INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet PROTHEO

Contraintes, Déduction automatique et Preuves de Propriétés de Logiciels

Lorraine

THÈME 2A

Table des matières

1. Composition de l'équipe

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

Responsable scientifique
Claude Kirchner [DR INRIA]

Responsable permanent
Michaël Rusinowitch [DR INRIA]

Assistante de projet
Christelle Bergeret-Etienne [TR INRIA, jusqu'au 31/05/2001]
Sophie Drouot [à partir du 30/04/2001]

Personnel INRIA
Olivier Bournez [CR]
Isabelle Gnaedig-Antoine [CR]
Christophe Ringeissen [CR]
Pierre-Etienne Moreau [CR]

Personnel CNRS
Hélène Kirchner [DR, à temps partiel]

Personnel Université
Laurent Vigneron [Maître de Conférences, Université Nancy 2]
Carlos Castro [Maître de Conférence associé, UHP, 4 mois]

Spécialiste étranger INRIA
Mark van den Brand [à partir du 1/11/2001]

Ingénieurs associés INRIA
Mohamed-Medhi Bouallagui [à partir du 1/09/2001]
Hassen Kacem [jusqu'au 31/08/2001]

Chercheurs doctorants
Tarek Abbes [BDI-PED, à partir du 1/10/2001]
Yannick Chevalier [ATER]
Alberto Ciaffaglione [bourse Italienne, en cotutelle INPL–université d'Udine, à partir du 1/04/2001]
Eric Deplagne [allocataire MNERT, puis ATER à partir du 1/09/2001]
Hubert Dubois [allocataire MNERT, jusqu'au 31/08/2001]
Liliana Ibănescu [contrat Peugeot]
Olivier Fissore [boursier BDI CNRS — Région Lorraine]
Paulin Jacobé de Naurois [Normalien, à partir du 1/10/2001, Thèse co-encadrée par les projets CALLIGRAMME et PROTHEO]
Julien Musset [boursier DGA]
Quang-Huy Nguyen [boursier INRIA]
Silvio Ranise [ATER, à partir du 1/10/2001]
Mathieu Turuani [allocataire MNERT]

Chercheurs post-doctorants
Horatiu Cirstea [ATER, à partir du 1/09/2001]
Hubert Dubois [ATER, à partir du 1/09/2001]
Zhebin Qian [Post-doc MNERT, jusqu'au 31/07/2001]
Jürgen Stuber [Post-doc INRIA]

Chercheurs invités
Mitch Harris [du 1/05/2001 au 31/07/2001]

Anamaria Martins Moreira [du 26/03/2001 au 6/04/2001]
Paliath Narendran [du 29/06/2001 au 29/07/2001]
Juan Ortega [du 1/07/2001 au 31/08/2001]
Rakesh Verma [du 16/06/2001 au 15/07/2001]
Eelco Visser [du 22/01/2001 au 25/01/2001]

Stagiaires

Djalel Abdemouche [DEA Nancy I, 8 mois]
Anouar Ben Hmida [Univ. Tunis, 4 mois]
Mohamed-Medhi Bouallagui [ENSI Tunis, 4 mois]
Mohamed Elhabib [Univ. Tunis, 4 mois]
Germain Faure [ENS Lyon, 4 mois]
Laurent Fousse [1 mois et demi]
Walid Gaaloul [ENSI Tunis, 4 mois]
Guillaume Ignatio [ESSTIN Nancy, 2 mois]
Nishant Pentapalli [2 mois]
Nicolas Robert [ESSTIN Nancy, 3 mois]
Preet Kamal Singh [IIT Delhi, 2 mois et demi]

2. Présentation et objectifs généraux du projet

2.1. (Sans titre)

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 essentiellement à la résolution d'équations et de contraintes sur les termes et à la collaboration de résolveurs de contraintes se combinant sur des domaines différents ou coopérant entre eux sur un même domaine.

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.*

Nous étudions la satisfaisabilité et la résolution de systèmes de contraintes, essentiellement sur des domaines symboliques à base de termes. Nous cherchons à combiner des techniques de résolutions, à faire collaborer des résolveurs de contraintes ainsi qu'à résoudre des mélanges de contraintes de différentes formes. 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 des domaines de 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.*

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

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

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, les preuves de propriétés de tels programmes, comme la terminaison. 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.

3.3. Démonstration automatique

Mots clés : *déduction, réécriture, récurrence, contrainte, paramodulation, résolution.*

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'application

4.1. (Sans titre)

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 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 6.7). A plus long terme, nous voulons appliquer nos techniques à la modélisation de systèmes réactifs et de systèmes physiques hybrides, c'est-à-dire ayant des comportements discrets et continus.

5. Logiciels

5.1. CASRUL

Mots clés : *Protocoles cryptographiques, vérification automatique*.

Participants : Mohamed-Mehdi Bouallagui, Yannick Chevalier, Michaël Rusinowitch [correspondant], Mathieu Turuani, Laurent Vigneron.

CASRUL est un système de vérification automatique de protocoles cryptographiques. Cet outil, écrit en Objective Caml, 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.

D'autre part, **CASRUL** est utilisé dans le cadre du projet européen AVISS par nos collègues de Fribourg et de Gênes pour la vérification de protocoles à l'aide de méthodes basées respectivement sur le model-checking et SAT.

CASRUL 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 : Mohamed-Mehdi Bouallagui, Yannick Chevalier, 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, en logique du premier ordre. 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** n'a pas évolué cette année, mais il est utilisé intensivement dans l'équipe pour vérifier des protocoles cryptographiques.

daTac est documenté, maintenu, diffusé par ftp et accessible sur le Web (version 0.92).

5.3. ELAN

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

Participants : Mark van den Brand, Horatiu Cirstea, Hubert Dubois, Olivier Fissore, Hassen Kacem, 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 [BCD+00]. La distribution est exécutable sur les architectures DEC-ALPHA, SUN4 et Linux-Intel.

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

Depuis avril 2001, nous distribuons également l'outil de compilation du filtrage TOM, qui est le premier élément d'une future « boite à outils » pour **ELAN**.

5.4. SPIKE

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

Participant : Michaël Rusinowitch [correspondant].

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 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 est documenté, diffusé par ftp et accessible sur le Web.

6. Résultats nouveaux

6.1. Réécriture et stratégies

Mots clés : *réécriture, stratégie, compilation, analyse syntaxique, règle de production.*

Nous avons développé notre travail autour du calcul de réécriture qui a fait l'objet de la thèse d'Horatiu Cirstea [Cir00], thèse distinguée cette année par l'AFIT (association française d'informatique théorique). Le calcul et ses applications ont également été présentés en conférence invitée aux journées AFDL [Kir01].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. Le ρ-calcul - calcul de réécriture

Participants : Horatiu Cirstea, Germain Faure, Claude Kirchner.

A partir de nos travaux précédents [BKKR01], nous avons remarqué que l'outil nécessaire pour contrôler la réécriture doit être explicite et peut être décrit tout naturellement en utilisant la réécriture elle-même. Cette observation nous a amenés à un nouveau calcul généralisant le λ-calcul et la réécriture que nous avons appelé le ρ-calcul ou le calcul de réécriture [CK01b][CK01c].

Dans ce calcul, l'opérateur d'abstraction ainsi que l'opérateur d'application sont des objets du calcul. Une ρ-abstraction est une règle de réécriture dont le membre gauche précise les variables abstraites et une information de contexte. Le résultat de l'évaluation d'une application (d'une ρ-abstraction ou d'un ρ-terme plus élaboré) est un ρ-terme structuré (*e.g.* un ensemble). Le mécanisme permettant d'instancier les variables avec leur valeur actuelle est le filtrage qui peut être syntaxique, équationnel ou d'ordre supérieur. En effet, nous avons introduit comme paramètre du ρ_T-calcul la théorie T décrivant le comportement des symboles de la signature.

La représentation des résultats au niveau objet du calcul permet la construction des règles de réécriture avec un ensemble comme membre droit et donc, avec un comportement explicitement non-déterministe. Ainsi, un ensemble vide de résultats représente un échec de filtrage et un ensemble ayant plus d'un élément représente tous les résultats d'une application non-déterministe. Nous avons également introduit un nouveau calcul dans lequel l'échec est distingué de l'ensemble vide et dans lequel le test de l'échec est possible. Si nous munissons ce calcul de l'appel par valeur, nous obtenons un calcul confluent et dans lequel le *first* est exprimable (et par conséquent les stratégies d'évaluation le sont aussi) [Fau01].

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. En partant de la représentation des règles de réécriture conditionnelles nous avons montré comment le ρ-calcul peut être employé pour donner une sémantique à l'application des règles du langage **ELAN** [CK01a].

Nous avons utilisé les ensembles comme moyen de représenter le non-déterminisme mais nous avons réalisé avec Luigi Liquori l'étude d'une autre description du ρ-calcul ayant comme paramètre non seulement la théorie de filtrage mais aussi la construction utilisée pour structurer les résultats. Nous avons ainsi obtenu un calcul plus flexible et nous avons déjà montré son pouvoir d'expressivité [CKL01a]. Plus précisément, nous avons analysé la correspondance entre le ρ-calcul et deux calculs qui ont fortement influencé la recherche dans le domaine des paradigmes orientés objets : le *« Object Calculus »* de Abadi et Cardelli [AC96] et le *« Lambda Calculus of Objects »* de Fisher, Honsell et Mitchell [FHM94].

Nous avons étudié une version simplement typée du calcul de réécriture [CK00] mais compte tenu de la puissance d'expression de l'abstracteur « $\longrightarrow$ », 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 proposer en

collaboration avec Luigi Liquori, un système de types basé sur l'unification complète des deux opérateurs « λ » et « Π » dans le seul symbole d'abstraction utilisé dans le ρ-calcul, *i.e.* l'opérateur « $\longrightarrow$ ». Cette unification est *built-in* dans la définition du ρ-calcul lui-même. Par conséquent, nous avons présenté et expérimenté une collection de neuf ($8 + 1$) systèmes de types pour le ρ-calcul décoré à la Church, qui peuvent être représentés dans un ρ-cube qui étend le λ-cube de Barendregt [CKL01b].

6.1.2. Compilation et intégration de règles de réécriture

Participants : Hélène Kirchner, Pierre-Etienne Moreau, Christophe Ringeissen.

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. Nous avons étudié des techniques de compilation qui améliorent l'efficacité de ce type de langages et plus particulièrement les algorithmes, à base d'automates déterministes, pour compiler efficacement le filtrage de termes [KM01a].

En collaboration avec Marian Vittek, maintenant à l'université de Bratislava, nous avons proposé un nouveau formalisme permettant d'intégrer des constructions de filtrage dans des langages tels que C, Java ou Eiffel. Ce nouveau formalisme, présenté dans [MRV01], fait l'objet d'un implantation diffusée : **TOM**. Ce logiciel peut être vu comme un préprocesseur capable de compiler des constructions de filtrage. Une de ses particularités est d'être indépendant du langage cible : **TOM** peut générer des automates de filtrages dans différents langages, dont C, Java et Eiffel. Une autre caractéristique essentielle est d'être indépendant de la représentation des termes. Ces caractéristiques font de **TOM** un outil idéal pour compiler tout langage fondé sur la notion de règles de réécriture (ASF+SDF, **ELAN** et Stratego par exemple).

6.1.3. Réécriture et objets

Participants : Hubert Dubois, Hélène Kirchner.

Dans sa thèse [Dub01], Hubert Dubois a proposé une extension objet du langage **ELAN**. Cette extension permet de définir des classes et des objets en **ELAN**. Elle permet aussi d'écrire des règles dont les membres gauches et droits sont des ensembles d'objets. Nous nous sommes intéressés aux aspects sémantiques en montrant que cette extension objet du langage **ELAN** respecte la sémantique donnée par le ρ-calcul, en s'appuyant sur une version particulière du calcul de réécriture, le ρ-calcul objet [CKL01a].

Nous avons défini pour chaque classe et pour chaque objet, une représentation algébrique, donnée par une signature et des règles. Une correspondance établie entre ρ-termes et termes représentant des objets, et une correspondance entre l'évaluation par réécriture de ces termes et l'évaluation dans le ρ-calcul nous ont permis de montrer qu'un sous-ensemble du ρ-calcul donne une sémantique opérationnelle à l'extension objet de **ELAN**. De plus, des règles de typage des objets ont été définies et la propriété de préservation des types a été établie. L'implantation de ce langage a été réalisée via l'écriture d'un programme **ELAN** qui compile les programmes écrits dans l'extension objet en des programmes **ELAN**.

Suite aux travaux sur la définition d'un formalisme de règles travaillant à la fois avec une base d'objets et avec une base de contraintes, nous avons également étendu les règles sur les objets pour prendre en compte des variables contraintes.

6.1.4. Réécriture de structures XML

Participants : Claude Kirchner, Zhebin Qian, Preet Kamal Singh, Jürgen Stuber.

Nous avons etudié la relation entre XSLT (XML Stylesheet Language Transformations) et la réécriture. XSLT est un nouveau standard de W3C pour la transformation des documents en format XML qui a été finalisé en octobre 2001. Il est décrit par un document d'environ 70 pages en anglais qui repose sur d'autres standards (XML, XML Namespaces, XPath) dont la description est donnée dans des documents de taille similaire. Les documents XML que transforme XSLT sont des arbres labelisés qui sont très proches des termes utilisés en réécriture, mais où en général l'arité des symboles est variable.

Pour mieux comprendre XSLT nous avons d'abord donné une spécification par règles d'inférence d'un noyau de XSLT, et nous avons implanté ce noyau en **ELAN**. Cela a mis en évidence les différences fondamentales entre la réécriture de terme et les transformations décrites par des règles XSLT.

Nous avons ainsi écrit un interpréteur qui comprend non seulement ce noyau XSLT mais aussi des spécifications de XML et de XPath assez complètes. Un détail remarquable est qu'en **ELAN** la specification XML est très courte, une vingtaine de lignes environ. Il faut cependent noter que le parseur lexical d'**ELAN** présente quelquels limitations qui entrainent des differences superficielles entre la syntaxe XML et la syntaxe accepté par notre interpreteur. Nous avons testé ce prototype sur des documents de taille moyenne tels qu'un catalogue de 300 *packages* TeX. Les tranformations se font en quelques secondes [KQSS01].

6.1.5. *Tactiques de réécriture dans les assistants au développement de preuve*

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

Dans le cadre de la coopération **Coq**-**ELAN** pour automatiser et améliorer les performances des preuves par réécriture en **Coq**, nous avons développé en collaboration avec Cuihtlauac Alvarado à France Télécom R&D une interface permettant de normaliser les termes dans **Coq** en utilisant **ELAN** comme un oracle. Dans cette interface, **Coq** rejoue la trace des réécritures retournée par **ELAN** en utilisant la technique de preuve par réflection afin d'avoir la forme normale. L'interface est disponible sur le site Web du projet `CALIFE`.

Naturellement, l'étape suivante de notre travail a été d'étendre l'interface afin de traiter la réécriture modulo Associativité et Commutativité (AC) en **Coq**. Cependant, cette extension n'est pas triviale car les informations gardées dans la trace ne suffisent pas à rejouer le calcul en **Coq**. En effet, le filtrage modulo AC peut retourner plus d'une solution et de plus, cette opération de filtrage n'est pas une tâche simple à refaire efficacement en Coq[1]. Autrement dit, il faut que la trace retournée par **ELAN** contienne la substitution utilisée à chaque pas de réécriture.

D'autre part, **ELAN**, comme la plupart d'autres systèmes traitant la réécriture modulo AC, aplatit et met un terme en forme canonique avant de le réduire. La forme canonique est construite en se basant sur un ordre total entre les termes clos. De ce fait, la position d'un radical devient dépendante de cet ordre et donc, si l'on souhaite utiliser cette information pour retrouver le radical en **Coq**, il faut implanter le même ordre sur les termes **Coq**. Cette solution est évidemment coûteuse et de plus, amène à créer une dépendance de code entre les deux systèmes.

Dans ces circonstances, nous avons opté pour garder dans la trace générée par **ELAN** le *contexte* de chaque pas de réécriture au lieu de la position du radical. La trace d'un pas de réécriture se compose donc, du *contexte*, de la *substitution* et de la *règle appliquée*. Le calcul de réécriture avec substitutions explicites ($\rho\sigma$-calcul) a été utilisé pour formaliser cette trace qui peut être considérée maintenant comme le terme de preuve par réécriture dans **ELAN**. Ce formalisme facilite les traitements syntaxiques sur les termes de preuve comme de les rendre plus compacts ou de les traduire dans une autre syntaxe par exemple, celle de **Coq**. En pratique, une telle traduction pourrait servir à certifier la réécriture effectuée en **ELAN** et aussi à « rejouer » la réécriture en **Coq** sans utiliser la réflection. Nous avons implanté ce mécanisme et obtenu une nouvelle tactique permettant de faire de la réécriture multi-sortée au premier ordre (syntaxique ou bien modulo AC) en **Coq**.

Une description préliminaire de ce travail est présentée dans [Ngu01a].

6.1.6. *Réécriture paresseuse pour la preuve*

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

Un problème fondamental que nous avons rencontré dans l'étude de l'interface **Coq**-**ELAN** est de minimiser le temps nécessaire pour calculer la forme normale en **Coq**. Ce temps dépend fortement de la longueur de la dérivation et aussi de la position des radicaux contractés. Les radicaux les plus proches de la racine du terme sont de préférence réduits d'abord car ces réductions sont moins coûteuses et génèrent des termes de preuve plus courts. Ce fait nous amène à rechercher des stratégies de réduction mieux adaptées à notre interface.

À la suite des travaux de Fokkink, Kamperman et Walters dans [FKW00] sur la simulation de la réécriture paresseuse par la réécriture *innermost* dans un autre système de réécriture, nous proposons une procédure de normalisation par réécriture paresseuse. En fait, la réécriture paresseuse permet de réduire les sous-termes actifs d'un terme en leur forme normale de tête. Les sous-termes paresseux, par contre, peuvent contenir un

[1] Sa complexité est simplement exponentielle dans le cas général.

radical. Nous proposons de réactiver récursivement ces sous-termes et de continuer la normalisation jusqu'au moment où tous les sous-termes sont en forme normale de tête. Le terme intégral est alors en forme normale. Puisque les sous-termes actifs sont les plus proches de la racine du terme, notre procédure réduit souvent des radicaux les plus extérieurs. En outre, grâce aux restrictions imposées par la réécriture paresseuse, les radicaux paresseux sont réduits « par nécessité ». En choisissant une annotation paresseuse appropriée sur les arguments des symboles de la signature, nous pouvons obtenir une dérivation plus courte aboutissant à la forme normale.

Par ailleurs, nous avons établi la correspondance entre la dérivation innermost simulant la réécriture paresseuse et la dérivation jusqu'à la forme normale dans le système de réécriture originel. Grâce à cette correspondance, la trace retournée par notre procédure reste utilisable pour rejouer la preuve en **Coq**.

Néanmoins, la transformation proposée dans [FKW00] ne permet de travailler qu'avec les systèmes de réécriture dont le membre gauche de chaque règle ne contient que des sous-termes paresseux qui sont des variables. De ce fait, le choix d'annotation paresseuse est relativement limité. Afin de pouvoir appliquer notre méthode à un plus grand ensemble de systèmes de réécriture, nous proposons une nouvelle transformation qui pré-traite un système de réécriture en remplaçant les sous-termes paresseux qui ne sont pas des variables dans les membres gauches de règles. La correction de cette transformation et la correspondance entre les dérivations dans les systèmes de réécriture originel et transformé sont établies et prouvées pour les systèmes de réécriture *constructor-based*.

Les résultats obtenus dans le cadre de ce travail sont présentés dans [Ngu01b].

6.2. Contraintes

Mots clés : *résolution de contraintes, filtrage, combinaison de résolveurs, surréduction.*

Au delà de nos travaux de synthèse sur la résolution de contraintes [CK01d], nous avons obtenu de nouveaux résultats sur la résolution de contraintes de filtrage (ou filtre-équations). Le mécanisme de filtrage est essentiel pour l'application de règles de réécriture, car il détermine si un terme est une instance d'un membre gauche de règle, modulo éventuellement une théorie équationnelle. L'étude du filtrage équationnel est primordiale dans la perspective d'améliorer l'expressivité des langages de programmation à base de règles. Nous nous intéressons plus précisement à résoudre des filtre-équations en utilisant des techniques de surréduction et de combinaison.

6.2.1. Résolution de contraintes de filtrage pour des mélanges de théories

Participant : Christophe Ringeissen.

Une équation de filtrage est une équation entre deux termes dont l'un est clos (éventuellement par des constantes libres). Il est trop souvent admis qu'il suffit d'utiliser un algorithme de filtrage dans une théorie équationnelle pour résoudre des équations de filtrage en présence de symboles additionnels n'appartenant pas à la présentation de la théorie équationnelle (des symboles qui sont alors dits **libres**, comme par exemple le symbole $\longrightarrow$ dans le calcul de réécriture). Ce problème est toutefois, en général, difficile. Nous avons ainsi montré l'existence d'une théorie équationnelle pour laquelle le filtrage est décidable (resp. finitaire) alors que le filtrage avec symboles libres ne l'est plus [Rin01]. Ce résultat fournit une première réponse partielle au problème encore ouvert concernant l'unification: on ne connait pour l'instant aucune théorie équationnelle pour laquelle unification et unification avec symboles libres peuvent être respectivement décidable et indécidable.

Nous avons également travaillé sur le problème du filtrage dans des mélanges de théories non-disjointes, dans le prolongement de nos travaux antérieurs concernant d'une part la combinaison des algorithmes de filtrage dans le cas disjoint et d'autre part, une notion de constructeur adapté au cas non-disjoint et étudié jusqu'à présent principalement pour le problème de satisfaisabilité. Nos premiers résultats ont été présentés lors du workshop UNIF'01. Il apparaît que le filtrage est un problème tout particulièrement intéressant pour le mélange de théories non-disjointes dans le sens où il est possible d'obtenir de vrais résultats de modularité sans avoir à poser des conditions difficiles à satisfaire et qui puissent donc faire douter de la portée des résultats en pratique.

6.2.2. Inversion de fonctions

Participant : Claude Kirchner.

Dans le cadre de la programmation par règle, une fonction, comme par exemple le passage au carré, est spécifiée par un système de réécriture confluent et terminant. Dans ce contexte il serait particulièrement utile de pouvoir inférer automatiquement l'inverse des fonctions ainsi définies, donc dans notre exemple la fonction racine carrée. Pour aller dans ce sens, nous avons considéré avec Nachum Dershowitz de l'université de Tel Aviv, le problème de l'inversion pour des fonctions définies par des systèmes de réécriture convergents sur les termes clos. Nous nous sommes donc intéressés à la recherche des unificateurs des problèmes de la forme $t = G$ où G est un terme clos.

Dans un premier temps, la procédure d'inversion de [DM99] est reformulée sous la forme de l'application de règles de réécriture du but, contrôlée par des stratégies. Puis, dans le cas des spécifications complètement définies, de nouvelles règles et stratégies de résolution du problème d'inversion sont présentées [DK01].

6.3. Déduction automatique

Mots clés : *déduction modulo, procédures de décision, théories contraintes.*

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. Nous avons présenté à la conférence RTA'01 un exposé invité sur l'application de la réécriture à la déduction et à la vérification [Rus01b], ainsi qu'un exposé invité à la conférence LPAR'00 sur la vérification de protocoles de sécurité par déduction automatique [JRV00].

6.3.1. Déduction modulo

Participants : Eric Deplagne, Claude Kirchner, Jürgen Stuber.

En collaboration avec Gilles Dowek (projet LOGICAL) et Thérèse Hardin (LIP6 et projet MOSCOVA), 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 [DHK98]. L'une des originalités de l'approche est que cette congruence est définie non seulement sur les termes mais aussi sur les propositions.

En formalisant l'axiome d'induction dans le cadre de $HOL_{\lambda\sigma}$(exprimant la logique d'ordre supérieur comme une théorie du premier ordre modulo [DHK01a]), nous avons montré 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**.

ENAR est une méthode de recherche de preuve générique dont l'intérêt, mais aussi l'inconvénient est qu'il est peu restrictif et autorise un grand nombre d'inférences redondantes. Nous avons donné dans [Stu01] une preuve de complétude différente en utilisant une technique de construction de modèle basée sur des ordres bien fondés. Cette technique est analogue à la technique utilisée dans la preuve de complétude de la superposition modulo des théories algébriques [Stu00]. Comme élément nouveau elle contient la construction d'un arbre infini qui permet de construire un modèle dans lequel les équivalences utilisées pour la surréduction sont vraies. Cette technique de preuve donne à la fois des restrictions fortes sur des inférences (il suffit d'appliquer la résolution et la surréduction aux literaux maximaux) et des critères forts de redondance (les clauses impliqués par des clauses plus petites sont redondantes). Une restriction de cette méthode vient de la nécessité de trouver une ordre bien fondé qui oriente les équivalences en règles de réécriture. Un cas important qui rentre dans ce cadre est celui des hiérarchies de définitions, comme on les trouve, par exemple, dans des fragments de la théorie des ensembles.

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

Participant : Hélène Kirchner.

Dans les spécifications formelles de type algébrique, le système de types est restreint à des sortes, sous-sortes et types fonctionnels du premier ordre. Ce n'est pas le cas des spécifications basées sur les modèles qui autorisent des types d'ordre supérieur interprétés de façon ensembliste comme des produits cartésiens, des espaces fonctionnels et des parties d'ensembles. Dans [KM01b], nous proposons 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. Nous avons fait le lien avec les modèles ensemblistes classiques et travaillé sur l'existence de modèles initiaux dans différentes classes de modèles.

6.3.3. *Procédures de décision*

Participants : Silvio Ranise, Michaël Rusinowitch, Laurent Vigneron.

Nous étudions comment obtenir de manière uniforme des procédures de décision pour des théories fréquemment utilisées en vérification (listes, tableaux, ...) en spécialisant des systèmes de preuves par réécriture. Ces algorithmes de décision sont dérivés par exemple en montrant la terminaison des règles de superposition (implantées dans **daTac**) sur certaines classes de formules [ARR01]. Nous étudierons en particulier avec cette approche des fragments de la théorie des ensembles.

La **clôture par congruence** est une procédure dont le but est de construire une représentation compacte d'une théorie équationnelle. En collaboration avec l'Université de Stony Brook, NY, nous avons utilisé cette technique pour définir des procédures de décision efficaces fondées sur la normalisation par réécriture, indispensable pour développer des environnements de preuve efficaces [BTV01]. Ces travaux permettent de mieux comprendre de nombreux résultats dans ce domaine et de voir les liens entre eux (Nelson et Oppen [NO80], Downey, Sethi et Tarjan [DST80], Shostak [Sho84]). Ces résultats s'étendent aux opérateurs associatifs-commutatifs. La méthode s'appuie 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 est également utilisée pour résoudre le problème du mot pour des théories AC closes, sans utiliser d'ordre de simplification AC.

6.3.4. *Preuves par induction*

Participant : Michaël Rusinowitch.

En collaboration avec Sorin Stratulat, nous poursuivons une étude sur l'intégration des procédures de décision dans les démonstrateurs par induction du type de **SPIKE**. Nous avons défini des schémas de règles d'inférence assurant la correction du démonstrateur [ARS01b]. Un prototype intégrant l'arithmétique linéaire et la théorie de l'égalité a été développé et validé sur des exemples. 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 dans une preuve de l'équivalence entre 2 algorithmes pour contrôler la conformité du débit de la source dans un réseau ATM. Ceux-ci nécessitent des tactiques spécifiques (introduites par l'utilisateur) dans la preuve en PVS [RSK00].

D'autre part nous avons simplifié notre technique de preuve observationnelle par induction et contextes critiques [BBR98] dans un article à paraître, en collaboration avec A. Bouhoula [BR01]. Rappelons que 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. Les possibilités d'augmenter l'automatisation de ce type de preuves sont importantes et l'approche par contextes critiques semble offrir une alternative prometteuse à la coinduction.

6.3.5. *Preuves avec contraintes*

Participant : Michaël 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 que la convergence de la réécriture ordonnée est décidable pour une large classe d'ordres dans l'article [CNNR02].

6.4. Preuve de propriétés de programmes

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

Nous avons développé de nouvelles techniques de preuves de propriétés dans les théories équationnelles (terminaison, complétude), plus particulièrement pour la réécriture avec stratégies.

6.4.1. Preuves de terminaison par induction

Participants : Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner.

En 1999, nous avons proposé une nouvelle méthode de preuve de terminaison pour la réécriture avec stratégies. Cette méthode, mise au point dans un premier temps pour la stratégie *innermost*, fait appel à l'induction explicite sur la propriété de terminaison elle-même. On établit qu'un terme termine sur l'algèbre des termes clos 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.

L'étape d'abstraction, qui n'est autre que l'application de l'hypothèse d'induction, nécessite un test de satisfiabilité des contraintes d'ordre. L'étape de surréduction est accompagnée d' un test de compatibilié des substitutions de *narrowing* avec les variables d'abstraction: on vérifie la satisfiabilié de contraintes dites d'abstraction.

L'étude de cette technique de preuve a été approfondie, notamment dans deux directions: l'automatisation et l'extension de la méthode à d'autres stratégies que la stratégie innermost. Pour profiter du fait que nos contraintes d'ordre étaient, grâce à la puissance de l'induction, en général très simples, et même plus simples qu'avec les méthodes traditionnelles de preuve de terminaison, nous avons cherché à supprimer le test de satisfiabilité des contraintes d'abstraction. Nous avons ainsi établi que sans ce test, notre mécanisme engendrait un arbre de preuve incluant l'arbre de preuve du mécanisme originel, mais qui, s'il convergeait, établissait également la preuve de terminaison. Une nouvelle procédure de preuve de terminaison a été proposée, qui est, lorsque les contraines d'ordre sont triviales - et c'est le cas pour bon nombre d'exemples - complètement automatique. Une implantation, automatisée de manière optimale, a été réalisée. Ces travaux ont été soumis à une conférence [GKF01].

D'autre part, notre principe de preuve reposant, contrairement aux autres méthodes de preuve de terminaison, sur la relation de réécriture elle-même, nous avons pu proposer une extension à d'autres stratégies que la stratégie innermost. Nous avons défini de nouvelles régles d'inférence pour la terminaison de la stratégie outermost, pour laquelle aucune technique spécifique de preuve n'existait jusqu'alors en réécriture, et pour les stratégies locales sur les opérateurs, telles qu'elles sont utilisées dans des langages comme OBJ3, CafeOBJ ou Maude. Dans ce dernier cas, un travail important a été réalisé pour étendre la notion de « règles utilisables » d'un terme, déjà utilisée pour augmenter la puissance de notre mécanisme dans le cas innermost. Nous avons ainsi défini, dans le cas des stratégies locales, le calcul d'un sur-ensemble des règles servant à réduire un terme donné, ce qui peut permettre aisément de prouver la terminaison de ce terme, lorsque l'hypothèse d'induction ne peut être appliquée. Ces travaux ont été publiés au workshop sur les stratégies de la conférence IJCAR'2001, et dans la revue électronique ENTCS [FGK01].

6.4.2. Test de complétude suffisante

Participants : Olivier Fissore, Isabelle Gnaedig, Anouar Ben Hmida.

Dans le cadre d'un stage de fin d'études du diplôme d'ingénieur de l'Université de Tunis, une étude comparative des différentes procédures de test de complétude suffisante d'une spécification de réécriture

simple a été réalisée, en vue d'une implantation en **ELAN**, qui puisse s'étendre ultérieurement à la réécriture conditionnelle et équationnelle. L'algorithme de Kounalis a été choisi, traduit sous forme de règles d'inférence puis programmé en **ELAN**. Cette implantation permet le test de complétude de programmes **ELAN** sous forme de règles non nommées.

6.4.3. Complexité implicite des calculs et optimisation de programmes

Participants : Isabelle Gnaedig, Claude Kirchner, Mitch Harris.

Dans le cadre de l'action « Analyse de la Complexité et Transformation de programmes » de l'opération « Qualité et Sûreté du logiciel » du Contrat de Plan Etat-Région, un stage a été effectué par Mitch Harris, et co-encadré par Jean-Yves Marion, du projet Calligramme, sur l'application de l'étude de la complexité implicite des programmes à leur optimisation. La complexité implicite est la complexité de la fonction dénotée par le programme. Son analyse est réalisée par l'intermédiaire d'outils provenant de la théorie de la démonstration et de la réécriture. Cette analyse permet (i) de déterminer la complexité du programme analysé, (ii) de savoir si le programme analysé est efficace, et (iii) de transformer le programme analysé en un programme efficace.

Les structures d'**ELAN** et de **TOM** ont été étudiées, dans le but de voir comment y intégrer un outil de preuve de terminaison, utilisant les ordre lexicographiques et les ordres polynomiaux pour calculer la complexité implicite d'un ensemble de règles. On en a déduit comment intégrer un mécanisme de mémorisation des étapes de réécriture déja effectuées. Une implantation de ce mécanisme a été réalisée dans **TOM**, augmentant considérablement l'efficacité des programmes pour lesquels certaines étapes de réécriture sont souvent répétées.

6.5. Vérification

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

Cette année nous avons obtenu des résultats pratiques et théoriques sur la vérification des protocoles de sécurité et des systèmes hybrides.

6.5.1. Vérification de protocoles cryptographiques

Participants : Mohamed-Mehdi Bouallagui, Yannick Chevalier, Michaël Rusinowitch, Mathieu Turuani, Laurent Vigneron.

Nos travaux sur ce thème ont profité de l'acceptation du projet AVISS, projet européen FET Open qui a débuté en mai 2001 en collaboration avec l'Université de Fribourg (Allemagne) et l'Université de Gênes (Italie).

En partant de nos résultats des années précédentes, concrétisés par une première version du logiciel **CASRUL**, nous avons spécifié le langage (HLPSL) utilisé pour décrire les protocoles puis écrit une nouvelle version de **CASRUL** compilant une spécification en HLPSL en un système de réécriture décrit dans un format intermédiaire (IF). Un petit traducteur est ensuite utilisé pour convertir ce système de réécriture en une spécification reconnue par l'outil chargé de chercher des attaques dans le protocole (un model-checker pour l'Université de Fribourg, SAT pour l'Université de Gênes, et **daTac** pour nous).

Les principaux résultats de ces travaux [Rus01a][Rus01e] sont :

- offrir une plus grande expressivité dans la description des protocoles cryptographiques : extension du vocabulaire de composition des messages (tables de clés, clés composées, encodage par *Xor*, clés fraîches), description de rôles pour les participants au protocole, réalisation de sessions parallèles, et description de nouveaux buts (authentification, secret à court terme),
- fournir un système de réécriture plus simple et plus efficace, avec en particulier la mise en place d'une stratégie paresseuse du comportement de l'intrus [CV01b][CV01a].

Ces nombreuses améliorations ont permis d'étudier un grand nombre de protocoles (plus d'une vingtaine) avec le démonstrateur **daTac**. Les temps obtenus sont très faibles, au regard de la non spécialisation de **daTac** pour ce genre de travaux.

Nous avons étudié la complexité de la recherche d'une attaque dans un protocole comportant un nombre fini de sessions, et avons montré que ce problème est NP-complet [RT01b][RT01a]. Ce résultat a été obtenu sans limiter la taille des messages, et supporte le codage par clés symétriques composées. En particulier il s'agit du premier résultat de décidabilité pour l'insécurité des protocoles comportant des clés composées (avec nombre de sessions borné). Signalons qu'un grand nombre de protocoles comme SSL manipulent des clés composées.

6.5.2. Vérification de systèmes infinis et calculs de points fixes

Participants : Julien Musset, Michaël Rusinowitch.

Les systèmes de transitions s'appliquent bien à la modélisation des systèmes complexes. Leurs propriétés peuvent se vérifier par des procédures de **model-checking** fondées sur des calculs itératifs de points fixes. L'approche symbolique dans laquelle les ensembles d'états sont représentés par des formules permet la validation de systèmes à nombre d'états infini. Mais la détection de la convergence des calculs de points fixes nécessite des tests d'inclusion d'ensembles qui sont très coûteux. Les tests approchés par **comparaison locale** sont suffisants pour quelques classes d'automates tels que les automates rectangulaires ou les systèmes hybrides o-minimaux. Notre objectif est d'améliorer les comparaisons locales afin de pouvoir traiter des systèmes situés hors de ces classes en exploitant la connaissance de la propriété à prouver, de manière à éviter le test d'inclusion. Nous avons d'abord proposé une transformation des systèmes de transition déterministes, exploitant la propriété à valider, de manière à éviter complètement le test d'inclusion. Cette transformation a ensuite été appliquée à des systèmes non déterministes pour améliorer les comparaisons locales [MR01a][MR01b]. Pour que les systèmes hybrides soient accessibles à cette approche nous avons dû en modifier un peu la sémantique.

6.5.3. 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. Hassen Kacem a développé un outil utilisant le système **ELAN** pour vérifier des propriétés d'atteignabilité de produits d'automates temporisés. Cet outil possède, sur les quelques exemples étudiés, des performances proches des outils dédiés et offre l'avantage d'être facilement modifiable ou configurable. En effet, le système fonctionne à l'aide de règles de réécriture sur des contraintes, écrites dans le système **ELAN**, compréhensibles, et facilement modifiables. Cela offre une souplesse pour l'expérimentation et l'utilisation qui n'est pas présente dans les outils classiques dédiés. Ces résultats ont été présentés dans [BBKK01b] et dans [BBKK01a].

Cet outil a par la suite été étendu en outil pour la vérification de propriétés d'atteignabilité de classes plus générales de systèmes hybrides comme les automates hybrides linéaires ou les p-automates définis dans le cadre du projet RNRT CALIFE dans lequel l'équipe est impliquée. Cette extension a été utilisée pour vérifier la conformité du protocole de l'ABR, en utilisant la modélisation de ce problème proposée par nos partenaires du projet RNRT CALIFE.

6.6. Complexité

Mots clés : *Calculabilité, complexité, problème de décision en théorie du contrôle, calculabilité sur les réels.*.

Nous avons obtenu des résultats sur la décidabilité/indécidabilité de propriétés en théorie du contrôle reliés à des problèmes de vérification de propriétés des systèmes hybrides. Nous travaillons aussi sur les problèmes de complexité dans le modèle de Blum Shub et Smale.

6.6.1. Complexité dans le modèle de Blum Shub et Smale

Participants : Olivier Bournez, Paulin Jacobé de Naurois.

Pour modéliser les calculs sur les réels, Blum Shub et Smale ont introduit en 1989 un modèle de calcul parfois appelé machine de Turing réelle, ou modèle B(C)SS. Ce modèle, contrairement à l'analyse récursive, mesure la complexité des problèmes en terme du nombre d'opérations arithmétiques nécessaires à leur résolution indépendemment des représentations des réels, ce qui s'avère naturel en complexité algébrique, ou plus généralement pour décrire les problèmes sur les réels. Ce modèle a par la suite été étendu en une notion de modèle de calcul sur une structure logique arbitraire. Puisque la complexité classique correspond à la restriction de cette notion aux structures booléennes, ce modèle apporte un éclairage nouveau sur les problèmes plus anciens de la complexité classique et sur ses liens avec la logique.

Paulin de Naurois a débuté en novembre une thèse avec Olivier Bournez et Jean-Yves Marion (Equipe Calligramme) sur l'étude de la complexité dans ce modèle, et en particulier sur ses liens avec la complexité implicite des calculs. La complexité implicite des calculs est une des approches, a priori distincte, qui permet de comprendre les relations entre la logique, et plus particulièrement la théorie de la démonstration, la complexité algorithmique, et la théorie de la programmation. Elle apporte l'avantage d'une apparente simplicité ainsi qu'une absence de référence aux ressources impliquées dans le calcul.

6.6.2. Contrôlabilité des systèmes linéaires commutés

Participant : Olivier Bournez.

Nous nous sommes intéressés à la décidabilité de la mortalité des matrices. Ce problème consiste, étant donné un ensemble fini de matrices carrées, à déterminer s'il existe un produit de ces matrices qui est la matrice nulle. Dire si ce problème est décidable est équivalent à dire s'il est possible de déterminer si un système linéaire commuté est contrôlable. Le problème est donc motivé par la vérification de propriétés des systèmes hybrides.

En collaboration avec Michaël Branicky, dans [BB01], nous avons dressé un panorama des résultats existants sur ce problème et prouvé quelques extensions. Nous avons ainsi montré que le problème est indécidable pour les matrices 3×3, décidable pour 2 matrices 2×2, et ouvert pour $k \geq 3$ matrices 2×2. Nous avons montré que le problème pour 2 matrices 2×2 était indécidable dans le modèle de calcul de Blum Cucker Shub et Smale sur les réels, et que le cas ouvert pour $k \geq 2$ matrices 2×2 se reliait à plusieurs autres problèmes mentionnés comme ouverts dans la littérature.

6.7. CALIFE

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

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

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 oeuvre 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.

Ce travail rentre dans le cadre du projet RNRT CALIFE.

7. Contrats industriels

7.1. GasEL

Participants : Hélène Kirchner, Olivier Bournez, Liliana Ibănescu.

Contrat CNRS/PSA.

L'objectif du projet GasEL est d'élaborer un logiciel basé sur la réécriture pour la génération de mécanismes détaillés d'oxydation et de combustion de molécules d'hydrocarbures polycycliques. Les partenaires dans ce projet sont l'INPL, le CNRS et Peugeot Citroën Automobiles. Les laboratoires impliqués sont le LORIA et le DCPR.

Les objectifs scientifiques sont d'utiliser les systèmes de réécriture comme **ELAN** pour la génération automatique de mécanismes détaillés, d'élaborer et de valider des mécanismes détaillés d'oxydation et de combustion de molécules d'hydrocarbures polycycliques, ce qui devrait permettre au partenaire industriel de reformuler des carburants pour réduire la consommation de carburant et les émissions de polluants et de revoir la conception des moteurs.

Ce projet peut être vu comme une extension du projet EXGAS qui a comme résultat un logiciel développé au DCPR de Nancy qui est actuellement capable de générer des mécanismes détaillés d'oxydation à basse température et de combustion à haute température pour les espèces suivantes: alcanes linéaires, alcanes ramifiés, alcènes linéaires, alcènes ramifiés, éthers, cyclanes monocycliques polysubstitués. Puisque l'extension de EXGAS aux hydrocarbures polycycliques est impossible, une autre approche informatique est nécessaire, basée sur le système **ELAN** développé au LORIA et c'est le but du projet GasEL.

Le travail réalisé depuis janvier 2001 se situe dans trois directions: l'analyse et la compréhension du problème chimique, le prototypage d'EXGAS en **ELAN** et la recherche d'une bonne représentation interne des molécules polycycliques.

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 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 participons à une Action Concertée Incitative Cryptologie du Ministère de la Recherche sur la vérification de protocoles de sécurité (VERNAM, 2001-2003) avec l'université de Provence et l'ENS Cachan.

Nous participons au groupe de travail « Sémantiques Logiques et Opérationnelles, Vérification et Optimisation »(SLOVO) du GDR « Spécifications, Preuves et Tests ».

8.3. Actions européennes

Nous participons au Groupe de Travail ESPRIT CoFI, **Common Framework Initiative for Algebraic Specification and Development**, et nous en avons coordonné le Tools Task Group jusqu'en avril 2001.

Nous participons au programme européen : Information Society Technologies (IST), dans le cadre d'un FET Open Assessment Project: IST-2000-26410 en 2001-2002 avec Software Engineering Group, Institut für Informatik, ALU Freiburg et Mechanized Reasoning Group, DIST, Università degli Studi di Genova. L'objectif de ce projet est la construction d'un vérificateur de protocoles de sécurité comprenant un compilateur et 3 technologies de validation.

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).

8.5. Relations bilatérales internationales

Nous entretenons des relations suivies et importantes avec l'université d'Amsterdam et le CWI dans le domaine de la réécriture et des spécifications formelles algébriques. C'est dans ce cadre que l'équipe associée AirCube (Rewrite Rule Research) a été initiée en juin 2001.

Nous sommes en relation avec l'Université de Gènes et avec l'université de Fribourg dans le domaine de la démonstration automatique.

Nous collaborons avec SUP'COM (École Supérieure des Communications de Tunis), dans le cadre d'un projet STIC-2000-2 avec A. Bouhoula , sur le thème : vérification formelle pour les logiciels de télécommunication. La thèse en cotutelle de T. Abbes se déroule dans ce cadre.

Nous participons à l'action franco-italienne du programme GALILEO, avec l'Université de Gênes (A. Armando): inégration de procédures de décision dans un système de preuve automatique. La thèse en cotutelle de S. Ranise se déroule dans ce cadre.

Nous avons commencé en 2001 une collaboration franco-brésilienne avec le département de mathématiques appliquées de l'Université Fédérale (UFRN) à Natal. Cette collaboration a pour cadre le développement d'outils pour le domaine des spécifications algébriques. Se fondant sur la complémentarité des connaissances des partenaires français et brésiliens dans les domaines des systèmes de réécriture et des spécifications algébriques, le projet FERUS, labellisé par le CNPq et l'INRIA suite à l'appel d'offres INRIA/PROTEM 2000, se propose de contribuer au développement de la Spécification et Réutilisation de Logiciels. Dans ce contexte, nous travaillons à la mise en oeuvre d'un gestionnaire de bibliothèques de composants logiciels réutilisables. L'objectif est de concevoir un outil générique qui puisse être utilisable aussi bien par le langage fédérateur de spécifications algébrique (CASL) développé par le groupe de travail européen CoFI, auquel participe PROTHEO, que par un langage basé sur la réécriture comme **ELAN**, qui fournit déjà un outil d'exécution pour un certain sous-langage de CASL.

8.6. Accueils de chercheurs étrangers

- Mitch Harris, de l'Université d'Illinois, à Urbana Champaing (USA), a travaillé dans l'équipe sur la complexité implicite des programmes et leur application à l'optimisation de programmes **TOM**.
- Anamaria Martins Moreira, de l'UFRN (Natal, Brésil), a effectué une visite à Nancy dans le cadre du projet franco-brésilien FERUS. Cette première visite a permis de mieux cerner le projet, et l'utilisation de la réécriture comme aide à la généralisation, un concept introduit dans la thèse d'Anamaria Martins Moreira et qui constitue une opération essentielle pour la réutilisation de composants logiciels.

- Paliath Narendran, de l'Université d'Albany, a travaillé 1 mois dans le projet sur des problèmes de contraintes ensemblistes.
- Juan Ortega, de l'Université de Malaga (Espagne), a programmé un traducteur des Messages Sequences Charts vers la syntaxe de protocoles CAS, prise en entrée par CASRUL.
- Rakesh Verma, a travaillé pendant son séjour à l'utilisation des techniques de partage et de mémorisation pour l'amélioration de l'efficacité de programme par réécriture et d'autre part à la reconstruction de preuve associative-commutative.

9. Diffusion des résultats

9.1. Animation de la Communauté scientifique

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

- Olivier Bournez : membre de la commission de spécialiste, 27ième section, de l'université Henri-Poincaré Nancy 1. Membre du conseil des opérations, co-responsable de l'action « Plateforme » de l'axe « Qualité et Sûreté du Logiciel » du Contrat de Plan Etat-Région.
- Claude Kirchner : Responsable du thème QSL du pôle intelligence logicielle, chargé de mission pour la formation par la recherche à l'INRIA, responsable de projet du développement de la troisième tranche du bâtiment LORIA, membre du conseil d'orientation scientifique du LORIA, membre de la section locale d'audition de l'INRIA Lorraine, membre du comité d'évaluation du laboratoire d'informatique de l'université de Porto et du Laboratoire d'Informatique Fondamentale de Marseille, membre de la commission de spécialistes (27^{e}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**, et **Journal of Information Science and Engineering**;
  Membre des comités de programmes des conférences « 2nd International Workshop on Rule-based Programming »: RULE 2001, « 8th Workshop on Logic, Language, Information and Computation »: WoLLIC 2001, « 13th European Summer School in Logic, Language and Information »: ESSLLI 2001, « 10th Portuguese Conference on Artificial Intelligence »: EPIA'01, co-organisateur du « INRIA / NSC Taiwan, MultiMedia and XML Workshop », organisateur de la journée « Qualité et Sûreté du Logiciel » de septembre.
  Président du groupe de travail IFIP WG 1.6 sur la réécriture, membre du conseil d'administration de CADE-inc (*Board of trustees*) jusqu'en juillet 2001, membre fondateur de IFCoLog.
- Hélène Kirchner : Directrice du LORIA et de l'INRIA Lorraine, animatrice du Pôle de Recherche Scientifique et Technologique Lorrain « Intelligence Logicielle ».
  Membre du Comité éditorial des revues **Annals of Mathematics and Artificial Intelligence** et **Computing and Informatics**, membre des Comités de programmes: AMAST 2002 (co-chair), FROCOS'2002, CP 2001, membre des Steering Committees des conférences PPDP et RTA.
  Responsable du Groupe Tools du Working Group Esprit CoFI (Common Framework Initiative for algebraic specification and development of software) jusqu'en mars 2001, membre des groupes IFIP WG 1.3 (**Foundations of Systems Specifications**) et IFIP WG 1.6 (**Term Rewriting**).
  Membre nommé du Comité National du CNRS (section 07) de septembre 2000 à septembre 2001, membre du Comité d'Orientation du RNTL depuis 2000, membre nommé aux Commissions de spécialistes de Nancy 1, Nancy 2 et de l'Université Louis Pasteur à Strasbourg, membre nommé dans les Comité d'Evaluation du LRI et du LSV (présidente).
- Pierre-Etienne Moreau : membre du comité de programme de LDTA'2001 et de CALCULEMUS'2001.

- Christophe Ringeissen : Correspondant pour les projets européens du LORIA, membre des comités de programme de STRATEGIES'01 (workshop de IJCAR'01), CoSolv'01 (workshop de CP'01), FroCoS'02, AMAST'02, chair-assistant de AMAST'02, co-animateur du groupe « Collaboration de Solveurs » du GDR ALP.
- Michaël Rusinowitch : membre du groupe IFIP WG 1.6 (Réécriture), membre du *Steering Committee* de FTP (1999-2001) (First-Order Theorem-Proving), membre du comité de programme de FROCOS-02, des dixièmes Journées Francophones de Programmation Logique et Programmation par Contraintes (2002), co-organisateur du Floc'02 Workshop on Complexity in Automated Deduction (Copenhague), membre du comité d'organisation de RTA (1999-2001), membre suppléant de la commission d'évaluation de l'INRIA, et de la CSE de l'INPL, Nancy 2 et Lille, membre du jury du prix de thèse SPECIF.
- Laurent Vigneron: membre élu du Conseil du laboratoire du LORIA ; membre du comité de rédaction de « La lettre du LORIA » ; correspondant communication de l'équipe PROTHEO ; secrétaire du groupe IFIP WG 1.6 sur la réécriture ; administrateur pages web : la « Rewriting Home Page », la page du groupe IFIP WG 1.6, la page web de l'équipe PROTHEO ; membre suppléant de la commission de spécialistes en 27e section à l'Université Nancy 2.

9.2. Enseignement universitaire

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

- 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 1.
- Olivier Bournez a été membre permanent du Jury du DEA 2001 de l'Université Henri-Poincaré Nancy 1.
- Isabelle Gnaedig a coordonné les enseignements du module **Spécification de Programmes Spécification de Logiciels** de l'ESIAL (3ème année) et du DESS d'informatique de l'Université Henri Poincaré – Nancy 1.
  Elle a également organisé et encadré des travaux dirigés et travaux pratiques autour des spécifications algébriques, du langage LOTOS et de la sémantique des processus communicants dans le cadre de ce module.
- Claude Kirchner a donné deux cours: « Contraintes et optimisation Combinatoire » avec Alexander Bockmair, et « Logique et démonstration automatique » avec Adam Cichon, dans le cadre du DEA d'Informatique de l'Université Henri Poincaré – Nancy 1.
- Pierre-Etienne Moreau a donné un cours sur les structures de données à l'ESIAL (1ère année).

Le projet comprend également des Moniteurs, ATERs et Maîtres de Conférences 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

- Olivier Bournez :
  représentation du LORIA et exposé au « Forum des Thèses » à l'ENS Paris en Janvier 2001,
  « Constraints in Model-Checking algorithms », exposé au Workshop « First Order Constraints » à Luminy en Mai 2001,
  « En dehors du monde des machines de Turing, les calcules terminent-ils? », exposé dans le cadre de la « Journée Terminaison » de l'axe « Qualité et Sûreté du Logiciel » du pôle « Intelligence Logicielle » du Contrat de Plan Etat Région, à Nancy, en Avril 2001.

- Isabelle Gnaedig :
  « Terminaison par induction », exposé dans le cadre de la « Journée sur la Terminaison » de l'axe « Qualité et Sûreté du Logiciel » du pôle « Intelligence Logicielle » du Contrat de Plan Etat Région, à Nancy, en Avril 2001.
- Claude Kirchner :
  « Induction as Deduction Modulo », présentation invitée au workshop « Deduktion », à Dagstuhl, Allemagne en mars 2001,
  « Elements of formal semantics for XSLT », séminaire Franco-Taiwannais à Poli, Taiwan en mars 2001,
  « The rewriting calculus: A semantics for rule based languages » et « **ELAN**: ideas for the future » présentations au workshop ELAN/ASF+SDF à Thionville en avril 2001,
  « **ELAN**, Modélisation et preuve en calcul de réécriture », présentation invitée à l'atelier AFADL (Nancy, juin) « Approches Formelles dans l'Assistance au Développement des Logiciels », à Nancy en juin 2001,
  « Deduction Modulo », Séminaire au « Automated Software Engineering Group » de la Nasa, centre de Ames à Mountain view, Californie en juillet 2001.
- Hélène Kirchner :
  « Rule-based computation and deduction », tutoriel à la 13th European Summer School in Logic, Language and Information, ESSLLI 2001, Helsinki, Finlande, en Août 2001,
  exposé invité « Prototyping and verification with rules and strategies », Cyber Assist International Symposium, Tokyo, en mars 2001.
- Pierre-Etienne Moreau : exposé invité à « Aachen University of Technology (RWTH) » dans le groupe du Prof. Dr. K. Indermark, participation à un cours sur la réécriture presenté à ESSLLI (Finlande), et à un tutorial **ELAN** presenté à ETAPS (Genes, Italie).
- Michaël Rusinowitch : exposés invités: LPAR (La Réunion, nov. 2000), RTA (Utrecht, mai 2001), Journées Systèmes Infinis (Paris, mars 2001) Workshop on Logical Aspects of Cryptographic Protocols (Paris, juillet 2001), Journées Sûreté des Logiciels Critiques (Grenoble, novembre 2001),
  un exposé au Workshop on Specification and Analysis of Secure Cryptographic Protocols (Dagstuhl, septembre 2001),
  un exposé au Workshop on First-Order Constraints CIRM (Marseille, mai 2001),
  des séminaires á TU, Vienne et à Université de Kiel.
- Christophe Ringeissen a été co-animateur d'un tutorial d'ETAPS'01 sur **ELAN**.
- Laurent Vigneron a participé à une réunion de l'ACI Cryptologie VERNAM, à Marseille, en mars et à la réunion du groupe IFIP WG 1.6, à Utrecht, Pays-Bas, en mai.

Mis à part les congrès et colloques où furent présentés nos travaux, nous avons participé aux événements suivants :

RTA'01, ETAPS'01, IJCAR'01, CAV'01, PAPM-PROBMIV'01, PNPM'01, Wollic'2001, Esslli'2001, RULE2001, PPDP2001, ICLP-CP-RCoRP, « 5th Dynamics Workshop: Dynamics and verification », Forum Franco-Allemand de la foire de Sarrebrück.

Les exposés suivants ont été présentés par des orateurs extérieurs dans le cadre du séminaire PROTHEO: Eelco Visser, « Language Independent Traversals for Program Transformation »; Stéphane Vaillant, « Une présentation finie de la théorie des ensembles au premier ordre »; Anamaria Martins Moreira, « Un outil pour l'aide à la Spécification et Réutilisation de Composants Logiciels »; Luigi Liquori, « Continuations »; Xavier Urbain, « Preuves automatiques et extensions hiérarchiques »; Marc Aiguier et Diane Bahrami, « Une axiomatisation de la réécriture »; Hitoshi Ohsaki, « Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories »; Jean-Louis Giavitto et Olivier Michel, « MGS: un langage de programmation utilisant des règles pour la modelisation et la simulation de systèmes dynamiques a structures dynamiques ».

9.3.2. Séjours de chercheurs

- Claude Kirchner : Visite invitée d'une semaine du 16 au 21 juillet dans le groupe « Automated Software Engineering Group » de la Nasa, centre de Ames (Mountain view, Californie).
  Visite d'une semaine à l'université de Natal (Brésil) dans le cadre de notre collaboration franco-brésilienne.
- Christophe Ringeissen : séjour d'un mois (2x15 jours) au DIMAp, UFRN, à Natal, au Brésil, dans le cadre du projet franco-brésilien FERUS, financé par le CNPq et l'INRIA.
- Jürgen Stuber : séjour à l'université de Sarrebrück, du 19 au 22 Décembre 2000.

9.4. Jurys de thèses et jurys divers

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

- Claude Kirchner : membre du jury de l'habilitation de Delia Kesner, Université d'Orsay.
- Hélène Kirchner : directeur et membre du jury de thèse de Hubert Dubois, Université Henri Poincaré – Nancy 1, rapporteur de la thèse de Laurent Kaiser.
- Michaël Rusinowitch : rapporteur de l'habilitation de R. Pichler (TUM Vienna, mai 2001) et de la thèse de X. Urbain (Orsay, octobre 2001).
- Jürgen Stuber : Rapporteur extérieur de la thèse de Guillem Godoy, Barcelone
- Isabelle Gnaedig et Pierre-Etienne Moreau ont participé aux jurys du concours d'entrée à l'ESIAL.

10. Bibliographie

Bibliographie de référence

[BBKT00] V. BLONDEL, O. BOURNEZ, P. KOIRAN, J. TSITSIKLIS. *The stability of saturated linear dynamical systems is undecidable.* in « Journal of Computer and System Science », 2000.

[BR95] A. BOUHOULA, M. RUSINOWITCH. *Implicit Induction in Conditional Theories.* in « Journal of Automated Reasoning », numéro 2, volume 14, 1995, pages 189–235.

[Cas98] C. CASTRO. *Building Constraint Satisfaction Problem Solvers Using Rewrite Rules and Strategies.* in « Fundamenta Informaticae », numéro 3, volume 34, 1998, pages 263-293.

[KKV95] C. KIRCHNER, H. KIRCHNER, M. VITTEK. *Designing Constraint Logic Programming Languages using Computational Systems.* éditeurs V. SARASWAT, P. VAN HENTENRYCK., in « Principles and Practice of Constraint Programming. The Newport Papers », The MIT Press, 1995, chapitre 1, pages 131–158.

[KM00] H. KIRCHNER, P.-E. MOREAU. *Promoting Rewriting to a Programming Language: A Compiler for Non-Deterministic Rewrite Programs in Associative-Commutative Theories.* in « Journal of Functional Programming », 2000, note : à paraître.

[KR94] H. KIRCHNER, C. RINGEISSEN. *Combining Symbolic Constraint Solvers on Algebraic Domains.* in « J. Symbolic Computation », numéro 2, volume 18, août, 1994, pages 113-155.

[KR98] C. KIRCHNER, C. RINGEISSEN. *Rule-Based Constraint Programming.* in « Fundamenta Informaticae », volume 34, 1998, pages 225–262.

[RV95] M. RUSINOWITCH, L. VIGNERON. *Automated Deduction with Associative Commutative Operators.* in « Applicable Algebra in Engineering, Communication and Computing », numéro 1, volume 6, janvier, 1995, pages 23–56.

Thèses et habilitations à diriger des recherche

[Dub01] H. DUBOIS. *Systèmes de Règles de Production et Calcul de Réécriture.* Thèse de doctorat d'université, Université Henri Poincaré – Nancy 1, septembre, 2001, file://ftp.loria.fr/pub/loria/protheo/THESES_2001/Dubois-These01.ps.gz note : Also available as Technical Report A01-T-200, LORIA, Nancy (France).

Articles et chapitres de livre

[BB01] O. BOURNEZ, M. BRANICKY. *On the mortality problem for matrices of low dimensions.* in « Theory of Computing Systems », 2001, note : à paraître. Also available as Technical Report A01-R-316, LORIA, Nancy (France).

[BKKR01] P. BOROVANSKÝ, C. KIRCHNER, H. KIRCHNER, C. RINGEISSEN. *Rewriting with strategies in **ELAN**: a functional semantics.* in « International Journal of Foundations of Computer Science », numéro 1, volume 12, février, 2001, pages 69–98.

[BR01] A. BOUHOULA, M. RUSINOWITCH. *Observational proofs by rewriting.* in « Theoretical Computer Science », 2001, note : à paraître. Also available as Technical Report A01-R-050, LORIA, Nancy (France).

[BTV01] L. BACHMAIR, A. TIWARI, L. VIGNERON. *Abstract Congruence Closure.* in « Journal of Automated Reasoning », 2001, note : à paraître. Also available as Technical Report A01-R-266, LORIA, Nancy (France).

[CK01a] H. CIRSTEA, C. KIRCHNER. *Rewriting and Multisets in Rho-calculus and **ELAN**.* in « Romanian Journal of Information, Science and Technology », numéro 1-2, volume 4, 2001, pages 33–48, note : ISSN: 1453-8245.

[CK01b] H. CIRSTEA, C. KIRCHNER. *The Rewriting Calculus - Part I.* in « Logic Journal of the Interest Group in Pure and Applied Logics », volume 9, May, 2001, pages 363-399, note : Also available as Technical Report A01-R-203, LORIA, Nancy (France).

[CK01c] H. CIRSTEA, C. KIRCHNER. *The Rewriting Calculus - Part II.* in « Logic Journal of the Interest Group in Pure and Applied Logics », volume 9, May, 2001, pages 401-434, note : Also available as Technical Report A01-R-204, LORIA, Nancy (France).

[CK01d] H. COMON, C. KIRCHNER. *Constraint Solving on Terms.* in « Lecture Notes in Computer Science », volume 2002, 2001.

[DHK01a] G. DOWEK, T. HARDIN, C. KIRCHNER. *HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic.* in « Mathematical Structures in Computer Science », numéro 1, volume 11, 2001, pages 21-45.

[KM01a] H. KIRCHNER, P.-E. MOREAU. *Promoting Rewriting to a Programming Language: A Compiler for Non-Deterministic Rewrite Programs in Associative-Commutative Theories.* in « Journal of Functional Programming (JFP) », numéro 2, volume 11, March, 2001, pages 207-251, note : Also available as Technical

Report A01-R-007, LORIA, Nancy (France).

[KM01b] H. KIRCHNER, P. D. MOSSES. *Algebraic Specifications, Higher-Order Types, and Set-Theoretic Models.* in « Journal of Logic and Computation », numéro 3, volume 11, 2001, pages 453–481.

[VRL01] R. VERMA, M. RUSINOWITCH, D. LUGIEZ. *Algorithms and Reductions for Rewriting Problems.* in « Fundamenta Informaticae », numéro 3, volume 46, May, 2001, pages 257-276, note : Also available as Technical Report A01-R-088, LORIA, Nancy (France).

Communications à des congrès, colloques, etc.

[ARR01] A. ARMANDO, S. RANISE, M. RUSINOWITCH. *Uniform Derivation of Decision Procedures by Superposition.* in « Conference of the European Association for Computer Science Logic, CSL'01 », address Paris (France), September, 2001, note : Also available as Technical Report A01-R-047, LORIA, Nancy (France).

[ARS01b] A. ARMANDO, M. RUSINOWITCH, S. STRATULAT. *Incorporating Decision Procedures in Implicit Induction.* in « 9th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (CALCULEMUS'2001) », address Sienne (Italy), June, 2001, note : Also available as Technical Report A01-R-304, LORIA, Nancy (France).

[BBKK01a] E. BEFFARA, O. BOURNEZ, H. KACEM, C. KIRCHNER. *Verification of Timed Automata Using Rewrite Rules and Strategie.* in « Seventh Biennial Bar-Ilan Symposium on the Foundations of Artificial Intelligence (BISFAI'01) », address Ramat-Gan (Israel), June, 2001, note : Also available as Technical Report A01-R-209, LORIA, Nancy (France).

[BBKK01b] E. BEFFARA, O. BOURNEZ, H. KACEM, C. KIRCHNER. *Verification of Timed Automata Using Rewrite Rules and Strategies.* in « Sixth Annual Workshop of the ERCIM Working Group on Constraints », address Prague (Czech Republic), June, 2001, note : Also available as Technical Report A01-R-223, LORIA, Nancy (France).

[CKL01a] H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *Matching Power.* éditeurs A. MIDDELDORP., in « 12th International Conference on Rewriting Techniques and Applications (RTA'2001) », série Lecture Notes in Computer Science, volume 2051, Springer, pages 77-92, address Utrecht (The Netherlands), May, 2001, note : Also available as Technical Report A01-R-201, LORIA, Nancy (France).

[CKL01b] H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *The Rho Cube.* éditeurs F. HONSELL, M. MICULAN., in « 4th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'2001) », série Lecture Notes in Computer Science, volume 2030, Springer, pages 168-183, address Genova (Italy), April, 2001, note : Also available as Technical Report A01-R-202, LORIA, Nancy (France).

[CNNR02] H. COMON, P. NARENDRAN, R. NIEUWENHUIS, M. RUSINOWITCH. *Deciding the Confluence of Ordered Term Rewrite Systems.* 2002, note : à paraître. Also available as Technical Report A02-R-001, LORIA, Nancy (France).

[CV01a] Y. CHEVALIER, L. VIGNERON. *A Tool for Lazy Verification of Security Protocols (short paper).* in « Proceedings of ASE-2001: The 16th IEEE Conference on Automated Software Engineering », IEEE CS Press, address San Diego (CA), November, 2001, note : Long version available as Technical Report A01-R-

140, LORIA, Nancy (France).

[CV01b] Y. CHEVALIER, L. VIGNERON. *Towards Efficient Automated Verification of Security Protocols.* in « Proceedings of the Verification Workshop (VERIFY'01) (in connection with IJCAR'01), Università degli studi di Siena, TR DII 08/01 », pages 19-33, address Siena (Italy), June, 2001, note : Also available as Technical Report A01-R-046, LORIA, Nancy (France).

[DK01] N. DERSHOWITZ, C. KIRCHNER. *Inversion Strategies.* in « Third Workshop on Rule-Based Constraint Reasoning and Programming (RCoRP'01) », address Paphos (Cyprus), décembre, 2001, note : Also available as Technical Report A01-R-259, LORIA, Nancy (France).

[FGK01] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *Termination of rewriting with local strategies.* éditeurs M. P. BONACINA, B. GRAMLICH., série Electronic Notes in Theoretical Computer Science, numéro 58.2, Elsevier Science Publishers, 2001, note : Also available as Technical Report A01-R-177, LORIA, Nancy (France).

[JRV00] F. JACQUEMARD, M. RUSINOWITCH, L. VIGNERON. *Compiling and Verifying Security Protocols.* éditeurs A. V. MICHEL PARIGOT., in « Logic for Programming and Automated Reasoning, Reunion Island, France », série Lecture Notes in Computer Science, volume 1955, Springer Verlag, novembre, 2000.

[Kir01] C. KIRCHNER. *ELAN : Modélisation et preuve en calcul de réécriture (présentation invitée).* in « Approches Formelles dans l'Assistance au Développement de Logiciels », address Nancy (France), June, 2001, note : Also available as Technical Report A01-R-242, LORIA, Nancy (France).

[MR01a] J. MUSSET, M. RUSINOWITCH. *An effective fixed point calculus for deterministic systems applied to model checking.* in « First International Workshop on Automated Verification of Infinite-State Systems (AVIS'01) », address Berlin (Germany), March, 2001, note : Also available as Technical Report A01-R-157, LORIA, Nancy (France).

[MR01b] J. MUSSET, M. RUSINOWITCH. *Improved Subsumption for CLP-based Verification.* in « ICLP'2001 Workshop on (Constraint) Logic Programming and Software Engineering », organisation Gupta, G., address Paphos (Cyprus), December, 2001, note : Also available as Technical Report A01-R-243, LORIA, Nancy (France).

[MRV01] P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. *A Pattern-Matching Compiler.* éditeurs M. VAN DENBRAND, D. PARIGOT., in « First Workshop on Language Descriptions, Tools and Applications (LDTA'01) », série Electronic Notes in Theoretical Computer Science, numéro 44.2, Elsevier, address Genova (Italy), April, 2001, note : Also available as Technical Report A01-R-059, LORIA, Nancy (France).

[Ngu01a] Q.-H. NGUYEN. *Certifying Term Rewriting Proof in ELAN.* in « 2nd International Workshop on Rule-based Programming (RULE'01) », série Electronic Notes in Theoretical Computer Science, numéro 59.4, Elsevier Science Publishers, address Firenze (Italy), September, 2001, note : Also available as Technical Report A01-R-138, LORIA, Nancy (France).

[Ngu01b] Q.-H. NGUYEN. *Compact Normalisation Trace via Lazy Rewriting.* in « 1st International Workshop on Reduction Strategies in Rewriting and Programming (WRS'2001) », série Electronic Notes in Theoretical Computer Science, numéro 57, organisation Lucas, S. and Gramlich, B., Elsevier Science Publishers, address Utrecht (The Netherlands), May, 2001, note : Also available as Technical Report A01-R-139, LORIA, Nancy

(France).

[Rin01] C. RINGEISSEN. *Matching with Free Function Symbols – A Simple Extension of Matching?.* éditeurs A. MIDDELDORP., in « 12th International Conference on Rewriting Techniques and Applications (RTA'2001) », série Lecture Notes in Computer Science, volume 2051, Springer, pages 276-290, address Utrecht (The Netherlands), May, 2001, note : Also available as Technical Report A01-R-023, LORIA, Nancy (France).

[RT01a] M. RUSINOWITCH, M. TURUANI. *Deciding insecurity for non-atomic keys.* in « Dagstuhl Seminar on Specification and Analysis of Secure Cryptographic Protocols », organisation Basin, D. and Denker, G. and Lowe, G. and Millen, J., address Dagstuhl (Germany), July, 2001, note : Also available as Technical Report A01-R-245, LORIA, Nancy (France).

[RT01b] M. RUSINOWITCH, M. TURUANI. *Protocol Insecurity with Finite Number of Sessions is NP-complete.* in « 14th IEEE Computer Security Foundations Workshop », address Cape Breton, Nova Scotia (Canada), June, 2001, note : Also available as Technical Report A01-R-051, LORIA, Nancy (France).

[Rus01a] M. RUSINOWITCH. *Réécriture et vérification.* in « Journées Systèmes Infinis », organisation Bouajjani, A., address Paris (France), March, 2001, note : Also available as Technical Report A01-R-246, LORIA, Nancy (France).

[Rus01b] M. RUSINOWITCH. *Rewriting for Deduction and Verification (invited talk).* éditeurs A. MIDDELDORP., in « 12th International Conference on Rewriting Techniques and Applications (RTA'2001) », série Lecture Notes in Computer Science, volume 2051, Springer, address Utrecht (The Netherlands), May, 2001, note : Also available as Technical Report A01-R-049, LORIA, Nancy (France).

[Rus01e] M. RUSINOWITCH. *Vérification automatique de protocoles cryptographiques avec CASRUL.* in « Journées Systèmes et Logiciels Critiques », address Grenoble (France), November, 2001, note : Also available as Technical Report A01-R-247, LORIA, Nancy (France).

[Stu01] J. STUBER. *A Model-based Completeness Proof of Extended Narrowing And Resolution.* éditeurs R. GORE, A. LEITSCH, T. NIPKOW., in « 1st International Joint Conference on Automated Reasoning (IJCAR-2001) », série Lecture Notes in Computer Science, volume 2083, Springer, pages 195-210, address Siena (Italy), June, 2001, note : Also available as Technical Report A01-R-068, LORIA, Nancy (France).

Rapports de recherche et publications internes

[Fau01] G. FAURE. *Etude des proprietes du calcul de reecriture : du rho calcul au rhoEpsilon calcul.* Stage du Magistere Informatique et Modelisation, septembre, 2001.

[GKF01] I. GNAEDIG, H. KIRCHNER, O. FISSORE. *Induction for innermost and outermost ground termination.* Research Report, numéro A01-R-178, institution LORIA, Nancy (France), September, 2001.

[KQSS01] C. KIRCHNER, Z. QIAN, P.-K. SINGH, J. STUBER. ***Xemantics**: A semantics of XSLT in **ELAN**.* rapport de recherche, institution LORIA, 2001.

Bibliographie générale

[AC96] M. ABADI, L. CARDELLI. *A Theory of Objects.* Springer Verlag, 1996.

[BBR98] N. BERREGEB, A. BOUHOULA, M. RUSINOWITCH. *Observational Proofs with Critical Contexts.* in « Fundamental Approaches to Software Engineering - ETAPS'98, Lisboa, Portugal », série Lecture Notes in Computer Science, volume 1382, Springer-Verlag, pages 38-53, avril, 1998.

[BCD+00] P. BOROVANSKÝ, H. CIRSTEA, H. DUBOIS, C. KIRCHNER, H. KIRCHNER, P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. ***ELAN** V 3.4 User Manual.* organisation LORIA, édition fourth, address Nancy (France), janvier, 2000.

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

[BR95] A. BOUHOULA, M. RUSINOWITCH. *Implicit Induction in Conditional Theories.* in « Journal of Automated Reasoning », numéro 2, volume 14, 1995, pages 189–235.

[Cir00] H. CIRSTEA. *Calcul de réécriture : fondements et applications.* Thèse de Doctorat d'Université, Université Henri Poincaré - Nancy I, 2000.

[CK00] H. CIRSTEA, C. KIRCHNER. *The Simply Typed Rewriting Calculus.* in « 3rd International Workshop on Rewriting Logic and its Applications », Electronic Notes in Theoretical Computer Science, address Kanazawa, Japan, September, 2000.

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

[DHK98] G. DOWEK, T. HARDIN, C. KIRCHNER. *Theorem Proving Modulo.* Rapport de Recherche, numéro 3400, institution Institut National de Recherche en Informatique et en Automatique, April, 1998, http://www.inria.fr/rrrt/rr-3400.html

[DM99] N. DERSHOWITZ, S. MITRA. *Jeopardy.* éditeurs P. NARENDRAN, M. RUSINOWITCH., in « Proceedings of RTA'99 », série Lecture Notes in Computer Science, volume 1631, Springer-Verlag, pages 16–29, July, 1999.

[DST80] P. J. DOWNEY, R. SETHI, R. E. TARJAN. *Variations on the Common Subexpressions Problem.* in « Journal of the Association for Computing Machinery », numéro 4, volume 27, 1980, pages 758–771.

[FHM94] K. FISHER, F. HONSELL, J. C. MITCHELL. *A Lambda Calculus of Objects and Method Specializatio n.* in « Nordic Journal of Computing », numéro 1, volume 1, 1994, pages 3–37.

[FKW00] W. FOKKINK, J. KAMPERMAN, P. WALTERS. *Lazy Rewriting on Eager Machinery.* in « ACM Transactions on Programming Languages and Systems », numéro 1, volume 2, January, 2000, pages 45–86.

[JK91] J.-P. JOUANNAUD, C. KIRCHNER. *Solving equations in abstract algebras: a rule-based survey of unification.* éditeurs J.-L. LASSEZ, G. PLOTKIN., in « Computational Logic. Essays in honor of Alan Robinson », The MIT press, address Cambridge (MA, USA), 1991, chapitre 8, pages 257–321.

[KB70] D. E. KNUTH, P. B. BENDIX. *Simple word problems in universal algebras.* éditeurs J. LEECH., in « Computational Problems in Abstract Algebra », Pergamon Press, address Oxford, 1970, pages 263–297.

[McC97] W. MCCUNE. *Solution of the Robbins Problem.* in « JAR », numéro 3, volume 19, 1997, pages 263–276.

[Mes92] J. MESEGUER. *Conditional rewriting logic as a unified model of concurrency.* in « TCS », numéro 1, volume 96, 1992, pages 73–155.

[NO80] G. NELSON, D. OPPEN. *Fast Decision Procedures Based on Congruence Closure.* in « Journal of the Association for Computing Machinery », numéro 2, volume 27, avril, 1980, pages 356–364.

[RSK00] M. RUSINOWITCH, S. STRATULAT, F. KLAY. *Mechanical Verification of an Ideal ABR Conformance Algorithm.* éditeurs E. A. EMERSON, A. P. SISTLA., in « Conference on Computer Aided Verification, Chicago, USA », volume 1770, Springer, juillet, 2000.

[Sho84] R. E. SHOSTAK. *Deciding Combinations of Theories.* in « Journal of the Association for Computing Machinery », numéro 7, volume 21, 1984, pages 583–585.

[Stu00] J. STUBER. *Deriving Theory Superposition Calculi from Convergent Term Rewriting Systems.* éditeurs L. BACHMAIR., in « Rewriting Techniques and Applications (RTA), Norwich, UK », série LNCS, volume 1833, Springer, pages 229-245, juillet, 2000.

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