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]

**Assistante de projet**
Sophie Drouot [jusqu'au 31/03/2002]
Chantal Llorens [à partir du 1/04/2002]

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

**Personnel CNRS**
Hélène Kirchner [DR, à temps partiel]

**Personnel Université**
Horatiu Cirstea [Maître de Conférences, Université Nancy 2]
Carlos Castro [Maître de Conférence associé, UHP, 3 mois]

**Spécialiste étranger INRIA**
Mark van den Brand [Centrum voor Wiskunde en Informatica]

**Ingénieur associé INRIA**
Julien Guyon [à partir du 1/11/2002, 1 an]

**Chercheurs doctorants**
Clara Bertolissi [allocataire MNERT, à partir du 1/10/2002]
Alberto Ciaffaglione [bourse Italienne, en cotutelle INPL-université d'Udine]
Eric Deplagne [ATER, jusqu'au 15/11/2002]
Olivier Fissore [boursier BDI CNRS - Région Lorraine]
Liliana Ibănescu [boursier INPL - Peugeot]
Paulin Jacobé de Naurois [Normalien, Thèse co-encadrée par les projets CALLIGRAMME et PROTHEO]
Fabrice Nahon [Professeur de l'enseignement secondaire, à partir du 1/09/2002]
Quang-Huy Nguyen [boursier INRIA, jusqu'au 31/08/2002]
Benjamin Wack [Normalien, à partir du 1/09/2002]

**Chercheurs post-doctorants**
Horatiu Cirstea [ATER, jusqu'au 31/08/2002]
Eric Deplagne [ATER, à partir du 16/11/2002]
Hubert Dubois [ATER, jusqu'au 31/08/2002]
Quang-Huy Nguyen [ATER, à partir du 1/09/2002]
Jürgen Stuber [Post-doc INRIA, ATER, à partir du 1/09/2002]

**Chercheurs invités**
Anamaria Martins Moreira [à partir du 1/04/2002, 1 an]
Jurgen Vinju [du 1/09/2002 au 30/11/2002]

**Stagiaires**
Aouadi Bacem [ENSI Tunis, 4 mois]
Clara Bertolissi [DEA Nancy I, Université Udine, 8 mois]
Issam Chebbi [ENSI Tunis, 4 mois]
Mathieu Hoyrup [ENS Lyon (Magister), 2 mois]

Anderson Santana [Universidade Federal do Rio Grande do Norte, 4 mois]
Benja Wack [ENS Lyon (DEA), 5 mois]

# 2. Présentation et objectifs généraux

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

Nos recherches s'appuient sur trois domaines scientifiques : contraintes, réécriture et mécanisation du raisonnement. 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 **ELAN** 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 la coopération entre assistants de preuve et procédures de décisions et sur les preuves par récurrence, d'autre part sur l'intégration dans les démonstrateurs de contraintes permettant de restreindre l'espace de recherche.

Les problématiques de la mécanisation du raisonnement et de la résolution de contraintes alimentent également des recherches sur la complexité des calculs et des problèmes de décision.

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.

Des évolutions majeures ont eu lieu ces deux dernières années avec l'émergence à partir du projet Protheo de deux nouvelles équipes : Modbio en 2001 sous la direction d'Alexander Bockmayr et Cassis en 2002 sous l'impulsion de Michael Rusinowitch. Il en résulte une adaptation des contours scientifiques du projet que l'on retrouve dans la description de nos résultats.

# 3. Fondements scientifiques

## 3.1. 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 formel 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. Le calcul de réécriture nous donne le cadre général sur lequel fonder ces travaux.

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 [63] 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 mécanisation du raisonnement pour simplifier les espaces de recherche, ou pour inclure des procédures de décision de l'égalité dans des démonstrateurs plus généraux.

Un 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 expressif pour 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.

La réécriture est enfin un paradigme fondamental de description de transformations, que celles-ci soit fonctionnelles ou non. A partir de nos travaux sur la réécriture et les stratégies a émergé un nouveau concept généralisant le lambda calcul et la réécriture et que nous avons appelé le calcul de réécriture ou rho calcul. Nous explorons actuellement les capacités et les propriétés de différentes instances de ce calcul, tout particulièrement dans ses versions typées où la généralisation de l'isomorphisme de deBruijn-Curry-Howard reste à comprendre.

## 3.2. 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[60]. 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.

## 3.3. Mécanisation du raisonnement

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

Nous développons des méthodes et des systèmes de recherche de preuve fondés sur la réécriture et la résolution de contraintes. Ces méthodes sont appliquées à la mécanisation du raisonnement en logique équationnelle, en logique du premier ordre et en logique d'ordre supérieure.

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.

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 [56][65] 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.

# 4. Domaines d'application

**Mots clés :** *modélisation*, *prototypage*, *vérification*, *logiciel de télécommunications*, *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, la vérification et la transformation de

programmes, en particulier de télécommunication (protocoles et services). A plus long terme, nous voulons appliquer nos techniques à la modélisation de systèmes réactifs et de systèmes physiques hybrides ayant des comportements discrets et continus.

# 5. Logiciels

## 5.1. ELAN

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

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

**ELAN** [51] 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 système **ELAN** inclut un interpréteur, un compilateur, une bibliothèque de support d'exécution, un ramasse miettes, une bibliothèque de gestion du *backtracking* ainsi qu'une bibliothèque de programmes standard, des exemples d'applications et un manuel d'utilisation [49].

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

## 5.2. TOM

**Mots clés :** *filtrage*, *compilation*.

**Participants :** Pierre-Etienne Moreau [correspondant], Christophe Ringeissen.

**TOM** [46] [67] est un outil de compilation du filtrage qui peut être vu comme un préprocesseur capable de compiler des constructions de filtrage. Une de ses particularités est d'être indépendant du langage natif : **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 : **TOM** accède aux termes en utilisant une interface commune permettant de le rendre indépendant de la représentation concrète. Ces caractéristiques font de **TOM** un outil idéal pour implanter des systèmes de transformation de termes et plus particulièrement les langages fondés sur la notion de règles de réécriture (ASF+SDF, **ELAN** et Stratego par exemple).

# 6. Résultats nouveaux

## 6.1. Calcul de réécriture

**Mots clés :** *réécriture*, *stratégie*.

A partir de nos travaux précédents [50], 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 $\lambda$-calcul et la réécriture que nous avons appelé le $\rho$-calcul ou le calcul de réécriture [53]. Dans ce calcul, l'opérateur d'abstraction ainsi que l'opérateur d'application sont

des objets du calcul. Une $\rho$-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 $\rho$-abstraction ou d'un $\rho$-terme plus élaboré) est un $\rho$-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.

Cette année nous a permis d'avancer dans l'étude des calculs fonctionnant suivant ces principes, y compris dans leurs versions typés, et de comparer notre approche aux cadres de réécriture d'ordre supérieur.

### 6.1.1. Rho-calcul et CRS

**Participants :** Clara Bertolissi, Horatiu Cirstea, Claude Kirchner.

Nous avons démontré que le $\lambda$-calcul et la réécriture sont des cas particuliers du $\rho$-calcul, dans le sens où la syntaxe et les règles d'inférence du $\rho$-calcul peuvent être restreintes afin d'obtenir les deux autres calculs.

Puisque le $\lambda$-calcul et la réécriture ont des caractéristiques complémentaires, l'unification de ces deux formalismes a été étudiée dans différents contextes. Cette intégration est réalisée soit en enrichissant la réécriture de premier ordre avec des fonctionnalités d'ordre supérieur (*e.g.* les *CRS* [62]), soit en ajoutant au $\lambda$-calcul des caractéristiques algébriques [61].

Dans le cadre de l'analyse de l'expressivité du $\rho$-calcul, nous avons analysé la correspondance entre le $\rho$-calcul et les relations de réécriture d'ordre supérieur. En particulier, nous montrons que pour toute réduction dans un *CRS* calcul de réécriture et nous fournissons une traduction directe des *CRS* dans le $\rho$-calcul [39].

### 6.1.2. Un traitement d'exception en calcul de réécriture

**Participants :** Horatiu Cirstea, Germain Faure, Claude Kirchner.

Pour nous permettre d'exprimer la stratégie `first` d'**ELAN**, nous avons étudié 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 dans lequel le `first` est exprimable (et par conséquent les stratégies d'évaluation le sont aussi) [31].

Cette approche nous permet de considérer l'échec comme une exception et d'implanter un mécanisme permettant l'interception et le traitement de ces exceptions.

Néanmoins, puisque tout comportement erroné est représenté de la même manière, on ne peut pas effectuer un traitement différent en fonction du type d'exception interceptée. Nous avons introduit par ailleurs une nouvelle construction permettant de distinguer entre les différentes exceptions [23] et nous avons proposé différentes sémantiques (*big-step*) pour le calcul ainsi obtenu.

### 6.1.3. Calculs de réécriture typés

**Participants :** Horatiu Cirstea, Benjamin Wack, Claude Kirchner.

Nous avons étudié une version simplement typée du calcul de réécriture [52] mais nous nous intéressons également à des système de types plus élaborés et nous avons proposé un $\rho$-cube généralisant le $\lambda$-cube de Barendregt.

Dans un premier temps, nous avons proposé et étudié avec Gilles Barthe et Luigi Liquori [19] un cadre que nous avons appelé « Pure Pattern Type System » (ou $P^2TS$) et qui est une extension algébrique des « Pure Type System ». Dans les $P^2TS$, les règles de réécriture peuvent être considérées comme des lambda termes avec motifs et l'application de règles est l'application du lambda-calcul avec des possibilités de filtrage. Ce contexte propose d'une manière complètement unifiée des fonctionnalités d'ordre supérieur et des mécanismes de filtrage ainsi qu'un système de types puissant et donc, il est particulièrement bien adapté pour les fondations des (meta)langages de programmation et des assistants de preuves. Nous avons prouvé certaines propriétés standard telles que la préservation des types et la confluence et nous conjecturons également la normalisation forte des termes bien typés.

Les opérateurs d'abstraction pour les termes et les types sont séparés dans les $P^2TS$ mais compte tenu de la puissance d'expression de l'abstracteur « $\longrightarrow$ » nous avons proposé un système de types plus élaboré où l'abstraction sur les types est également décrite par réécriture [55]. En unifiant les symboles d'abstraction pour les termes et les types du lambda-calcul dans le symbole d'abstraction du calcul de réécriture, nous perdons

l'unicité du type mais nous caractérisons des termes ayant un nombre de types arbitraire mais fini. Néanmoins, nous montrons l'unicité du type pour certains systèmes particuliers, tel que le $\rho$-calcul polymorphe [24]. Nous montrons également que les types sont préservés par la réduction pour tous les systèmes de types.

Nous considérons ce travail comme une contribution à l'étude de la correspondance entre réécriture et logique. Dans ce contexte, on peut associer aux $\rho$-termes des programmes basés sur la réécriture et nous voulons trouver une logique appropriée et un isomorphisme à la Curry-Howard entre les $\rho$-termes typés et cette logique.

## 6.2. Environnement de programmation par réécriture

**Mots clés :** .

Les techniques de réécriture que nous concevons ou utilisons dans le projet nous conduisent à être moteur dans leur implantation et leur mise en œuvre. Pour ce faire, nous concevons et développons une architecture atomique et modulaire nous permettant une approche à la « lego » c'est à dire reposant sur l'intégration de briques modulaires, facilement maintenables et utilisable en dehors du projet.

Dans ce cadre, nous avons commencé à isoler et étudier deux briques fondamentales : les structures de termes complètement partagés et le filtrage. Ce dernier joue un rôle critique, et son intégration dans des langages comme JAVA, C ou Eiffel est maintenant possible grâce aux techniques de compilation non-intrusive développées pour **TOM**.

Par ailleurs, l'évolution du langage **ELAN** et de son implantation depuis 1992, nous a conduit à réaliser une évolution majeure du système nous permettant de mieux maîtriser son évolution ultérieure et l'intégration de nouvelles capacités comme les stratégies probabilistes. Le premier prototype **ELAN** 4 est maintenant opérationnel.

### *6.2.1. Compilation non-intrusive du filtrage*

**Participants :** Pierre-Etienne Moreau, Christophe Ringeissen.

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 impératifs tels que C, Java ou Eiffel. Ce nouveau formalisme, présenté dans [46], fait l'objet d'une implantation diffusée. 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. Enfin, une troisième caractéristique est de supporter la compilation du filtrage équationnel et plus particulièrement le filtrage associatif avec élément neutre.

Ces caractéristiques font de **TOM** un outil idéal pour décrire des transformations de programmes ou compiler tout langage fondé sur la notion de règles de réécriture (ASF+SDF, **ELAN** et Stratego par exemple).

### *6.2.2. ATerms et glanage de cellules*

**Participant :** Pierre-Etienne Moreau.

Les ATerms (Annoted Terms) correspondent à un type de données abstrait permettant de représenter des structures arborescentes, et plus particulièrement des termes. La bibliothèque des ATerms est originellement développés par une équipe du CWI à Amsterdam. Une particularité de cette bibliothèque est de proposer une implantation en C avec une gestion automatique de la mémoire. D'un point de vue pratique, les termes sont représentés avec un partage maximum, ce qui permet de réduire l'espace mémoire nécessaire à leur représentation et de permettre leur comparaison en temps constant. En collaboration avec Olivier Zendra, nous avons proposé un nouvel algorithme de ramasse miettes à générations exploitant les caractéristiques liés au partage maximum. Le temps passé dans le gestionnaire de mémoire est ainsi réduit par un facteur 3 environ [47].

### *6.2.3. ELAN 4*

**Participants :** Mark van den Brand, Claude Kirchner, Pierre-Etienne Moreau.

L'évolution du langage **ELAN** et de son implantation depuis 1992 au cours de la thèse de Marian Vittek, nous a conduit à réaliser une évolution majeure du système nous permettant de mieux maîtriser sa conception, son évolution ultérieure et l'intégration de nouvelles capacités comme les stratégies probabilistes.

Dans ce cadre et en collaboration étroite avec le CWI à Amsterdam, une nouvelle version d'**ELAN** a été mise au point [38]. Dans un premier temps la syntaxe du langage a été simplifiée pour permettre une écriture plus simple des programmes. Puis une implantation a été réalisée dans le cadre du méta-environnement développé au CWI [34][35]. L'analyse est maintenant assurée par SDF et les communications entre composants par le ToolBus. Un interprète spécifique a été développé à partir de travaux antérieurs sur l'utilisation des A-termes.

Nous prévoyons de diffuser cette nouvelle version du système, intégrée à une interface utilisateur évoluée (voir figure 1), dès le début 2003.

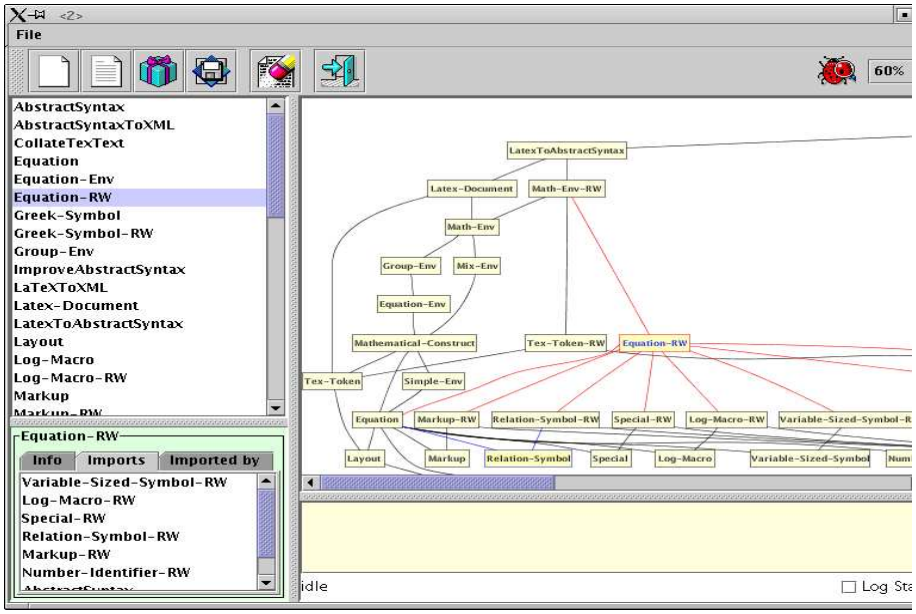


*Figure 1. Interface utilisateur d'**ELAN**4*

### *6.2.4. Manipulation de spécifications algébriques avec des représentations de graphes et de termes*

**Participants :** Anamaria Martins Moreira, Christophe Ringeissen, Anderson Santana.

Dans le cadre du projet FERUS, nous développons un outil qui intègre une collection d'opérations sur des spécifications algébriques écrites dans le langage CASL. L'objectif de FERUS comprend à la fois la modification de spécifications existantes, la création ou la dérivation de nouvelles spécifications, tout comme leur preuve et leur exécution qui sont réalisées par l'interoperabilité avec des outils existants.

Puisque FERUS implique la manipulation de spécifications et l'interopérabilité avec d'autres outils, la question du choix des formats de représentation est importante. Dans [45], nous discutons les avantages et limitations des ATerms comme format d'échange et de manipulation dans le cadre de FERUS. On utilise également un nouveau format à base de graphes, qui offre des caractéristiques complémentaires à un format à base de termes. Des outils de visualisation sont présentés pour ces formats.

## 6.3. Réécriture et stratégies

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

Le cadre de base sur lequel nous fondons maintenant notre conception de l'environnement de réécriture pour la preuve et la programmation est le calcul de réécriture, décrit dans la section 6.1.

Cette section montre d'une part comment étendre la notion de stratégies à des situations probabilistes ou floues, d'autre part comment utiliser **ELAN** pour formaliser la programmation objet et pour transformer des termes dans un contexte XML, et enfin comment dégager de nos expériences de programmation en particulier de prouveurs et de solveurs de contraintes une méthodologie de la programmation par règles.

### *6.3.1. Réécriture en présence de choix probabilistes ou de flou*

**Participants :** Olivier Bournez, Mathieu Hoyrup, Claude Kirchner.

Depuis plusieurs années les langages à base de règles de réécriture se concentrent sur l'utilisation de la réécriture en tant qu'outil de modélisation. Les langages à base de règles permettent ainsi de prototyper facilement et efficacement de nombreux systèmes, comme par exemple les règles de déduction d'une logique ou encore les règles d'évolution d'un système réactif [66].

Pour étendre les possibilités de modélisation des langages à base de règles, nous avons exploré la possibilité de rendre le déclenchement des règles de réécriture sujet à des choix probabilistes.

Nous avons ainsi proposé dans [22] l'ajout d'un opérateur de choix probabiliste aux langages à base de règles comme le système **ELAN**, et montré que cet opérateur donnait un moyen naturel de spécifier les algorithmes randomisés ou tous les systèmes sujets à des choix probabilistes.

L'utilisation de systèmes à base de règles qui peuvent être probabilistes pose alors de nombreux problèmes théoriques sur la compréhension des notions sous-jacentes. Dans [22], nous avons proposé une notion de système probabiliste abstrait de réductions qui généralise les systèmes abstraits de réductions au cas probabiliste. Nous avons introduit des notions comme la confluence probabiliste, la confluence presque sûre et prouvé plusieurs résultats sur les relations existantes entre les différentes notions.

L'étude de la logique de réécriture en présence de choix probabilistes à été amorcée par le stage de première année de l'École Normale Supérieure de Lyon de Mathieu Hoyrup. Nous avons mis en évidence plusieurs résultats soulignant les difficultés derrière la mise au point d'une logique de réécriture probabiliste, et proposé un cadre restreint permettant de définir une telle logique [44].

En cherchant à comprendre s'il était possible de construire une théorie de la preuve en présence d'équations avec probabilités, nous avons mis en évidence qu'il était en fait possible de construire une théorie de la preuve équationnelle dans un cadre distinct, celui de la logique floue. Nous avons ainsi montré dans [20][40] qu'il était possible de construire une théorie de la preuve valide et complète pour la preuve équationnelle en présence de flou.

### *6.3.2. ELAN et la logique de réécriture*

**Participants :** Claude Kirchner, Pierre-Etienne Moreau, Hélène Kirchner.

**ELAN** implémente la notion de système de calcul, un concept qui combine deux entités de première classe : les règles de réécriture et leurs stratégies d'application.

En complément de nos travaux récents sur le calcul de réécriture, nous avons terminé l'étude de l'utilisation de la logique de réécriture pour donner une sémantique au langage [16].

### *6.3.3. Réécriture et objets*

**Participants :** Hubert Dubois, Hélène Kirchner.

Dans sa thèse [59], 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 [28] que cette extension objet du langage **ELAN** respecte la sémantique donnée par le $\rho$-calcul, en s'appuyant sur une version particulière du calcul de réécriture, le $\rho$-calcul objet [54].

Nous avons défini pour chaque classe et pour chaque objet, une représentation algébrique, donnée par une signature et des règles. Une correspondance établie entre $\rho$-termes et termes représentant des objets, et une correspondance entre l'évaluation par réécriture de ces termes et l'évaluation dans le $\rho$-calcul nous ont permis

de montrer qu'un sous-ensemble du $\rho$-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** [29].

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.

### *6.3.4. Méthodologie de la programmation par règle*

**Participants :** Claude Kirchner, Carlos Castro.

La programmation par règles a été développée depuis les années 70. Les langages récents basés sur la réécriture de terme tels qu'**ELAN**, ASF [17] Maude ou Stratego nécessitent de développer une méthodologie adaptée pour la conception et l'écriture de programmes. De plus dans les langages disposant d'un langage se stratégies évolué comme **ELAN**, la description du contrôle et des données peut être raisonnablement séparée, ce qui offre un paradigme de programmation déclarative.

Nous mettons au point dans ce cadre une méthodologie de programmation qui prenne en compte les spécificités de la programmation par règle. Cette méthodologie est mise en œuvre sur des exemples classiques d'algorithmique avec pour objectif de faciliter dans une phase ultérieure leur preuve de correction de complétude ou de terminaison. Nos premiers résultats dans ce sens sont présentés dans [42].

### *6.3.5. Latex2MathML*

**Participants :** Mark van den Brand, Gaurav Kwatra, Claude Kirchner, Jürgen Stuber.

Nous avons examiné si et de quelle façon il est possible d'analyser des formules mathématiques de documents écrits en LaTeX, puis de les convertir en d'autres formats, par exemple MathML, ceci en utilisant la nouvelle version d'**ELAN**, **ELAN**4.

MathML est une instance de XML conçu pour représenter des formules mathématiques. Il existe en deux variantes, MathML pour la présentation par exemple par un butineur comme Mozilla (Présentation MathML), et MathML pour la représentation du contenu sémantique. Au dessus de MathML existent d'autres standards, par exemple OMDOC, pour représenter globalement des documents mathématiques.

Afin de traiter un exemple de taille significative et correspondant à un besoin réel, nous avons utilisé en entrée un chapitre sur les fonctions d'Airy de la Bibliothèque digitale des fonctions mathématiques (Digital Library of Mathematical Functions, DLMF), qui nous était fourni gracieusement par l'institut national des standards des États-Unis (National Institute of Standards, NIST http://www.nist.gov). Le DLMF fait partie d'un projet de réédition du livre *Handbook of Mathematical Functions* de Abramowitz et Stegun. Ce chapitre contient surtout des formules arithmétiques et analytiques sur les nombres réels et complexes, avec des sommes, des différentielles et des intégrales. Le format est assez bien cadré, et les auteurs utilisent des macros LaTeX pour tous les fonctions spéciales traitées dans le livre.

Même dans un tel contexte la tâche reste difficile, du fait qu'en mathématiques des conventions syntaxiques assez évoluées sont utilisées. Par exemple, il n'est pas toujours simple de décider si une juxtaposition de deux symboles comme $fx$ est une multiplication ou une application de fonction.

**ELAN**4 est assez bien adapté à la tâche d'analyse de syntaxes élaborées du fait qu'il utilise le parseur LR généralisé sans scanner (scannerless generalized LR parser, SGLR) de SDF qui a été développé au CWI à Amsterdam. Nous avons choisi l'approche des grammaires d'îles (island grammars), qui utilisent le mécanisme de préférence de SDF pour analyser uniquement les îles de l'entrée représentant les formules mathématiques.

Nous avons réalisé l'analyse syntaxique de la plus grande partie du chapitre Airy. La grammaire est d'une taille importante avec plusieurs centaines de règles, dont environ la moitié peut être caractérisée comme des règles dictionnaires, par exemple pour toutes les fonctions. Les conventions plus évoluées pour l'omission de parenthèses se manifestent par une prolifération de règles en particulier pour la multiplication, où la

structure dépend aussi des opérateurs et fonctions. Un exemple typique est l'expression $x \sum y \sin z$ qui doit être comprise comme $x \cdot (\sum (y \cdot (\sin z)))$.

Après l'analyse syntaxique nous utilisons plusieurs systèmes de réécriture pour réaliser les différentes étapes de la conversion. Nous commençons par une réécriture de la syntaxe concrète LaTeX en une syntaxe abstraite plus régulière pour faciliter les étapes suivantes de réécriture vers le document MathML.

Actuellement nous ne traitons pas les cas suivants : (1) les compositions de formules des environnements mathématiques LaTeX comme par exemple lorsque plusieurs équations sont données dans une seule paire de $. Ceci peut être facilement corrigé dans l'entrée. (2) quelques constructions « maison » inventés par les rédacteurs, ce qui peut être corrigé par le NIST en ajoutant de nouvelles macros. (3) les équations entre fonctions (au lieu de nombres), ce qui nécessitera une extension de tous les opérateurs aux fonctions et augmentera la taille de la description de la grammaire. (4) des constructions exotiques comme une intégrale itérée $n$ fois dont la sémantique reste à éclaircir.

En résumé ce travail est un premier pas qui montre la faisabilité et l'intérêt de l'utilisation de langage à base de règles tels qu'**ELAN**4 pour décrire de façon fructueuse de telles transformations. Il reste bien sur un gros travail d'ingénierie pour finaliser ce travail.

## 6.4. Mécanisation du raisonnement

**Mots clés :** *déduction modulo*, *procédures de décision*, *théories contraintes*, *preuve par récurrence*, *preuve équationnelle*, *procédure de complétion*, *terminaison*, *propriété observable*.

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 développé d'une part de nouvelles techniques de preuves de propriétés utilisant la réécriture avec stratégies et d'autre part étudié pour elle même la terminaison de la réécriture avec stratégies.

### *6.4.1. Réécriture de formules CTL*

**Participants :** Anamaria Martins Moreira, Christophe Ringeissen.

La complexité des algorithmes conventionnels pour le model-checking de formules CTL dépend de la taille des formules, et plus précisément du nombre d'opérateurs de point-fixe. Dans [30], nous abordons les questions suivantes : Etant donnée une formule CTL, est-il possible de trouver une formule équivalente avec moins d'opérateurs de point fixe ? Et comment des techniques de réécriture peuvent-elles être appliquées pour déterminer cette formule plus simple ? De plus, pour certaines sous-logiques de CTL, comme par exemple la sous-logique NFCTL introduite par David Deharbe et Anamaria Martins Moreira, des algorithmes de vérification plus efficaces sont connus. La classe de formules NFCTL est définie syntaxiquement par induction structurelle. Afin d'accroître l'applicabilité des méthodes de vérification dédiées à NFCTL, nous considérons également le problème de l'appartenance sémantique d'une formule à NFCTL, pour déterminer l'existence d'une formule équivalente qui soit syntaxiquement une formule de NFCTL. Dans ce but, nous proposons et étudions un système de réécriture modulo $AC$, et nous discutons sa mise en œuvre en **ELAN**, en montrant comment intégrer ce processus de simplification basé sur la réécriture dans un outil de vérification formelle.

### *6.4.2. Tactique de réécriture ELAN en Coq*

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

Nous avons proposé une nouvelle approche dans la coopération **Coq/ELAN**, appelée *externe*, où les calculs sont effectués entièrement en **ELAN** qui retourne ensuite un terme de preuve en syntaxe **Coq**. Ce terme de preuve est certifié par le noyau **Coq** afin d'assurer la correction de ces calculs. Pour cela, nous avons d'abord proposé une représentation formelle de la trace des calculs **ELAN** en utilisant le calcul de réécriture avec substitutions explicites ($\rho\sigma$-calcul). Cette représentation est appelée le *terme de preuve* et fournit à la trace des calculs **ELAN** une sémantique « calculatoire », au sens où un terme de preuve doit permettre de reconstituer son calcul correspondant en utilisant les règles d'évaluation du $\rho\sigma$-calcul. En se basant sur cette sémantique, nous pouvons démontrer des propriétés d'équivalence entre les termes de preuve et les réduire en une forme

canonique compacte. Par ailleurs, nous pouvons également traduire les termes de preuve en syntaxe d'autres cadres logiques. Une traduction vers la syntaxe **Coq** a été décrite avec une démonstration de sa correction. Ce travail permet donc de transférer de façon sûre les calculs **ELAN** en **Coq**.

Afin d'accélérer la vérification de la réécriture modulo AC en **Coq**, nous avons proposé une nouvelle procédure de décision constructive pour les égalités modulo AC. Cette procédure, implantée en **ELAN**, fournit pour chaque égalité modulo AC une preuve qui peut être traduite en un *script* **Coq** permettant de la vérifier. Ce travail est décrit dans [36].

Les résultats théoriques obtenus ont été partiellement implantés dans une nouvelle tactique de réécriture modulo AC. Nous avons évalué la performance de notre tactique et observé une amélioration significative en terme de temps par rapport à son homologue dans la distribution de **Coq**. L'approche externe et sa mise en œuvre sont décrites en détail dans [14] ainsi que [18].

Une des premières applications de notre tactique consiste à automatiser partiellement la preuve par récurrence en **Coq**. En effet, dans nombreuses preuves par récurrence, la démonstration du cas de base ainsi que du cas d'induction consiste simplement à normaliser le but par rapport aux égalités dans le contexte. Nous devons assurer également que l'hypothèse d'induction ne s'applique qu'à l'intérieur du but et non pas à la racine. Cette stratégie permet d'automatiser la preuve, à condition que la variable de récurrence soit indiquée par l'utilisateur.

### *6.4.3. Systèmes d'inférence canoniques abstraits*

**Participant :** Claude Kirchner.

La notion d'ordre noethérien joue un rôle fondamental en mécanisation du raisonnement car elle permet de restreindre énormément les espaces de recherche à explorer. Nous avons conçu avec Nachum Dershowitz de l'université de Tel Aviv une abstraction de la notion bien connue de la complétion de Knuth et Bendix qui fournit une procédure de semi-décision pour la logique équationnelle.

En fait la notion de compilation de la présentation d'une théorie, qu'elle soit équationnelle ou pas, est une notion très générale que l'on retrouve aussi bien en théorie de la preuve qu'en résolution de contrainte. Le cadre que nous développons est basé sur la connaissance d'un ordre bien fondé sur la notion de preuve ici considérée de façon abstraite. Nous donnons ainsi des définitions et des caractérisations abstraites des notions de saturation d'ensemble d'axiomes et de critères de redondance. Ces résultats ont été présentées dans [26].

### *6.4.4. Système de preuve modulo récurrence*

**Participants :** Eric Deplagne, Claude Kirchner, Jürgen Stuber.

La preuve par récurrence est une méthode de preuve fondamentale, tant pour les nombreuses preuves du domaine mathématique qui l'utilisent que parce que les propriétés que l'on souhaite prouver au sujet d'un programme ou de sa spécification sont souvent prouvables par récurrence. C'est également le cas pour les propriétés sur les protocoles, comme l'absence de possibilité d'intrusion, dont la preuve nécessite également la récurrence, sur la longueur des messages notamment.

Les méthodes de preuve par récurrence, et donc plus encore les systèmes qui les implémentent, sont très diverses. La méthode de preuve par récurrence noethérienne est générale et ses fondements sont clairs. Cependant son automatisation est difficile, ce qui a conduit à la conception d'assistants de preuve qui sont fortement voire complètement guidés par l'utilisateur. La méthode de preuve par récurrence par réécriture est au contraire par nature très automatique.

Le lien entre récurrence noethérienne et récurrence par réécriture est classiquement établi de façon sémantique. La thèse d'Eric Deplagne établit le lien au niveau des preuves à l'aide du formalisme de la déduction modulo. La preuve par récurrence par réécriture est ainsi vue comme le résultat de l'internalisation en déduction modulo des hypothèses de récurrence [13][25].

Pour parvenir à établir ce lien, le formalisme de la déduction modulo est étendu au traitement de congruences exprimées à l'aide de règles et d'équations conditionnelles. L'évaluation de ces conditions nous a également amené à prendre en compte le contexte dans lequel est appliquée la congruence. Enfin, l'ordre de récurrence

ne pouvant pas être compatible avec la congruence, il a fallu le définir comme protecteur, c'est-à-dire bloquant l'application de la congruence.

Les résultats obtenus sur l'internalisation des hypothèses de récurrence permettent d'expliquer certains comportements de la méthode de récurrence par réécriture : l'utilisation d'un but comme hypothèse de récurrence pour la preuve d'un autre but et la possibilité de ne pas vérifier explicitement la condition de récurrence.

Le lien établi dans cette thèse, parce qu'il se situe au niveau des preuves, permettra aussi la coopération des systèmes dans un mode sceptique où la réponse n'est pas de type « oui » ou « non » mais un terme de preuve à partir duquel la correction de la preuve peut être vérifiée grâce à l'isomorphisme de Curry-Howard.

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 [69] 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 [68]. 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 litéraux 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.

### *6.4.5. Logique de lieur*

**Participant :** Claude Kirchner.

La notion de lieurs de variables se retrouve systématiquement en informatique comme en mathématiques. Par exemple on la retrouve dans l'expression de points fixes, de sommes ou de produits, d'intégrales ou de dérivées. Elle est souvent traitée via des encodages spécifiques (par exemple en syntaxe abstraite d'ordre supérieur), ce qui rend plus difficile son expression et les preuves de propositions dans ce contexte.

A partir de nos travaux précédents sur la notion de variable en logique d'ordre supérieur, nous avons défini une extension de la logique des prédicats appelée *logique de lieurs*, où les variables peuvent être liées dans les termes comme dans les propositions. Nous avons introduit une notion appropriée de modèles pour cette logique et montré les résultats de correction et de complétude de la logique par rapport aux modèles. Ces résultats sont obtenus par un encodage de la logique en logique du premier ordre reprenant de nos travaux précédents la notion de « pré-cuisson » d'un terme contenant des variables libres et liées [27].

### *6.4.6. Preuves de terminaison par induction*

**Participants :** Olivier Fissore, Isabelle Gnaedig, Hélène Kirchner.

Les travaux sur la terminaison pour la réécriture avec stratégies ont été poursuivis. Nous avons continué à développer des résultats sur une méthode originale de preuve par induction explicite que nous avions proposée en 1999. Nous établissons 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.

Notre effort a porté cette année sur le développement de cette méthode inductive dans le cas de la stratégie outermost, apportant pour la première fois une solution au problème de la terminaison spécifique de la réécriture outermost. Nous avons proposé une nouvelle définition du narrowing, et un mécanisme de renommage sémantique des variables des termes dérivés par la preuve, qui permettent de modéliser la réécriture outermost sur les termes clos. L'algorithme de preuve repose, comme dans le cas innermost et le cas de stratégies locales déjà traités, sur la satisfiabilité de contraintes d'ordre. Or, contrairement à ces

autres stratégies, on n'engendre pas de contraintes d'ordre tout au long de la preuve, mais seulement aux étapes finales des différents cas de cette preuve. La plupart du temps, ces contraintes peuvent être résolues automatiquement, soit par l'utilisation d'ordres simples comme l'ordre de plongement, soit par des solveurs de contraintes engendrant des ordres polynomiaux, ce qui rend la preuve complètement automatique. Ce travail a été publié au workshop WRLA'2002[33].

Nous avons également porté notre effort à la réalisation d'un atelier logiciel de preuve de terminaison des programmes à bases de règles. Cet atelier, CARIBOO, repose sur les algorithmes inductifs sus-cités, et a été développé avec le souci d'être spécifiquement adapté au contexte de la programmation, et accessible à des utilisateurs non experts. Il permet la preuve de terminaison innermost, sous stratégies locales et outermost. Sur bon nombre de cas, il est automatique. Sinon, le contexte théorique de notre principe inductif permet, pour des sous-preuves, d'intégrer la coopération d'autre outils permettant d'établir la terminaison d'un terme par un moyen quelconque, ou l'utilisation de solveurs de contraintes d'ordre indépendants. Une interface avec C*i*ME, solveurs de contraintes d'ordre par génération d'interprétations polynomiales développé au LRI, a d'ailleurs été réalisée dans le système. L'implantation a été réalisée en **ELAN** et permet la preuve de terminaison d'une sous-classe importante de programmes **ELAN**. Un article sur CARIBOO a été publié à la conférence PPDP'2002 [32].

### *6.4.7. Une boîte à outils pour la validation et la preuve de programmes à base de règles*

**Participants :** Olivier Fissore, Isabelle Gnaedig, Issam Chebbi.

Dans le cadre d'un stage de fin d'études du diplôme d'ingénieur de l'ENSI de Tunis, une boîte à outils pour la validation et la preuve de programmes à base de règles a été réalisée. Nous sommes partis du constat que très peu d'outils existaient pour prouver des propriétés sur les programmes à base de règles, que ceux que l'ont pouvait trouver, conçus pour la théorie de la réécriture en général n'étaient pas adaptés aux programmes, et qu'ils demandaient la plupart du temps une grande expertise de la part de l'utilisateur. Aussi nous avons développé un environnement flexible, pouvant inclure des outils de preuve de propriétés intrinsèques de programmes par règles : terminaison, complétude, confluence, atteignabilité...

Un premier prototype a été proposé, incluant CARIBOO, l'environnement de preuve de terminaison inductive sous stratégies développé par ailleurs dans l'équipe, divers tests de terminaison utilisant des ordres de simplification usuels, un mécanisme issu de nos travaux sur la terminaison par induction, permettant d'établir la terminaison d'un programme sur un terme donné, et un algorithme de vérification de complétude suffisante. Un premier noyau d'expertise a été proposé, qui permet d'aider l'utilisateur dans le choix des outils de preuve, et dans l'ordre avec lequel les preuves doivent être effectuées pour un maximum de cohérence et d'efficacité. La structure de ce prototype, incluant des outils réalisés en ELAN, a été développée en Java, et permet en particulier la validation de programmes **ELAN** [43].

## 6.5. Complexité

**Mots clés :** *Complexité*, *complexité implicite*, *modèle de Blum Shub et Smale*, *complexité sur les réels*.

Nous avons obtenu plusieurs résultats caractérisant syntaxiquement les classes de complexité dans le modèle de Blum Shub et Smale. Nous continuons également l'étude des algorithmes de résolution de contraintes utilisés en déduction automatique et réécriture, notamment la complexité des algorithmes de filtrage nécessaires pour le déclenchement de règles.

### *6.5.1. 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.

Le sujet de la thèse de Paulin de Naurois, coencadrée par Olivier Bournez et Jean-Yves Marion (Équipe Calligramme) a pour objet l'étude de la complexité dans ce modèle, et en particulier 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.

En 2002, nous avons obtenu une caractérisation des fonctions calculables, ainsi que des fonctions calculables en temps polynômial dans ce modèle en termes de fonctions récursives sûres. Ce résultat a été présenté dans [21] et généralise la caractérisation de Bellantoni et Cook dans [48] aux structures arbitraires.

Nous avons ultérieurement obtenu une caractérisation des fonctions calculables en temps parallèle polynômial en termes de fonctions récursives sûres avec substitutions dans le rapport de recherche [41]. Ce résultat généralise le résultats de [64] au cas des structures arbitraires.

### *6.5.2. Filtrage avec contextes stratifiés*

**Participant :** Juergen Stuber.

Le filtrage de contextes consiste à résoudre des équations entre un terme qui contient des variables de contexte et un terme clos. Par exemple, l'équation $f(X(h(a,y)))) = f(g(h(a,b),c)$ contient la variable de contexte $X$ et la variable objet $y$ et peut être résolu en mettant $X = g(...,c)$ et $y = b$. Cette forme de filtrage est intéressante pour accéder à des informations stockées par exemple dans des arbres XML.

Le filtrage avec contextes est connu comme étant NP-complet. Cependant, pour l'unification avec contextes, on connaît plusieurs restrictions pour lesquelles l'unification est décidable, comme les problèmes stratifiés et linéaires. Nous nous sommes posés la question de déterminer la complexité des problèmes de filtrage correspondants. Nous avons trouvé que le filtrage avec contextes linéaire, où chaque variable a une seule occurrence, est dans P, et peut être résolu en temps $O(n^3)$ et espace $O(n^2)$ par programmation dynamique. Par contre si autorise deux occurrences de variable, le problème devient alors NP-complet.

Le problème stratifié est également NP-complet en général, mais la restriction à des symboles de fonction monadiques est encore dans P avec temps $O(n^3)$ et espace $O(n^2)$.

Dans [37], nous donnons également un algorithme pour résoudre des problèmes de filtrage avec contextes généraux, et nous discutons quelques optimisations possibles.

# 7. Contrats industriels

## 7.1. CALIFE

**Participants :** Olivier Bournez, Claude Kirchner, Hélène Kirchner, Quang-Huy Nguyen.

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

L'objectif du projet CALIFE, est de développer un environnement capable de supporter la spécification, la vérification formelle et la génération de tests de composants logiciels destinés à garantir la qualité des algorithmes critiques dans les réseaux de télécommunications. Les partenaires industriels impliqués dans ce projet RNRT sont la société CRIL Ingénierie et France-Télécom R&D.

L'utilisation à grande échelle d'outils d'aide à la vérification formelle passe par la mise en œuvre de procédures efficaces permettant de démontrer automatiquement des propriétés valides dans certaines théories courantes, comme la logique des propositions, l'arithmétique de Presburger, les systèmes de transitions finis. Les applications visées se situent dans le cadre de la certification des logiciels de télécommunications. Notre objectif dans ce projet est de combiner l'efficacité de la démonstration automatique avec la puissance, la souplesse et la fiabilité des systèmes de preuve généralistes. Nous avons étudié 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** peut utiliser une technique de preuve par réflection qui permet d'internaliser une procédure de décision afin d'automatiser certaines preuves de manière sûre et efficace. Dans un deuxième temps, nous avons étudié la génération directe du terme de preuve **Coq**. Ce travail est décrit dans la section 6.4.2 et disponible à l'adresse CALIFE.

## 7.2. 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 2002 concerne :

- l'analyse et la compréhension du problème chimique ;
- le choix de SMILES [57][58], une notation linéaire existante, comme notation externe pour GasEl ;
- la définition d'une notation interne des molécules polycycliques et des réactions chimiques ;
- la traduction de toutes les réactions élémentaires génériques primaires de EXGAS en **ELAN** ;
- l'utilisation des stratégies **ELAN** pour permettre aux chimistes d'activer ou désactiver les réactions élémentaires génériques primaires ;
- l'analyse de l'utilisation des stratégies **ELAN** pour enchaîner les réactions élémentaires génériques primaires et pour calculer des données cinétiques.

# 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.

PROTHEO participe au thème « Qualité et sûreté des logiciels et systèmes informatiques » du PRST Intelligence Logicielle. Claude Kirchner en assure l'animation, Olivier Bournez a été responsable de la conception de la plateforme disponible à l'adresse http://plateforme-qsl.loria.fr/plateforme_QSL/. Nous participons à l'opération ACT en collaboration avec le projet Calligramme et à l'opération VALDA en coopération avec l'équipe Cassis.

## 8.2. Actions nationales

Nous sommes impliqués dans deux projets RNTL labellisés en 2002 Le premier, AVEROES a débuté en octobre et prolonge le projet RNRT CALIFE qui s'arrête en janvier 2003. Le second, MANIFICO, est un projet pré-compétitif avec ILOG et le projet Contraintes de l'INRIA-Rocquencourt qui a pour sujet la compilation non-intrusive du filtrage dans le domaine de la résolution de contraintes.

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.

## 8.3. Réseaux et groupes de travail internationaux

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

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

## 8.4. 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 avons depuis 2001 une collaboration franco-brésilienne avec le département de mathématiques appliquées de l'Université Fédérale (UFRN) à Natal. Le projet correspondant, 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 œuvre 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 algébriques (CASL), que par un langage basé sur la réécriture comme **ELAN**. Un article est en cours de soumission sur la manipulation de spécifications algébriques grâce des représentations de termes et de graphes.

Un projet de coopération franco-chilienne avec l'Université Technique Federico Santa Maréa, Valparaéso, a été accepté en 2002. Ce projet est financé dans le cadre du programme de coopération CONICYT/INRIA, il porte sur l'utilisation de la réécriture et des stratégies pour la résolution de contraintes.

Paulin de Naurois effectue sa thèse en co-tutelle avec *City University* de Hong Kong. Il est dirigé à City University par Felipe Cucker, et par Olivier Bournez et Jean-Yves Marion (projet Calligramme) au LORIA. Son travail de thèse concerne la complexité dans le modèle de Blum Shub et Smale et ses relations avec la complexité implicite.

Clara Bertolissi et Alberto Ciaffaglione effectuent leur thèse en co-tutelle avec l'université d'Udine sous la co-direction de Furio Honsell à Udine et de Claude Kirchner, Luigi Liquori et Horatiu Cirstea à Nancy.

## 8.5. Accueils de chercheurs étrangers

- Notre coopération avec l'équipe de Paul Klint du CWI à Amsterdam s'est vue renforcée par l'accueil de Mark van den Brand sur un poste de spécialiste étranger à compter de novembre 2001. Dans le cadre l'équipe associée AirCube, Jurgen Vinju a fait un séjour de 3 mois dans PROTHEO de septembre à novembre 2002.
- Dans le cadre de notre coopération franco-brésilienne, Anamaria Martins Moreira, de l'UFRN (Natal, Brésil), effectue une année sabbatique dans PROTHEO. Nous avons eu d'autre part les visites de Benjamin Callejas Bedregal et Regivan Hugo Nunes Santiago, pendant deux semaines en juin 2002. De plus, nous accueillons un stagiaire brésilien, Anderson Santana, pour un séjour de quatre mois à compter de novembre 2002.

- Notre coopération franco-chilienne permet des visites régulières de Carlos Castro de l'Université Technique Federico Santa Maria de Valparaiso.

Par ailleurs, nous avons accueilli deux élèves ingénieurs de l'ENSI Tunis, Bacem Aouadi et Issam Chebbi, en stage de fin d'étude (quatre mois), ainsi qu'un stagiaire indien, Gaurav Kwatra, pendant deux mois et demi.

# 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 jusqu'au 31 Septembre 2002.
- 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 et de l'Université de Metz, Membre du comité de direction du Réseau Thématique Pluridisciplinaire SECC (Systèmes embarqués complexes et contraints).
  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 « 18th Conference on Automated Deduction » : CADE 18, « 2nd International Workshop on Reduction Strategies in Rewriting and Programming » : WRS'2002, « 4th International Workshop on Rewriting Logic and its Applications » : WRLA'2002, « 3rd International Workshop on Rule-Based Programming » : RULE'2002, « 4th International Conference on Principles and Practice of Declarative Programming » : PPDP'2002 (président du comité de programme) [10], « 5th SBC Workshop on Formal Methods » : WML'2002, Organisateur des journées QSL : « assistants à la preuve » (mai) et « Journée QSL » (novembre).
  Président du groupe de travail IFIP WG 1.6 sur la réécriture, Membre fondateur de IFCoLog (`www.ifcolog.org`), Membre du QPQ (QED Pro Quo) « advisory board », Co-lauréat avec Jean-Pierre Jouannaud du « Prix de la fondation culturelle Franco-Chinoise » délivré par l'académie des sciences.
- 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 groupes IFIP WG 1.3 (*Foundations of Systems Specifications*) et IFIP WG 1.6 (*Term Rewriting*).
  Membre du Comité de direction du RTP MicroRobotique.
  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, du LSV (présidente), du LIFC, du Leibniz.
  Co-présidente du comité de programme de AMAST'02 [9], membre du comités de programme FroCoS'02.

- Pierre-Etienne Moreau :
  membre des comités de programme de LDTA'2002 et WRLA'2002
- Christophe Ringeissen :
  correspondant pour les projets européens du LORIA,
  membre des comités de programme de CoSolv'02 (workshop de CP'02), FroCoS'02, AMAST'02, co-président du comité de programme de AMAST'02 [9],
  membre du comité d'organisation de UNIF'02 (workshop de RTA'02, dans le cadre de FLoC'02) [11],
  co-animateur du groupe « Collaboration de Solveurs » du GDR ALP
- Mark van den Brand :
  membre des comités de programmes de LDTA'2002 [12] (président du comité de programme), RULE'2002, and CSMR'2002,
  editeur d'un numero special de la revue *Journal of Logic and Algebraic Programming (JLAP)*.

## 9.2. Enseignement universitaire

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

- Olivier Bournez a donné un cours « Vérification algorithmique » et un cours « Analyse d'algorithmes. Complexité de Problèmes » dans le cadre du DEA d'Informatique 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 un cours : « 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 d'informatique de base et un cours de traduction à l'ESIAL.
- Christophe Ringeissen a donné un cours : « Contraintes et optimisation Combinatoire » avec Alexander Bockmayr, dans le cadre du DEA d'Informatique de Lorraine.
- Juergen Stuber a assuré des TD et TP de programmation fonctionnelle en OCaml, en 1ère année de DEUG MIAS 1 à l'Université Henri Poincaré - Nancy 1.

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 :
  Exposé invité « Modèles de calcul à temps continu » pour le séminaire du magistère d'informatique de l'Ecole Normale Supérieure de Lyon en Mai 2002.
  Organisation d'une journée de formation sur « Les outils de la vérification » au LORIA (Juin 2002) et exposé « Vérification des systèmes temporisés et hybrides » dans ce cadre.

- Claude Kirchner :
  Séminaire au laboratoire PPS (Preuves, Programmes et Systèmes) : *Le calcul de réécriture*.
  Présentation invitée au Workshop « Automath2002, on Thirty Five Years of Automath » : *The rewriting calculus*, Herriot-Watt University, Edinburgh (Avril 2002)
  Présentation invitée à la session commume de la conférence AISC (Artificial Intelligence and Symbolic Computation) et du workshop Calculemus : *Deduction versus computation : the case of induction*.
  Présentation invitée à UNIF'2002 : *Abstract Canonical Inference Systems*.
  Séminaire au SRI-CSL : *Deduction versus computation : the case of induction*.
- Hélène Kirchner :
  Exposé invité au Workshop « Automath2002, on Thirty Five Years of Automath » Herriot-Watt University, Edinburgh (Avril 2002)
  Séminaire au SRI-CSL : *External rewriting in proof assistants based on constructive type theory*.
- Pierre-Etienne Moreau :
  Co-organisateur d'une journée QSL sur la compilation des langages fonctionnels et objets (Mars 2002).
- Mark van den Brand :
  Exposé invité pendant la semaine internationale de l'Ecole de Mines de Nancy.
  Exposé invité dans le cadre de la journée QSL sur la compilation des langages fonctionnels et objets (Mars 2002).

Mis à part les congrès et colloques où furent présentés nos travaux, nous avons participé aux événements suivants : AMAST'2002, RTA'02, WADT'02, EFSD'02, CiAD 2002, FROCOS'2002 RULE'2002, PPDP'2002, WRLA'2002, Workshop « Aircube », Gulpen, Workshop « Scientific meeting in honour of Zohar Manna », LSV Cachan, « INRIA / NSC Taiwan, MultiMedia and XML Workshop », Sophia-Antipolis, MiniWorkshop on DLMF, Pittsburgh, 8-9 octobre, Séminaire d'évaluation du programme 2A, Paris, Séminaire Sécurité, Microsoft Research, Cambridge, Workshop privé a Carnegie Mellon sur Latex2MathML, Digital Library of Mathematical Functions.

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 modélisation et la simulation de systèmes dynamiques a structures dynamiques ».

### *9.3.2. Séjours de chercheurs*

- Claude Kirchner :
  Visite invitée du 29 juillet au 7 Août au SRI-CSL (Menlo Park, Californie).
- Hélène Kirchner :
  Visite invitée du 29 juillet au 7 Août au SRI-CSL (Menlo Park, Californie).
- Juergen Stuber :
  Séjour à l'Université de Sarrebrück du 19 au 21 novembre, pour une rencontre avec Harald Ganzinger.
- Mark van den Brand :
  Visite invitée du 1 Décembre au 5 Décembre au BRICS (Aarhus, Denmark).

## 9.4. Jurys de thèses et jurys divers

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

- Claude Kirchner :
  Eric Deplagne, « Système de preuve modulo récurrence », Thèse de l'université Henri Poincaré, Nancy, novembre, directeur.
  Eric Monfroy, « Résolution coopérative de Contraintes », HDR Université de Nantes, novembre, président.
  Jérémie Blanc, « Les Systèmes de Réécriture de Termes Temporisés », Thèse de l'université Joseph Fourier, Grenoble, décembre, rapporteur.
  Cuihtlauac Alvarado, « Réflexion pour la réécriture dans le calcul des constructions inductives », Thèse de l'université de Paris XI Orsay, décembre.
- Hélène Kirchner :
  Quang Huy Nguyen, Thèse de l'université Henri Poincaré - Nancy I, « Calcul de réécriture et automatisation du raisonnement dans les assistants de preuve », octobre.
  Frank Ledoux, Thèse de l'université d'Evry, Val d'Essonne, « Etude et spécifications formelles de l'arrondi d'objets géométriques », décembre.
  Jurys de concours ITA INRIA.
- Isabelle Gnaedig et Pierre-Etienne Moreau ont participé aux jurys du concours d'entrée à l'ESIAL.

# 10. Bibliographie

## Bibliographie de référence

[1] 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.

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

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

[4] 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.

[5] 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, à paraître.

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

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

[8] 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.

## Livres et monographies

[9] *Algebraic Methodology And Software Technology.* éditeurs C. R. HÉLÈNE KIRCHNER., série Lecture Notes in Computer Science, volume 2422, Springer-Verlag, septembre, 2002.

[10] *Proceedings of the 4th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'02).* octobre, 2002.

[11] *Proceedings of the 16th International Workshop on Unification (UNIF 2002).* éditeurs C. RINGEISSEN, C. TINELLI, R. TREINEN, R. M. VERMA., Department of Computer Science, University of Iowa, July, 2002, Technical Report 02-05. UNIF 2002 was affiliated with RTA 2002, part of FLoC'02, Copenhagen, Denmark.

[12] M. VAN DEN BRAND, R. LAMMEL. éditeurs M. VAN DEN BRAND, R. LAMMEL., *Second Workshop on Language Descriptions, Tools and Applications (LDTA 2002).* série Electronic Notes in Theoretical Computer Science, volume 65, Elsevier Science Publishers, 2002.

## Thèses et habilitations à diriger des recherche

[13] E. DEPLAGNE. *Système de preuve modulo récurrence.* Thèse de doctorat, Université Henri Poincaré - Nancy 1, novembre, 2002, http://www.loria.fr/publications/2002/A02-T-513/A02-T-513.ps.

[14] Q. H. NGUYEN. *Calcul de réécriture et automatisation du raisonnement dans les assistants de preuve.* Thèse d'université, UHP - Nancy 1, octobre, 2002.

## Articles et chapitres de livre

[15] E. ASTESIANO, M. BIDOIT, H. KIRCHNER, B. KRIEG-BR ?CKNER, P. D. MOSSES, D. SANNELLA, A. TARLECKI. *CASL : The Common Algebraic Specification Language.* in « Theoretical Computer Science », numéro 2, volume 286, septembre, 2002, pages 153-196.

[16] P. BOROVANSKY, C. KIRCHNER, H. KIRCHNER, P.-E. MOREAU. *ELAN from a rewriting logic point of view.* in « Theoretical Computer Science », numéro 285, juillet, 2002, pages 155-185.

[17] M. V. D. BRAND, J. HEERING, P. KLINT, P. OLIVIER. *Compiling language definitions : The ASF+SDF compiler.* in « ACM Transactions on Programming Languages and Systems », numéro 4, volume 24, 2002, pages 334-368.

[18] Q. H. NGUYEN, C. KIRCHNER, H. KIRCHNER. *External rewriting for skeptical proof assistants.* in « Journal of Automated Reasoning », janvier, 2002, à paraître.

## Communications à des congrès, colloques, etc.

[19] G. BARTHE, H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *Pure Patterns Type Systems.* in «

Principles of Programming Languages - POPL2003, New Orleans, USA », ACM, janvier, 2002, http://www.loria.fr/publications/2003/A03-R-002/A03-R-002.ps, à paraître.

[20] O. BOURNEZ. *A Generalization of Equational Proof Theory ?.* in « Second Joint International Workshop on Process Algebras and Performance Modeling / Probabilistic Methods In Verification - PAPM-PROBMIV'02, Copenhagen, Denmark », série Lecture Notes in Computer Science, volume 2399, Springer, éditeurs R. S. H. HERMANNS., pages 207-208, juillet, 2002.

[21] O. BOURNEZ, P. DE NAUROIS, J.-Y. MARION. *Safe Recursion and Calculus over an Arbitrary Structure.* in « Implicit Computational Complexity - ICC'2002, Copenhagen, Denmark », juillet, 2002.

[22] O. BOURNEZ, C. KIRCHNER. *Probabilistic rewrite strategies. Applications to ELAN.* in « 13th International Conference on Rewriting Techniques and Applications - RTA'2002, Copenhagen, Denmark », série Lecture Notes in Computer Science, volume 2378, Springer, éditeurs S. TISON., pages 252-266, juillet, 2002.

[23] H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *Rewriting Calculus with(out) Types.* in « Workshop on Rewriting Logic and its Applications - WRLA'2002, Pisa, Italy », ENTCS, septembre, 2002.

[24] H. CIRSTEA, C. KIRCHNER, L. LIQUORI, B. WACK. *The rho cube : some results, some problems.* in « First International Workshop on Higher-Order Rewriting - HOR'02, Copenhague, Danemark », D. Kesner, T. Nipkow & F. van Raamsdonk, juillet, 2002, http://www.loria.fr/publications/2002/A02-R-470/A02-R-470.ps, Held in conjunction with FLOC'02.

[25] E. DEPLAGNE, C. KIRCHNER. *Deduction versus Computation : the Case of Induction.* in « Sixth International Conference on Artificial Intelligence and Symbolic Computation - AISC'2002, Marseille, France », série Lecture notes in Artificial Intelligence, volume 2385, Springer, éditeurs O. C. L. H. V. S. J. CALMET., pages 4-6, juillet, 2002.

[26] N. DERSHOWITZ, C. KIRCHNER. *Abstract Canonical Inference Systems.* in « 16th International Workshop on Unification - UNIF'2002, Copenhagen, Denmark », juillet, 2002, http://www.loria.fr/publications/2002/A02-R-501/A02-R-501.ps.

[27] G. DOWEK, T. HARDIN, C. KIRCHNER. *Binding Logic : proofs and models.* in « 9th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning - LPAR'2002, Tbilisi, Georgia », série Lecture notes in Artificial Intelligence, volume 2514, Springer, éditeurs A. V. M. BAAZ., pages 130-144, octobre, 2002.

[28] H. DUBOIS, H. KIRCHNER. *An algebraic semantics for objects in a rule-based language.* in « 16th International Workshop on Algebraic Development Techniques - WADT'2002, Frauenchiemsee, Germany », septembre, 2002.

[29] H. DUBOIS, H. KIRCHNER. *Object Programming in a Rule-Based Language with Strategies.* in « Workshop on Multiparadigm Programming with OO Languages - ECOOP'02, Malaga, Spain », juin, 2002.

[30] D. DÉHARBE, A. MARTINS MOREIRA, C. RINGEISSEN. *Improving Symbolic Model Checking by Rewriting Temporal Logic Formulae.* in « Conference on Rewriting Techniques and Applications - RTA'2002, Copenhagen, Denmark », série Lecture Notes in Computer Science, volume 2378, Thomas Arts, Springer-Verlag,

éditeurs S. TISON., pages 207-221, juillet, 2002.

[31] G. FAURE, C. KIRCHNER. *Exceptions in the rewriting calculus.* in « 13th International Conference on Rewriting Techniques and Applications - RTA'2002, Copenhagen, Denmark », série Lecture Notes in Computer Science, volume 2378, Springer, éditeurs S. TISON., pages 66-82, juillet, 2002.

[32] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *CARIBOO : An Induction Based Proof Tool for Termination with Strategies.* in « Fourth International Conference on Principles and Practice of Declarative Programming - PPDP'02, Pittsburgh, USA », ACM Press, éditeurs F. PFENNIG., octobre, 2002.

[33] O. FISSORE, I. GNAEDIG, H. KIRCHNER. *Outermost ground termination.* in « 4th International Workshop on Rewriting Logic and its Applications - WRLA' 02 , Pisa, Italy », Electronic Notes in Computer Science, Pisa, Italy, septembre, 2002, To appear.

[34] P. K. M.G.J. VAN DEN BRAND, J. VINJU. *Term Rewriting with Type-safe Traversal Functions.* in « 2nd International Workshop on Reduction Strategies in Rewriting and Programming (WRS 2002) », série Electronic Notes in Theoretical Computer Science, volume 70, Elsevier Science Publishers, éditeurs B. GRAMLICH, S. LUCAS., 2002.

[35] J. V. M.G.J. VAN DEN BRAND, E. VISSER. *Disambiguation Filters for Scannerless Generalized LR Parsers.* in « Compiler Construction (CC'02) », série Lecture Notes in Computer Science, volume 2304, Springer-Verlag, éditeurs R. HORSPOOL., pages 143-158, 2002.

[36] Q. H. NGUYEN. *A constructive decision procedure for equalities modulo AC.* in « 16th International Workshop on Unification - UNIF 16, Copenhagen, Denmark », juillet, 2002.

[37] M. SCHMIDT-SCHAUSS, J. STUBER. *On the complexity of linear and stratified context matching problems.* in « 2nd International Workshop on Complexity in Automated Deduction - CiAD'02, Copenhagen, Denmark », juillet, 2002.

[38] M. G. J. VAN DEN BRAND, P.-E. MOREAU, C. RINGEISSEN. *The ELAN Environment : an Rewriting Logic Environment based on ASF+SDF Technology.* in « Workshop on Language Descriptions, Tools and Applications - LDTA'02, Grenoble, France », série Electronic Notes in Theoretical Computer Science, numéro 3, volume 65, avril, 2002.

## Rapports de recherche et publications internes

[39] C. BERTOLISSI. *Traduction des Combinatory Reduction Systems en Rho-Calcul.* Stage de DEA, juillet, 2002, http://www.loria.fr/publications/2002/A02-R-497/A02-R-497.ps.

[40] O. BOURNEZ. *A Generalization of Equational Proof Theory ?.* Rapport de recherche, novembre, 2002.

[41] O. BOURNEZ, P. DE NAUROIS, J.-Y. MARION. *Safe Recursion over an Arbitrary Structure : Deterministic Polynomial Time.* Rapport de recherche, octobre, 2002.

[42] C. CASTRO, C. KIRCHNER. *Towards a Methodology for Rule-Based Programming.* Rapport de recherche, décembre, 2002, http://www.loria.fr/publications/2002/A02-R-529/A02-R-529.ps.

[43] I. CHEBBI. *Une boîte à outils pour la programmation par règles.* rapport technique, LORIA, Nancy, France, 2002, Rapport de projet de fin d'études d'Ingénieur de l'Ecole Nationale des Sciences de l'Informatique de Tunis.

[44] M. HOYRUP. *Réécriture en présence de choix probabilistes.* Stage Magistère 1ère année, septembre, 2002.

[45] A. MARTINS MOREIRA, C. RINGEISSEN, D. DÉHARBE, G. LIMA. *Manipulating Algebraic Specifications with Term-based and Graph-based Representations.* Rapport de recherche, décembre, 2002, Submitted to "Journal of Logic and Algebraic Programming".

[46] P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. *A Pattern Matching Compiler for Multiple Target Languages.* Rapport de recherche, octobre, 2002.

[47] P.-E. MOREAU, O. ZENDRA. *GC2 : A Generational Conservative Garbage Collector for the ATerm Library.* Rapport de recherche, septembre, 2002, http://www.inria.fr/rrrt/rr-4548.html.

## Bibliographie générale

[48] S. BELLANTONI, S. COOK. *A new recursion-theoretic characterization of the poly-time functions.* in « Computational Complexity », volume 2, 1992, pages 97-110.

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

[50] 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.

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

[52] 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, Kanazawa, Japan, September, 2000.

[53] H. CIRSTEA, C. KIRCHNER. *The rewriting calculus - Part I and II.* in « Logic Journal of the Interest Group in Pure and Applied Logics », numéro 3, volume 9, 2001, pages 427-498.

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

[55] H. CIRSTEA, C. KIRCHNER, L. LIQUORI. *The Rho Cube.* in « 4th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'2001) », série Lecture Notes in Computer Science,

volume 2030, Springer, éditeurs F. HONSELL, M. MICULAN., pages 168-183, Genova (Italy), April, 2001, Also available as Technical Report A01-R-202, LORIA, Nancy (France).

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

[57] DAYLIGHT. *SMILES - A Simplified Chemical Language.* Daylight Chemical Information Systems Inc., http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html.

[58] DAYLIGHT. *SMILES Tutorial.* Daylight Chemical Information Systems Inc., http://www.daylight.com/dayhtml/smiles/smiles-intro.html.

[59] 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, Also available as Technical Report A01-T-200, LORIA, Nancy (France).

[60] 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, Cambridge (MA, USA), 1991, chapitre 8, pages 257-321.

[61] J. JOUANNAUD, M. OKADA. *Abstract data type systems.* in « Theoretical Computer Science », numéro 2, volume 173, 1997, pages 349-391.

[62] J. KLOP, V. VAN OOSTROM, F. VAN RAAMSDONK. *Combinatory reduction systems : introduction and survey.* in « Theoretical Computer Science », volume 121, 1993, pages 279-308.

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

[64] D. LEIVANT, J.-Y. MARION. *Ramified recurrence and computational complexity II : substitution and poly-space.* in « Computer Science Logic, 8th Workshop, CSL '94 », série Lecture Notes in Computer Science, volume 933, Springer, éditeurs L. PACHOLSKI, J. TIURYN., pages 486-500, Kazimierz,Poland, 1995.

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

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

[67] P.-E. MOREAU, C. RINGEISSEN, M. VITTEK. *A Pattern-Matching Compiler.* in « First Workshop on Language Descriptions, Tools and Applications - LDTA'01, Genova, Italy », série Electronic Notes in Theoretical Computer Science, volume 44-2, Elsevier, éditeurs M. VAN DEN BRAND, D. PARIGOT., avril, 2001.

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

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