cadre d'une action concertée incitative Cryptologie (avec le LIM Marseille et l'ENS Cachan). Un projet FET Open a également été soumis sur ce thème avec l'Université de Fribourg (Allemagne) et l'Université de Gênes (Italie).

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 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 80 lemmes et une analyse par cas complexe [48]. Les algorithmes de contrôle de conformité ABR vérifiés par d'autres auteurs sont seulement des approximations de celui que nous traitons.

En utilisant une version de SPIKE étendue par des procédures de décision, nous avons pu montrer automatiquement plus de la moitié des 80 lemmes requis alors que ceux-ci nécessitent des tactiques spécifiques (introduites par l'utilisateur) dans la preuve en PVS.

6.5.4 Automates temporisés et calcul de réécriture

Participants : Olivier Bournez, Hassen Kacem, Claude Kirchner.

Nous nous sommes intéressés à l'étude des liens qui existent entre les algorithmes de *model-checking* pour les automates temporisés et le calcul de réécriture. Emmanuel Beffara dans le cadre de son stage de magistère de l'Ecole Normale Supérieure de Lyon a mis en évidence qu'on pouvait traduire tout automate temporisé en un ensemble de règles du calcul de réécriture équivalentes manipulant un certain type de contraintes stables par élimination des quantificateurs. Ces idées ont été appliquées et étendues par Hassen Kacem pour construire un système basé sur le langage ELAN qui permet de vérifier des propriétés d'atteignabilité de produits d'automates temporisés. Le système semble pouvoir prouver son efficacité sur des exemples concrets.

Ce système est en cours d'extension pour traiter des classes plus générales de systèmes hybrides, et en particulier pour traiter les produits de p-automates comme définis dans le

cadre du projet RNRT CALIFE dans lequel l'équipe est impliquée.

6.5.5 Preuve de systèmes réactifs

Participants : Adel Bouhoula, Julien Musset, Michaël Rusinowitch.

Le comportement de systèmes réactifs décrits à partir de langages de programmation synchrones comme LUSTRE ou SIGNAL, se modélise sous forme de systèmes de transitions discrètes. Nous cherchons alors à prouver automatiquement des propriétés d'invariance. Deux techniques sont généralement utilisées : la vérification de modèles (ou *model-checking*) et les preuves par induction. Ces deux méthodes se réduisent à un calcul de point fixe. Notre objectif est d'améliorer ce calcul à travers des méthodes d'abstractions, qu'elles soient exactes ou approchées. La mise en pratique utilise le logiciel de calcul formel MAPLE.

6.6 Complexité

Mots clés : complexité du calcul, complexité de décision et de comptage, #P-complétude, #·coNP-complétude, indécidabilité..

Résumé : *Nous avons obtenu des résultats sur l'impossibilité d'une méthode de combinaison polynomiale pour les algorithmes d'unification, sur l'indécidabilité des propriétés de stabilité des systèmes hybrides et sur la complexité de la reconnaissance de la base de Hilbert.*

6.6.1 Impossibilité d'une méthode de combinaison polynomiale pour les algorithmes d'unification

Participant : Nicolas Hermann.

Différents problèmes en déduction automatique nous ramènent au problème d'unification équationnelle, qui est le problème de trouver des instances équivalentes pour un ensemble fini de paires de termes, dont l'équivalence est prouvable dans une théorie équationnelle donnée. Il existe souvent des algorithmes d'unification pour des théories équationnelles avec des signatures disjointes, mais aucun algorithme explicite d'unification pour le mélange disjoint de ces théories n'est connu. Au lieu de développer un algorithme d'unification ad-hoc pour chaque cas de mélange, il est préférable d'avoir une méthode générale de combinaison des algorithmes d'unification avec des signatures disjointes. Des méthodes pratiques de combinaison ont été présentées à la fin des années 80 et au début des années 90. Toutes ces méthodes ont le défaut d'avoir une complexité exponentielle. Rappelons que l'algorithme d'unification syntaxique, présenté par Robinson, avait aussi une complexité exponentielle, mais finalement Paterson et Wegman d'une part et Martelli et Montanari d'autre part ont présenté un algorithme d'unification linéaire. La même question se posait pour les méthodes de combinaison, notamment de savoir si les méthodes existantes ont été seulement mal conçues, ou bien si le problème de combinaison possède intrinséquement une complexité supérieure à polynomiale. Ce problème a été résolu dans [21], où nous avons démontré qu'une méthode générale de combinaison en temps polynomial est impossible.

6.6.2 Décidabilité des propriétés de stabilité des systèmes hybrides

Participant : Olivier Bournez.

Depuis les travaux de Moore, il est connu que les systèmes dynamiques à temps discret et à fonction de transition affine par morceaux peuvent simuler les machines de Turing. Par conséquent, il est connu que la vérification des propriétés *locales* comme les propriétés d'atteignabilité des systèmes hybrides est impossible dans le cas général. Toutefois, il restait un espoir que l'on puisse décider les propriétés *globales* de ces systèmes comme les propriétés de stabilité. Ce problème a été posé pour la première fois explicitement par Eduardo Sontag dans le cadre de la théorie du contrôle.

Dans [15], en collaboration avec Vincent Blondel, Pascal Koiran, John Tsitsiklis et Christos Papadimitriou, nous avons pu apporter une réponse partielle à cette question pour les systèmes dynamiques à temps discret et à fonction de transition affine par morceaux *discontinue*.

Avec Vincent Blondel, Pascal Koiran et John Tsitsiklis, nous avons ultérieurement réussi à résoudre en totalité le problème ouvert précis posé par Sontag en prouvant l'indécidabilité des problèmes de la stabilité globale, de la stabilité globale asymptotique et de la mortalité pour les systèmes *continus* dynamiques linéaires seuillés. La preuve de ce résultat a été présentée dans [31], puis légèrement étendue dans [16].

6.6.3 Réduction *substractive* et problèmes complets pour les classes de comptage

Participant : Nicolas Hermann.

En collaboration avec Arnaud Durand et Phokion G. Kolaitis, nous avons introduit et étudié un nouveaux type de réduction entre les problèmes de comptage, intitulée la *réduction substractive* [40]. Nous avons démontré que les classes de complexité de comptage majeures, comme #P ou #NP, ainsi que toutes les classes universelles $\#\cdot\Pi_k\mathrm{P}$ dans la hiérarchie de comptage sont closes par rapport à cette réduction. Ensuite, nous avons étudié des problèmes complets pour ces classes par les réductions *substractives*. Nous nous sommes focalisés en particulier sur la classe #NP (qui est équivalente à la classe $\#\cdot\mathrm{coNP}$) et avons démontré que cette classe contient des problèmes complets intéressants, tels que le problème de compter le nombre des modèles minimaux d'une formule propositionnelle en forme normale conjonctive ou le problème de calculer la cardinalité de l'ensemble des solutions minimales non-triviales d'un système diophantien d'inéquations linéaires homogène.

6.6.4 Base de Hilbert

Participants : Nicolas Hermann, Laurent Juban.

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, les questions de savoir si un ensemble de vecteurs constitue la base de Hilbert d'un système donné ou inconnu sont polynomialement équivalentes [20].

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 2000, nous avons poursuivi le développement du logiciel qui est maintenant en exploitation dans l'application utilisée quotidiennement par le GIHP-Champagne. Les évolutions ont porté d'une part sur un accroissement des performances, qui permet de planifier maintenant 500 courses en moins d'une minute et d'autre part sur l'intégration de nouvelles fonctionnalités. Le système de planification est actuellement en cours d'installation à Amiens et devrait être prochainement installé à St-Brieuc. Par ailleurs, une nouvelle convention de recherche devrait être signée avec le GIHP-Champagne avant fin 2000. Cette convention de recherche porte sur l'intégration de procédures de post-optimisation du planning.

7.2 Application à un problème de transport

Participants : Alexander Bockmayr, Jiany Zhou, Jianyang Zhou.

Dans le cadre d'un projet de création d'entreprise, nous avons mené une étude sur la

[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, mars 1997.

planification de transport de bouteilles de gaz chez Gaz Est Distribution, concessionaire d'Elf Antargaz [64]. Nous avons développé un prototype dans le langage de contraintes NCL [25] pour calculer les fréquences des visites chez les clients, déterminer les tournées des véhicules de transport et planifier l'itinéraire de chaque tournée.

7.3 CALIFE

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

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

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 et France-Télécom R&D.*

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 Coq utilise une technique de preuve par réflection qui permet d'internaliser une procédure de décision 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 régionales

Au niveau du Contrat de Plan Etat-Région 2000-2006, nous sommes impliqués dans le Pôle de Recherche Scientifique et Technologique (PRST) *Intelligence Logicielle* dont Hélène Kirchner assure l'animation.

L'équipe PROTHEO participe au

– Génopôle Strasbourg Alsace-Lorraine

- Thème « Bioinformatique et Applications à la Génomique » du PRST Intelligence Logicielle

- Thème « Qualité et sûreté des logiciels et systèmes informatiques» du PRST Intelligence Logicielle

8.2 Actions nationales

Nous participons aux projets suivants du GDR/PRC ALP, sur les thèmes contraintes et déduction automatique : Collaboration de résolveurs, Langages et Outils pour la Déduction 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.

8.3 Actions européennes

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

Nous participons également au projet européen LISCOS (Large Scale Integrated Supply Chain Optimisation Software). Les partenaires de ce projet qui a commencé le 1/1/2000 et qui est prévu pour une durée de 3 années sont : Barbot (P), BASF (D), CORE (B), COSYTEC (F), Dash (UK), DEIO (P), LORIA (F), PSA (F), Procter and Gamble (B).

8.4 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.

8.5 Relations bilatérales internationales

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.

Un contrat bilatéral du programme Socrates/Erasmus a été signé entre l'UHP et l'Université de Gênes (Italie) pour un échange d'enseignants et des séjours de recherche. Nous participons par ailleurs à un projet PAI Galileo avec l'Université de Gênes (Italie, Prof. Armando) sur l'intégration de procédures de décision au sein d'un prouveur.

Le projet STIC 2000-2 sur « Vérification formelle pour les logiciels de télécommunication» avec l'école Sup'Com de Tunis (Adel Bouhoula) a été retenu pour une durée d'un an renouvelable une fois.

8.6 Accueils de chercheurs étrangers

- Abdessamad Imine, Université d'Oran, a travaillé dans l'équipe sur la validation de spécifications TLA en SPIKE.
- Sofiene Tahar, Université Concordia (Montréal, Canada), a travaillé dans l'équipe sur la vérification du « Fairisle ATM Switch Fabric» avec SPIKE.
- Marian Vittek, Université Comenius (Bratislava, Slovaquie), a travaillé dans l'équipe sur l'intégration de la programmation par règle dans les langages impératifs.

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 et de l'INRIA Lorraine, responsable du projet «Bioinformatique et Applications à la Génomique» du PRST Intelligence Logicielle ; expert du Ministère de la Recherche dans le domaine de la bioinformatique ; membre du comité de coordination de la bioinformatique des génopoles ; membre du conseil scientifique du PRST Intelligence Logicielle ; membre du *Steering Committee* du projet européen LISCOS, associate Editor de INFORMS J. Computing ; co-Responsable de la filière « Algorithmique numérique et symbolique» du DEA Informatique ; membre des comités de programme de CP'2000, CPAIOR'2000, AAAI-2000 Workshop AI-OR Techniques for Combinatorial Optimization.
- Eric Domenjoud : membre de la commission de spécialiste, 27ème section de l'université, de Metz et nouvellement élu au comité national de la recherche scientifique en section 07.
- Nicolas Hermann : Responsable de la partie francaise du projet franco-autrichien AMADEUS sur les schématisations.
- 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 Conseil de Direction du LORIA ; membre de la commission de spécialistes (27ème section) de l'INPL, de l'Université de Metz et de Paris 6 ;

  Membre des comités éditoriaux des journaux : *Journal of Automated Reasoning*, *Journal of Symbolic Computation*, *The Journal of Theory and Practice of Logic Programming*, *Journal of the Egyptian Mathematical Society* et *Journal of Information Science and Engineering* ;

Membre des comités de programmes des conférences « Twenty-Seventh International Colloquium on Automata, Languages and Programming » : ICALP'2000, « Workshop on Implicit Computational Complexity-2000 » : ICC'00, « 2nd International Conference on Principles and Practice of Declarative Programming» : PPDP 2000.

Co-organisateur avec Nachum Dershowitz du « First international workshop on rule based programming » : RULE2000. Organisateur du « INRIA / NSC Taiwan, MultiMedia Workshop ».

Vice président du groupe de travail IFIP WG 1.6 sur la réécriture. Membre du conseil d'administration de CADE-inc (*Board of trustees*). Membre fondateur de IFCoLog[5].

- Hélène Kirchner :

  Animatrice du Pôle de Recherche Scientifique et Technologique Lorrain «Intelligence Logicielle» dans le cadre du Contrat de Plan Etat-Région Lorraine 2000-2006 ; membre du Conseil de Direction du LORIA ; membre du Comité des Projets du LORIA et de l'INRIA-Lorraine ; membre nommé du Conseil de Laboratoire du LORIA ; membre nommé de la Commission de Spécialistes de Nancy 2 ; membre nommé au Comité National du CNRS section 07 depuis septembre 2000.

  Membre du Comité éditorial de la revue *Annals of Mathematics and Artificial Intelligence* ; membre des comités de programmes : AMAST 2000, FROCOS'2000 (co-chair), FM-TOOLS 2000, Calculemus 2000, WRLA'2000 ; membre du *Steering Committee* de la Conférence RTA, responsable d'un contrat avec France Telecom R & D sur la coopération sûre de techniques de preuves (1999-2001.

  Responsable du Groupe Tools du Working Group Esprit CoFI (depuis octobre 98) (Common Framework Initiative for algebraic specification and development of software) ; membre des groupes de travail *ERCIM Constraints* coordonné par K. Apt (CWI, Amsterdam) et *Programming Languages Technology* coordonné par N. Jones (Diku, Copenhague) ; 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 [10] ; co-animateur avec Philippe Devienne du groupe du GDR ALP *Collaboration de Solveurs*.

- Michaël Rusinowitch : membre du groupe IFIP WG 1.6 (Réécriture), membre du *Steering Committee* de FTP2000 (First-Order Theorem-Proving), membre du comité de programme de FTP, membre du comité d'organisation de RTA en 1999-2001, membre du jury du prix de thèse SPECIF, membre du comité d'organisation des Journées Vérification à Orléans (4-5 juin).

- Laurent Vigneron : membre élu du Conseil du laboratoire du LORIA ; membre du comité de rédaction de «La lettre du LORIA» ; représentant de l'équipe PROTHEO à la COMIN (commission des utilisateurs des moyens informatiques), secrétaire du groupe IFIP WG 1.6, administrateur pages web : la « Rewriting Home Page » ; la page de la conférence RTA ; la page du groupe IFIP WG 1.6 ; la page web de l'équipe PROTHEO.

5. `http://www.ifcolog.org`

9.2 Enseignement universitaire

On trouvera ci-dessous la liste des enseignements universitaires effectués par les membres du projet, chercheurs à temps plein.

- Olivier Bournez et Michaël Rusinowitch ont donné un cours de vérification algorithmique dans le cadre du DEA d'Informatique de l'Université Henri Poincaré - Nancy I.

- 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 I.

  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 en IUP GEII RNC 2000-2001, 2ème année, et en DEA d'informatique de l'Université Henri Poincaré - Nancy I, la théorie de l'information et du codage ; en DEA, la complexité des algorithmes ; à l'Université Technique de Vienne, la théorie du codage ; en IUP GEII RNC 1999-2000, 3ème année, l'imagerie numérique (JPEG, MPEG).

- Claude Kirchner a donné un cours de DEA informatique de l'Université Henri Poincaré - Nancy I : « Contraintes et optimisation Combinatoire » avec Alexander Bockmair, « Logique et démonstration automatique » avec Adam Cichon ; un cours invité « Computation and deduction by rewriting with ELAN», à l'école « EEF Foundations school in Deduction and Theorem Proving» à Edinburgh ; un cours à l'Ecole Jeunes Chercheur en Programmation : « Déduction et calcul par réécriture».

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 : Conférences invitées à FROCOS'2000, à l'Université d' Uppsala, à l'Université de Carnegie Mellon.

- Olivier Bournez a obtenu un des deux accessits du prix de thèse 1999 de l'association SPECIF et à cette occasion il a été invité à présenter ses travaux de thèse lors du congrès annuel de l'association en décembre 99. Sa thèse a également été distinguée par l'AFIT (Association Francaise d'Informatique Théorique) et à cette occasion a été invité à présenter ses travaux de thèse lors du congrès annuel de l'association à Lyon en mars 2000.

- Eric Deplagne a donné un exposé : « L'induction comme déduction modulo », lors de la réunion du groupe du GDR ALP « mélange de systèmes de réécriture algébriques et logiques» (Besancon, 29 juin 2000) et dans le cadre du séminaire du projet LogiCal (Coq) (le 13 octobre 2000).

- Nicolas Hermann : exposés au séminaire du laboratoire LAMSADE, Université Paris-Dauphine, le 8 mars 2000, à l'ENS-Lyon, pour les étudiants, sur la complexité descriptive du comptage, le 27 mars 2000, à l'Université d'Aix-la-Chapelle (Aachen), Allemagne, le 8 juin 2000, à l'Université de Würzburg, Allemagne, le 16 octobre 2000.

- Claude Kirchner : Séminaire d'informatique de l'université de Metz, «Déduction Modulo» (19 octobre 2000),

  séminaire à l'université Concordia «Induction as deduction modulo» (18 septembre 2000),

  exposé « ELAN ou la programmation par réécriture », Journées FAC2000 (Formalisation des Activités Concurrentes), Toulouse (Mai 2000),

  présentation d'ELAN et des techniques de réécritures et de filtrage équationels au Workshop « MuPAD » Oberwolfahr (7-9 Novembre 2000),

  séminaire d'évaluation du programme 4B de l'INRIA, Paris (13-14 mars 2000), séminaire d'évaluation du programme 4A de l'INRIA, Nice, (octobre 2000), séminaire Microsoft Research, Cambridge (17-18 avril 2000).

- Hélène Kirchner : Exposé invité au 3rd Int. Workshop on Rewriting Logic and Applications WRLA'2000 septembre 2000, Japon, « Termination and normalisation under strategies–Proofs in ELAN»,

  exposé invité au AMiLP workshop (Algebric Methods in Language Processing) à Iowa City (USA) : «Objects, constraints, rules and strategies in ELAN»,

  exposé invité au Winter Workshop in Logics, Types and Rewriting, Edinburgh. « Algebraic Specifications,Higher-Order Types and Set-Theoretic Models»,

  exposé invité au séminaire franco-allemand à Munich, « Programming with rules and strategies in ELAN».

- Pierre-Etienne Moreau a donné un exposé dans l'équipe Contrainte (F. Fages) à Rocquencourt, dans l'équipe Lande (T. Jensen) à l'Irisa, et au laboratoire Leigniz/IMAG (équipe de R. Echahed).

  Présentation d'ELAN et des techniques de réécritures et de filtrage équationels au Workshop « MuPAD » Oberwolfahr (7-9 Novembre 2000).

- Michaël Rusinowitch a été conférencier invité au colloque LPAR (St-Gilles, Réunion, Novembre 2000). Il 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 :

RTA'2000 ; 2ème workshop IFIP, à Norwich (UK) ; Westapp2000, Norwich (UK) ; workshop on Implementations of Logic, La Réunion ; colloque International Conference on Computational Logic a Londres ; Workshop MuPad ; RULE2000, Montréal ; PPDP2000, Montréal ; FREGE'2000, Fribourg.

9.3.2 Séjours de chercheurs

- Alexander Bockmayr a effectué une visite à l'université de Carnegie Mellon (3 jours en août 2000).
- Olivier Bournez a effectué plusieurs séjours au laboratoire VERIMAG de Grenoble.
- Claude Kirchner a fait un séjour à l'université de Tel-Aviv. Il a donné un séminaire « 2 + 2 = 4: Shall we prove it or ... ?» (4-11 mai 2000).

9.4 Jurys de thèses et jurys divers

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

- Alexander Bockmayr :

  G. Ottosson (Uppsala), F. Eisenbrand (Sarrebruck).

- Claude Kirchner :

  Horatiu Cirstea, « Calcul de Réécriture : Fondements et Applications », UHP, octobre 2000, directeur de thèse,

  Anne Liret, « Intégration de mécanismes de réécriture dans un système de satisfaction de contraintes », UP6, octobre 2000, rapporteur,

  François Cuny, « Radiosité à base d'ondelettes sur des surfaces paramétriques », INPL, octobre 2000, rapporteur,

  Jean-Yves Marion, HDR : « Complexité implicite des calculs, de la théorie à la pratique», Université Nancy 2, décembre 2000, rapporteur,

  Dominique Colnet, HDR : «Compilation, langages à objets et programmation par contrats», UHP, décembre 2000, rapporteur,

  Pierre Gradit, «Spécification et conceptions d'applications coopératives», UPS, décembre 2000, rapporteur.

- Hélène Kirchner :

  Vincent Schachter, « Programmation concurrente avec contraintes fondée sur la logique linéaire », Université de Paris Sud, Orsay, décembre 1999, rapporteur,

  Lilian Burdy, « Un traitement des expressions dépourvues de sens dans la théorie des ensembles : application à la méthode B », CNAM Paris, mai 2000, rapporteur,

  Catherine Dubois, HDR « Un parcours de la programmation à la preuve », Evry, janvier 2000, examinateur,

Bernhard Gramlich, HDR, « Towards a structure theory of term rewriting», Vienne, mars 2000, rapporteur.

– Michaël Rusinowitch :

J.-C. Janodet, « Réécriture et surréduction des graphes admissibles», INPG, janvier, rapporteur,

D. Larchey-Wendling, « Preuves, réfutations, contre-modèles dans des logiques intuitionnistes », UHP, décembre, rapporteur,

S. Stratulat, « Preuves par récurrence avec ensembles couvrants contextuels. Application à la vérification de logiciels de télécommunications», UHP, novembre, directeur de thèse.

Isabelle Gnaedig et Pierre-Etienne Moreau ont participé aux jurys du concours d'entrée à l'ESIAL.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] V. Blondel, O. Bournez, P. Koiran, J. Tsitsiklis, « The stability of saturated linear dynamical systems is undecidable», *Journal of Computer and System Science*, 2000.

[2] A. Bockmayr, T. Kasper, « Branch-and-Infer: A unifying framework for integer and finite domain constraint programming», *INFORMS J. Computing 10*, 3, 1998, p. 287 – 300.

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

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

[5] 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.

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

[7] H. Kirchner, P.-E. Moreau, « Promoting Rewriting to a Programming Language: A Compiler for Non-Deterministic Rewrite Programs in Associative-Commutative Theories », *Journal of Functional Programming*, 2000, à paraître.

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

[9] 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

[10] Hélène Kirchner, Christophe Ringeissen, *Frontiers of Combining Systems*, *Lecture Notes in Artificial Intelligence*, *1794*, Berlin Heidelberg New York, Springer-Verlag, mars 2000.

Thèses et habilitations à diriger des recherches

[11] H. Cirstea, *Calcul de réécriture : fondements et applications*, Thèse d'université, Université Henri Poincaré - Nancy I, octobre 2000.

[12] S. Stratulat, *Preuves par récurrence avec ensembles couvrants contextuels. Application à la vérification de logiciels de télécommunications*, Thèse d'université, Université Henri Poincaré, Nancy I, novembre 2000.

Articles et chapitres de livre

[13] E. Asarin, O. Bournez, T. Dang, O. Maler, A. Pnueli, «Effective Synthesis of Switching Controllers for Linear Systems», *Proceedings of the IEEE, Special Issue on "Hybrid Systems"*, 2000.

[14] E. Astesiano, M. Bidoit, H. Kirchner, B. Krieg-Brückner, P. Mosses, D. Sanella, «CASL: The Common Algebraic Specification Language», *Theoretical Computer Science*, 2000.

[15] V. D. Blondel, O. Bournez, P. Koiran, C. Papadimitriou, J. N. Tsitsiklis, «Deciding stability and mortality of piecewise affine dynamical systems», *Theoretical Computer Science A*, 2000.

[16] V. D. Blondel, O. Bournez, P. Koiran, J. Tsitsiklis, «The stability of saturated linear dynamical systems is undecidable», *Journal of Computer and System Science*, 2000, à paraître.

[17] P. Borovanský, C. Kirchner, H. Kirchner, C. Ringeissen, «Rewriting with strategies in ELAN: a functional semantics», *International Journal of Foundations of Computer Science*, 2001.

[18] G. Dowek, T. Hardin, C. Kirchner, «Higher-order unification via explicit substitutions», *Information and Computation 157*, 1/2, 2000, p. 183–235.

[19] G. Dowek, T. Hardin, C. Kirchner, «HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic», *Mathematical Structures in Computer Science 11*, 1, 2001, p. 1–25.

[20] A. Durand, M. Hermann, L. Juban, «On the Complexity of Recognizing the Hilbert Basis of a Linear Diophantine System», *Theoretical Computer Science*, 2000.

[21] M. Hermann, P. G. Kolaitis, «Unification algorithms cannot be combined in polynomial time», *Information and Computation 162*, 1/2, 2000, p. 24–42.

[22] H. Kirchner, P.-E. Moreau, «Promoting Rewriting to a Programming Language: A Compiler for Non-Deterministic Rewrite Programs in Associative-Commutative Theories», *Journal of Functional Programming*, 2000, à paraître.

[23] S. Stratulat, «A General Framework to Build Contextual Cover Set Induction Provers», *Journal of Symbolic Computation*, 2001.

[24] R. Verma, M. Rusinowitch, D. Lugiez, «Decision Problems in Ordered Rewriting», *Fundamenta Informatica*, 2000.

[25] J. Zhou, « Introduction to the constraint language NCL », *Journal of Logic Programming 45*, 1-3, 2000, p. 71–103.

Communications à des congrès, colloques, etc.

[26] C. Alvarado, Q.-H. Nguyen, « ELAN for equational reasoning in Coq », *in : Proceeding of 2nd Workshop on Logical Frameworks and Metalanguages*, J. Despeyroux (éditeur), juin 2000. Santa Barbara (California), USA.

[27] A. Armando, M. Rusinowitch, S. Stratulat, « Incorporating Decision Procedures in Implicit Induction », *in : 8th International Conference on Computer Aided Systems Theory. Eurocast'2001*, février 2001.

[28] E. Asarin, O. Bournez, T. Dang, O. Maler, « Approximate reachability analysis of piecewise-linear dynamical systems », *in : Hybrid Systems : Computation and Control, Pittsburgh, PA, USA*, *Lecture Notes in Computer Science*, *1790*, Springer, p. 20–31, mars 2000.

[29] L. Bachmair, I. V. Ramakrishnan, A. Tiwari, L. Vigneron, « Congruence Closure Modulo Associativity and Commutativity », *in : Proceedings of 3rd International Workshop on Frontiers of Combining Systems*, H. Kirchner, C. Ringeissen (éditeurs), *Lecture Notes in Computer Science*, *1794*, Springer, p. 245–259, Nancy (France), mars 2000.

[30] L. Bachmair, A. Tiwari, L. Vigneron, « Abstract Congruence Closure for Normalization », *in : First International Workshop FreGe 2000*, Genova (Italy), février 2000.

[31] V. D. Blondel, O. Bournez, P. Koiran, J. N. Tsitsiklis, « The stability of saturated linear dynamical systems is undecidable », *in : Symposium on Theoretical Aspects of Computer Science (STACS), Lille, France*, H. R. S. Tison (éditeur), *Lecture Notes in Computer Science*, *1770*, LIFL, Springer, p. 479–490, février 2000.

[32] A. Bockmayr, Y. Dimopoulos, « Integer programs and valid inequalities for planning problems », *in : 5th European Conference on Planning, ECP'99, Durham, UK*, M. F. S. Biundo (éditeur), *LNAI*, *1809*, Springer, Berlin Heidelberg, 2000.

[33] A. Bockmayr, F. Eisenbrand, « Combining logic and optimization in cutting plane theory », *in : Frontiers of Combining Systems, FroCoS'2000, Nancy*, H. Kirchner, C. Ringeissen (éditeurs), *LNAI*, *1794*, Springer, p. 1–17, Berlin, Heidelberg, mars 2000.

[34] O. Bournez, O. Maler, « On the representation of timed polyhedra », *in : International Colloquium on Automata Languages and Programming (ICALP'00)*, p. 793–807, Geneva, Switzerland, juillet 2000.

[35] H. Cirstea, C. Kirchner, « Rewriting and Multisets in the Rewriting Calculus and ELAN », *in : Workshop on Multiset Processing*, août 2000, `http://www.loria.fr/publications/2000/A00-R-238/A00-R-238.ps`.

[36] H. Cirstea, C. Kirchner, « The simply typed rewriting calculus », *in : 3rd International Workshop on Rewriting Logic and its Applications - WRLA2000*, septembre 2000, `http://www.loria.fr/publications/2000/A00-R-237/A00-R-237.ps`.

[37] E. Deplagne, « Sequent Calculus Viewed Modulo », *in : ESSLLI2000 Student Session, Birmingham, angleterre*, C. Pilière (éditeur), FoLLI, University of Birmingham, août 2000, `http://www.loria.fr/publications/2000/A00-R-256/A00-R-256.ps`.

[38] H. Dubois, H. Kirchner, «Objects, rules and strategies in ELAN», *in : Algebraic Methods in Language Processing (AMILP 2000), Iowa City, Iowa, USA*, mai 2000, `http://www.loria.fr/publications/2000/A00-R-246/A00-R-246.ps`.

[39] H. Dubois, H. Kirchner, «Rule Based Programming with Constraints and Strategies», *in : Joint ERCIM/Compulog Net Workshop. Selected Papers, Paphos, Cyprus, October 1999*, K. R. Apt, A. C. Kakas, E. Monfroy, F. Rossi (éditeurs), *Lecture Notes in Artificial Intelligence*, *1865*, Springer-Verlag, p. 274–297, mai 2000.

[40] A. Durand, M. Hermann, P. G. Kolaitis, «Subtractive Reductions and Complete Problems for Counting Complexity Classes», *in : 25th International Symposium on Mathematical Foundations of Computer Science (MFCS 2000), Bratislava (Slovaquie)*, M. N. et B. Rovan (éditeur), *Lecture Notes in Computer Science*, *1893*, Springer, p. 323–332, août 2000.

[41] F. Jacquemard, M. Rusinowitch, L. Vigneron, «Compiling and Verifying Security Protocols», *in : Logic for Programming and Automated Reasoning*, M. Parigot, A. Voronkov (éditeurs), *Lecture Notes in Computer Science*, *1955*, Springer, p. 131–160, novembre 2000.

[42] F. Jacquemard, M. Rusinowitch, L. Vigneron, «Narrowing Semantics for Cryptographic Protocols and Verification», *in : First International Workshop FreGe 2000*, Genova (Italy), février 2000.

[43] H. Kirchner, I. Gnaedig, «Termination and normalisation under strategy - Proofs in ELAN», *in : Third International Workshop on Rewriting Logic and Applications, WRLA'2000*, *Elecronic Notes In Theoretical Computer Science*, Elsevier, Kanazawa, JAPAN, septembre 2000.

[44] P.-E. Moreau, « REM (Reduce Elan Machine) : Core of the New ELAN Compiler », *in : Rewriting Techniques and Applications (RTA), Norwich*, L. Bachmair (éditeur), *LNCS*, 1833, Richard Kennaway, Springer, p. 265–269, juillet 2000.

[45] P. Narendran, M. Rusinowitch, « The theory of total unary rpo is decidable », *in : First International Conference on Computational Logic*, J. L. et al. (éditeur), *Lecture Notes in Computer Science*, *1861*, Springer, p. 660–672, juillet 2000.

[46] C. Ringeissen, E. Monfroy, «Generating Propagation Rules for Finite Domains : a Mixed Approach», *in : Joint ERCIM/Compulog Net Workshop, Paphos, Cyprus*, *Lecture Notes in Artificial Intelligence*, *1865*, Springer-Verlag, p. 150–172, 2000.

[47] C. Ringeissen, « Handling Relations over Finite Domains in the Rule-Based System ELAN», *in : Third International Workshop on Rewriting Logic and Applications, WRLA'2000*, *Elecronic Notes In Theoretical Computer Science*, Elsevier, Kanazawa, JAPAN, septembre 2000.

[48] M. Rusinowitch, S. Stratulat, F. Klay, «Mechanical Verification of an Ideal ABR Conformance Algorithm», *in : Conference on Computer Aided Verification*, E. A. Emerson, A. P. Sistla (éditeurs), *Lecture Notes in Computer Science*, *1770*, Springer, juillet 2000.

[49] J. Stuber, «Deriving Theory Superposition Calculi from Convergent Term Rewriting Systems», *in : Rewriting Techniques and Applications (RTA), Norwich*, L. Bachmair (éditeur), *LNCS*, *1833*, Richard Kennaway, Springer, p. 229–245, juillet 2000.

[50] M. van den Brand, C. Ringeissen, «ASF+SDF parsing tools applied to ELAN», *in : Third International Workshop on Rewriting Logic and Applications, WRLA'2000*, *Elecronic Notes In Theoretical Computer Science*, Elsevier, Kanazawa, JAPAN, septembre 2000.

Rapports de recherche et publications internes

[51] A. Aggoun, M. Beauseigneur, A. Bockmayr, N. Pisaruk, « LISCOS Growth Project : Modelling and requirements of PSA test cases », *Rapport intermédiaire*, octobre 2000.

[52] A. Aggoun, A. Bockmayr, Y. Colombani, S. Heipcke, A. Miller, N. Pisaruk, Y. Pochet, M. Van Vyve, L. Wolsey, «LISCOS Growth Project : Manual of modelling conventions with examples», *Rapport intermédiaire*, octobre 2000.

[53] A. Aggoun, A. Bockmayr, Y. Colombani, S. Heipcke, A. Miller, N. Pisaruk, Y. Pochet, L. Wolsey, « LISCOS Growth Project : Initial report on generic supply chain models», *Rapport intermédiaire*, octobre 2000.

[54] A. Bockmayr, F. Eisenbrand, « Cutting Planes and the Elementary Closure in Fixed Dimension », *Rapport de recherche*, Max-Planck-Institut für Informatik, Saarbrücken, Germany, décembre 1999, Numéro du rapport MPI-I-1999-2-008.

[55] A. Bockmayr, N. Pisaruk, « The convex hull of the disjunction of polymatroids », *Rapport interne*, août 2000.

[56] P. Borovanský, H. Cirstea, H. Dubois, C. Kirchner, H. Kirchner, P.-E. Moreau, C. Ringeissen, M. Vittek, ELAN *V 3.4 User Manual*, édition fourth, LORIA, Nancy (France), janvier 2000.

[57] H. Cirstea, C. Kirchner, L. Liquori, «A Rho Cube», *Rapport de recherche*, novembre 2000.

[58] H. Cirstea, C. Kirchner, L. Liquori, «Matching Power», *Rapport de recherche*, novembre 2000.

[59] H. Cirstea, C. Kirchner, «The rewriting calculus», *Research report*, LORIA, 2000.

[60] H. Dubois, H. Kirchner, « Rules, strategies and objects in ELAN », *Rapport de recherche*, novembre 2000, `http://www.loria.fr/publications/2000/A00-R-245/A00-R-245.ps`.

[61] O. Fissore, « Terminaison par Induction », *Mémoire de DEA, Université Henri Poincaré - Nancy I*, juin 2000.

[62] I. Gnaedig, H. Kirchner, O. Fissore, « Induction for Termination », *rapport de recherche*, LORIA, Nancy (France), 2000.

[63] H. Kacem, « Implantation de l'algorithme de Stalmarck dans le cadre de la coopération COQ/ELAN», *Stage de fin d'etudes*, INRIA-Lorraine, juillet 2000.

[64] J. Zhou, A. Bockmayr, J. Zhou, «Préétude pour la planification de transport de Gaz Est Distribution», *Rapport de fin de contrat*, mai 2000.