Rapport INRIA 1994 — Programme 2

Preuve, calcul symbolique et logique

PROJET EURÉCA

3 mai 1995

PROJET EURÉCA

Preuve, calcul symbolique et logique

Localisation : *Nancy*[1]

Mots-clés : analyse d'algorithme (1, 11), automate (11, 14), calcul formel (11), complexité (9, 11), COQ (7), expression régulière (1, 11), grammaire (9), lambda-calcul (1, 4, 5), langage d'arbres (9), langage formel (9), logique (1, 8), marche aléatoire (11), modélisation statistique (11), parallélisme (5, 14), preuve de circuit (14), preuve de programme (1, 7, 14), programmation fonctionnelle (4), réécriture (1, 4, 8, 9, 11), séquence génétique (11), spécification formelle (7).

1 Composition de l'équipe

Responsable scientifique

Pierre Lescanne, directeur de recherche, CNRS

Secrétariat

Christiane Guyot

Personnel INRIA

Grégory Kucherov, chargé de recherche
Paul Zimmermann, chargé de recherche

Personnel CNRS

Roberto Amadio, chargé de recherche, jusqu'au 31/10/94
Nicolas Hermann, chargé de recherche

[1] Projet commun Inria-Crin.

Jean-Luc Rémy, chargé de recherche

Personnel Université

François Bronsard, ATER à l'université Nancy 2
Adam Cichon, professeur
à l'Université Henri Poincaré-Nancy 1
Jocelyne Rouyer, maître de conférences
à l'Université Henri Poincaré-Nancy 1
Joseph Rouyer, professeur agrégé
à l'Université Henri Poincaré-Nancy 1

Chercheurs invités

Phokion Kolaitis, professeur invité
Leszek Pacholski, professeur invité

Chercheurs post-doctorants

Valentin Antimirov, INRIA
Gernot Salzer, INRIA-CHM
Andreas Weiermann, INRIA-CHM

Chercheurs doctorants

Otmane Aït-Mohamed, boursier du gouvernement algérien
Zine-El-Abidine Benaïssa, boursier INRIA
Daniel Briaud, allocataire MESR
Boutheina Chetali, boursière du gouvernement français
Barbara Heyd, allocataire MESR, depuis le 1/10/94
Stefan Krischer, boursier du Conseil régional de Lorraine, jusqu'au 30/9/94
François Leclerc, allocataire MRE, jusqu'au 30/11/94
Sohame Selhab, boursière INRIA
Elias Tahhan-Bittar, boursier du gouvernement vénézuelien, jusqu'au 31/10/94
Hélène Touzet, allocataire MESR

2 Présentation du projet

Historiquement, EURÉCA s'est fait connaître par ses travaux sur la réécriture. Cette recherche reste l'un des points forts du projet, tout en s'enrichissant de nouvelles applications et l'arrivée de nouveaux chercheurs. 1994 a vu la consolidation d'une nouvelle structure où se

dégagent deux axes : l'algorithmique et la logique avec un domaine d'application sur la preuve de systèmes informatiques.

La logique mathématique nous sert à comprendre et certifier formellement le comportement des programmes sous plusieurs aspects. En *λ-calcul et parallélisme* nous étudions la sémantique et les problèmes de calcul qui sont à la base de la programmation fonctionnelle et parallèle. Le *calcul des constructions inductives* et son implantation dans le logiciel COQ est une logique fondée sur le λ-calcul polymorphe avec types dépendants. Nous l'avons utilisé pour programmer-spécifier-prouver des algorithmes classiques et représentatifs de la logique de la réécriture. Le *λ-calcul avec substitutions explicites* est une approche du λ-calcul qui intègre naturellement le concept de substitution. Nous l'avons décrit simplement et en avons dérivé une machine abstraite. En *logique mathématique et réécriture,* nous abordons la terminaison qui est un problème important en théorie de la démonstration, avec comme application privilégiée, la preuve de terminaison des systèmes de réécriture.

Dans la composante algorithmique et calcul symbolique se regroupent des recherches qui apparaissaient en 1993 sous la rubrique *outils mathématiques*, mais à leurs côtés, démarrent d'autres recherches sur le *génome* et sur les évaluations de la complexité en taille des résultats (#P-complexité). En particulier, l'intérêt se porte sur la fabrication d'outils d'analyse et d'aide à la décision. Pour ce faire, l'usage du calcul formel a été systématisé et une expertise plus particulière a été acquise dans les domaines de la théorie des nombres et de la génération aléatoire de structures combinatoires.

À ces deux axes de la recherche fondamentale du projet s'ajoute un domaine très important d'études des applications de la preuve à des problèmes concrets.

En preuve de programmes et de circuits, nous avons mis au point le logiciel FANCY qui propose une structure d'accueil pour plusieurs algorithmes de vérification de circuits digitaux fondés sur la comparaison de machines d'états finis. Nous avons également étudié comment des protocoles de communication décrits dans le langage UNITY pouvaient être prouvés corrects à l'aide du logiciel de démonstration automatique LARCH PROVER.

1994 est aussi une année de mouvement à l'intérieur du projet EURÉCA: R. Amadio, chargé de recherche CNRS, quitte le projet pour aller rejoindre le laboratoire I3S de Sophia-Antipolis, tandis que P. Zimmermann et N. Hermann sont en année sabbatique, le premier à l'Université de Paderborn (Allemagne) et le second à l'Université de Californie à Santa Cruz (États-Unis).

3 Actions de recherche

3.1 Lambda-calcul, logique mathématique et réécriture

3.1.1 Lambda-calculs avec substitutions explicites et machines abstraites

Participants : Zine-El-Abidine Benaïssa, Daniel Briaud, Pierre Lescanne, Jocelyne Rouyer, Joseph Rouyer

La normalisation forte est fondamentale en déduction automatique : elle joue entre autres un rôle important en calcul des constructions. La recherche en λ-calcul avec substitutions explicites est une continuation privilégiée de nos travaux antérieurs sur la réécriture du premier ordre. Nous comptons exploiter ces connaissances dans une mise en œuvre efficace par partage et parallélisme de la normalisation forte, à travers la définition de machines abstraites adaptées. Nos travaux sur la η-réduction devraient nous permettre de mieux comprendre l'unification d'ordre supérieur, en complément des travaux des projets PROTHÉO, PARA et COQ.

Le mécanisme de réduction du λ-calcul repose sur la règle β qui utilise la substitution. Celle-ci est habituellement décrite dans un formalisme extérieur à celui du λ-calcul lui-même, dans l'épithéorie comme l'appelle Curry. Le calcul avec substitutions explicites a pour objectif d'intégrer l'évaluation de la substitution au même niveau que la règle β et à partir de là, il y a essentiellement deux écoles. Celle du système $\lambda\sigma$ (étudié dans le projet PARA et au LIENS) s'intéresse essentiellement à calculer sur les substitutions pour les simplifier autant que possible et capturer l'essentiel de la sémantique. Pour ce faire, elle introduit la *composition* des substitutions et privilégie la confluence sur les termes ouverts. Dans EURÉCA, nous privilégions plutôt la compacité de la présentation et la minimalité des concepts. Nous avons découvert récemment que l'ini-

tiateur de cette démarche était, en 1978, Nicolas de Bruijn avec son système $C\lambda\xi\phi$. Le système $\lambda\upsilon$ qui en résulte préserve la forte normalisation (terminaison) des termes du λ-calcul classique. À partir de là quatre directions ont été poursuivies, certaines en collaboration avec Philippe De Groote (projet PROGRAIS):

Le calcul $\lambda\upsilon$ a été présenté originellement à la conférence POPL [27]. Le résultat le plus important concernant $\lambda\upsilon$ est la preuve de préservation de la forte normalisation ou terminaison. Ce résultat est d'autant plus intéressant que Paul-André Melliès du projet PARA a montré que $\lambda\sigma$ ne préserve pas la forte normalisation, même celle des termes simplement typés. Pierre Lescanne et Jocelyne Rouyer [36] ont aussi montré la correction de ce calcul et d'une machine associée à ce calcul.

Le calcul $\lambda\upsilon_l$ étend $\lambda\upsilon$ dans le but de dériver des machines abstraites efficaces. Ce calcul regroupe les substitutions élémentaires de $\lambda\upsilon$ dans des structures de liste pour constituer des environnements. Dans [39], nous avons décrit ce calcul et montré certaines de ses propriétés, telles que la convergence du calcul de substitutions $\lambda\upsilon_l$, la relation entre les deux calculs, la correction et la confluence sur les termes clos de $\lambda\upsilon_l$.

Substitutions explicites et règle η**:** dans [18], nous décrivons une manière d'expliciter la η-réduction. Nous utilisons à cet effet une règle non-conditionnelle et générique nommée *Eta*. Les précédents travaux sur ce sujet dûs à T. Hardin (projet PARA) et A. Ríos définissaient *Eta* comme une extension de la η-contraction. En conséquence cette règle *Eta* était conditionnelle et ne respectait pas la philosophie des substitutions explicites. Nous avons prouvé quelques propriétés du calcul obtenu en alliant notre définition de *Eta* à $\lambda\upsilon$, à savoir sa correction, sa confluence sur les termes clos et la forte normalisation des termes simplement typés. Par ailleurs, cette *Eta* explicite conduit à une nouvelle définition de η, plus générale dans le sens où elle autorise plus de contractions confluentes que η.

Le calcul $\lambda\chi$ est un calcul qui utilise les niveaux de De Bruijn au lieu des indices. En effet, dans son article originel, de Bruijn avait proposé deux moyens de nommer canoniquement les variables : les *niveaux* et les *indices*. Les niveaux sont plus intuitifs dans la mesure où chaque variable garde le numéro qui lui est associé à travers tout le terme.

Tous les calculs de substitutions explicites utilisent les indices, il est apparu intéressant d'explorer un calcul fondé sur les niveaux [26]. Ce calcul, appelé $\lambda\chi$, a toutes les propriétés intéressantes des autres calculs à substitutions explicites.

3.1.2 Lambda-calcul et Parallélisme

Participants : Otmane Aït-Mohamed, Roberto Amadio

Les travaux décrits ici étudient la *sémantique* et des *problèmes de vérification* de calculs, qui sont à la base de la programmation fonctionnelle et parallèle.

Dans [32], nous présentons un langage simplement typé,CORE FACILE, inspiré par des travaux précédents sur le langage de programmation FACILE (développé à l'ECRC). Les trois ingrédients fondamentaux en sont:

- Un λ-calcul avec appel par valeur étendu avec la possibilité d'évaluer des expressions en parallèle.
- Une notion de canal et d'opérateurs pour lire-écrire les canaux d'une façon synchrone.
- La possibilité d'engendrer dynamiquement de nouveaux canaux pendant l'exécution.

Le but est de faire le pont entre des langages de programmation comme FACILE et CML et des calculs théoriques comme CHOCS et le π-calcul. Nous apportons deux contributions principales. D'une part, nous introduisons une nouvelle sémantique basée sur la notion de bisimulation à barbes qui fournit un traitement satisfaisant de l'opérateur de restriction. En particulier, nous définissons une traduction adéquate de CORE FACILE dans le π-calcul. D'autre part, nous étudions, à un niveau abstrait, certains aspects de la compilation de programmes FACILE. Nous présentons une version "asynchrone" du langage, disons F_a, où un processus termine après avoir accompli une action d'output. Nous analysons une traduction adéquate de type CPS (*Continuation Passing Style*) de FACILE vers F_a. De plus, nous décrivons une machine abstraite qui exécute le code de programmes F_a. Cette machine est essentiellement un multi-ensemble de machines à environnement pour l'évaluation par valeur, enrichies avec un mécanisme pour la génération dynamique de noms de canaux.

Dans [16], nous présentons une extension simple du π-calcul avec actions et canaux localisés et avec noms de locations comme données de première classe, qui modèlise la notion de localité et échec présente dans le langage FACILE mentionné précédemment. L'interaction entre les localités et l'échec distingue notre approche d'approches précédentes où la notion de localité est considérée isolément. La combinaison de ces deux traits mène, au moins du point de vue de la programmation de systèmes distribués, à une sémantique plus naturelle. Ensuite, nous discutons la traduction de ce calcul dans le π-calcul et nous montrons son adéquation par rapport à une bisimulation à barbes. Dans la traduction, chaque localité est représentée par un processus spécial qui interagit à l'aide d'un protocole simple, avec chaque processus du programme original qui souhaite accéder à des ressources dépendantes de cette localité. Nous appliquons aussi notre traduction à la vérification d'un programme simple, tolérant aux fautes.
Dans [31], nous considérons le problème de la spécification et de la vérification de calculs de processus comme CHOCS, CML et FACILE dans lesquels processus ou fonctions sont des valeurs transmissibles. Notre travail se situe dans le contexte d'un traitement statique de l'opérateur de restriction et d'une sémantique basée sur la bisimulation. Comme cas simple et paradigmatique, nous nous concentrons sur (PLAIN) CHOCS. Nous montrons que la bisimulation de CHOCS peut être caractérisée par une extension de la logique de Hennessy-Milner incluant une implication constructive. Ce résultat est une extension non-triviale du résultat classique pour les systèmes de transition étiquetés. Dans la deuxième partie de ce papier, nous considérons le problème du développement d'un système de preuve pour la vérification des spécifications de processus. En étendant des travaux précédents sur CCS, nous présentons un système infinitaire cohérent et complet pour un fragment du calcul qui ne comprend pas la restriction.
Un autre axe de recherche concerne la définition d'outils de vérification pour le π-calcul. En particulier, comme partie de son travail de doctorat Ait-Mohamed a continué à développer la théorie du π-calcul dans le système de preuve interactive HOL. L'approche suivie est purement *définitionnelle* : la théorie du π-calcul est developpée en termes des notions primitives de la logique. Ainsi, on obtient une extension cohérente de la théorie de HOL et un outil *correct* pour développer des preuves d'équivalence de π-processus. Les premiers exemples de vérification développés dans cet outil concernent la correction de la représentation de

certaines structures de données dans le π-calcul.
Enfin, un effort important a été dédié à la rédaction de rapports de synthèse [1, 30, 29].

3.1.3 Calcul des constructions

Participants : Joseph Rouyer, François Leclerc, Pierre Lescanne

Dans sa thèse [4] sur le développement d'algorithmes dans le Calcul des Constructions, J. Rouyer a étudié les problèmes de représentation des termes et des graphes pondérés et a synthétisé dans COQ un algorithme d'unification et les algorithmes de Prim et Kruskal sur les recouvrements connexes de graphes.

Pour sa part, F. Leclerc propose dans [25] une formalisation complète d'une preuve de terminaison de système de réécriture avec l'ordre récursif sur les chemins dit MPO. Il a construit une fonction de terminaison qui borne le nombre de réductions de tout système de réécriture orientable avec cet ordre. La preuve est constructive et a été vérifiée dans COQ. Il s'agit d'un exemple non trivial de preuve complètement mécanisée (10 000 lignes) qui constitue une bibliothèque de preuves réutilisables. Il est possible d'appliquer cette technique à COQ pour justifier la terminaison de fonctions ou de structures de données définies par des équations.

3.1.4 Terminaison et preuves

Participants : Adam Cichon, Pierre Lescanne, Sohame Selhab, Elias Tahhan-Bittar, Hélène Touzet, Andreas Weiermann

En théorie de la démonstration, le théorème d'élimination des coupures de Gentzen, en calcul des séquents, est un résultat fondamental, en partie à la base des progrès que connaît la démonstration automatique de théorèmes. Il stipule que toute preuve en calcul des séquents classique (intuitionniste) peut être transformée en une preuve ne faisant pas intervenir la règle de la coupure. Les premières démonstrations du théorème de Gentzen montrent que pour une stratégie particulière cette réduction des preuves termine. Nous nous proposons de montrer que cette transformation des preuves termine pour toute stratégie (normalisation forte), en adoptant une approche syntaxique et des techniques de

réécriture. Un système de réécriture S réalisant l'élimination des coupures est construit pour lequel il reste à établir la terminaison. Plus précisément, les preuves sont représentées par des termes du 1^{er} ordre, ce qui permet l'extraction des règles de réécriture appropriées en faisant abstraction de la stratégie de réduction employée dans la procédure de normalisation de Gentzen. On prouve alors que le système S termine par des méthodes standards de la théorie de la réécriture. La même approche a permis d'étendre ces résultats aux calculs de séquents intuitionniste (LJ1) et linéaire (MALL1). Une description détaillée de ce travail figure dans [35].
De plus, des techniques classiques de réécriture permettent l'extraction de bornes sur la longueur des séquences de réécriture.

D'autres travaux ont été menés sur l'étude de structures combinatoires de la terminaison. En déduction automatique, deux exemples fondamentaux sont le théorème de Higman et celui de Kruskal. Une caractérisation complète de ces deux théorèmes devrait légitimement conduire à une meilleure connaissance, en terme de complexité notamment, de leur champ d'application.
Un cadre prometteur pour ce travail est celui des hiérarchies de fonctions indexées par des *ordinaux*. Il serait intéressant de définir un cadre uniforme pour l'analyse et la résolution de problèmes combinatoires. Le point de départ est la logique Π^1_2 de Girard, avec sa théorie de dénotations ordinales : celle-ci permet de capturer le principe de récursion, présent dans toute programmation non triviale.

Par ailleurs, P. Lescanne a proposé une preuve très simple de l'indécidabilité de la terminaison pour les systèmes de réécriture à une règle [11].

3.1.5 Réécriture de mots et de termes

Participants : François Bronsard, Nicolas Hermann, Grégory Kucherov, Pierre Lescanne

Schématisation d'ensembles infinis de termes: Dans son mémoire d'habilitation [2], N. Hermann donne une présentation complète de ses travaux sur la divergence des systèmes de réécriture, les schématisations des ensembles infinis de termes, les grammaires primales et leur unification.

Langages d'arbres et systèmes de réécriture de termes : G. Kucherov, D. Hofbauer (Université Technique de Berlin) et M. Huber (projet PROTHÉO) ont présenté leur travail commun sur la théorie des langages d'arbres à la conférence internationale CAAP à Édimbourg (Écosse) [23]. La motivation originelle de ce travail est d'étudier les conditions sous lesquelles le langage des termes clos réductibles par un système de réécriture est un langage non-contextuel (algébrique) d'arbres. Ce problème dans sa généralité s'est révélé très complexe puisqu'il nécessite une étude approfondie de plusieurs questions de la théorie des langages non-contextuels d'arbres. (Notons que contrairement aux langages réguliers, les langages non-contextuels d'arbres sont relativement peu étudiés et même les réponses aux questions de base ne sont pas universellement connues.) L'article [23] peut être considéré comme une première étape dans la résolution du problème ci-dessus. En particulier, l'article a mis en évidence une nouvelle sous-classe intéressante des langages non-contextuels appelés *co-réguliers* (*top-context-free*).
Ces travaux ont été poursuivis lors de la visite de G. Kucherov à l'Université Technique de Berlin en septembre 1994 pendant laquelle de nouvelles voies ont été explorées.

Il faut aussi noter dans ce domaine l'activité liée à la distribution de l'atelier de réécriture ORME, qui contient les idées et les algorithmes développés dans le projet [37].

Déduction clausale réductrice: ces recherches sur l'induction s'articulent selon trois axes:
- la déduction inductive [19]: la différence essentielle entre l'induction implicite et explicite se ramène à l'usage d'ordres sur les termes au lieu d'ordres sémantiques. Ceci a amené à définir une procédure d'induction explicite utilisant les ordres sur les termes. Cette procédure généralise strictement les procédures d'induction implicite tout en préservant les principaux avantages de ces procédures. Un prototype de cette méthode a été implémenté et a donné des résultats pratiques intéressants.
- la déduction clausale réductive: cette recherche s'inscrit dans le prolongement des travaux menés depuis deux ans en collaboration avec U. S. Reddy (Université d'Urbana-Champaign, Illinois). L'aspect principal porte sur l'implantation de la méthode. Il est prévu de distribuer le prototype du système de déduction d'ici peu.
- la complétude de la déduction clausale: nous avons proposé une nouvelle technique de représentation de preuves inspirée des réseaux de

preuve de Girard [2]. Cela a permis d'adapter à la déduction clausale la preuve de complétude de Bachmair, Dershowitz et Hsiang [3] pour l'algorithme de complétion équationnelle. En utilisant cette méthode, nous avons pu obtenir des preuves simples de la complétude de la résolution ordonnée, de l'algorithme de complétion clausale et de la paramodulation en l'absence de paramodulation dans des variables. Le manuscrit de ces résultats est en cours de préparation.

3.2 Algorithmique

Participants : Valentin Antimirov, Nicolas Hermann, Grégory Kucherov, Jean-Luc Rémy, Paul Zimmermann

3.2.1 Complexité d'algorithmes en calcul symbolique

Les algorithmes sur les expressions régulières : V. Antimirov étudie l'algèbre des expressions régulières, qui jouent un rôle fondamental en informatique en ayant de nombreuses applications pratiques. Le but initial de cette recherche est l'application de méthodes de logique équationnelle et de réécriture à des structures algébriques concrètes. Les problèmes de preuve d'égalité ($a = b$) et d'inclusion ($a \leq b$) des expressions régulières ont une importance particulière pour les applications. Ils sont décidables, mais PSPACE-complets, donc de complexité élevée. C'est pourquoi il est important d'étudier les propriétés logiques et algébriques des expressions régulières qui pourraient permettre d'obtenir des algorithmes plus efficaces. Les résultats antérieurs de la recherche de V. Antimirov sur ce sujet ont été obtenus en collaboration avec Peter Mosses et publiés dans [6, 7]. A partir de janvier 1994, V. Antimirov continue ses recherches à l'INRIA-Lorraine. Il a réalisé (dans le langage algébrique OBJ3) un nouvel algorithme de preuve d'équivalence d'expressions régulières basé sur un calcul algébrique décrit dans [6, 7]. L'algorithme utilise la réécriture pour la simplification d'expressions, ce qui conduit à des dérivations plus courtes. Une nouvelle notion de *dérivée partielle* d'une expression régulière a été définie et étudiée [17]. C'est une généralisation de la notion bien connue de *dérivées* d'expressions régulières introduite par Janusz Brzozowski[4]. Les dérivées partielles

[2] *Linear Logic* J.Y. Girard TCS volume 50 (1987)

[3] *Orderings for equational proof* L. Bachmair,N. Dershowitz and J. Hsiang -LICS86

[4] *Derivatives of regular expressions* J. A. Brzozowski - J. ACM, 11:481–494, 1964.

ont beaucoup de propriétés intéressantes. Par exemple, elles fournissent un lien naturel entre les expressions régulières et les automates non-déterministes à états finis (ANEF). Un algorithme exploitant cette relation pour transformer des expressions régulières en des ANEF a été développé. Dans [17], il est démontré que dans beaucoup de cas cet algorithme donne des ANEF plus petits que ceux obtenus avec les autres algorithmes connus (par exemple, l'algorithme de Gérard Berry et Ravi Sethi[5]). V. Antimirov a également développé une nouvelle méthode pour tester l'inclusion d'expressions régulières [38]. La méthode est basée sur les dérivées partielles et la réécriture de termes. Elle est présentée comme un système de réécriture normalisant qui transforme une inégalité régulière $a \leq b$ en la forme normale *false* si et seulement si $a \leq b$ n'est pas valide. Dans [38], il établit que dans certains cas cette méthode permet de tester l'inclusion en temps polynômial alors que les autres algorithmes connus le font en temps exponentiel.

Comptage dans l'unification et le filtrage équationnels: dans la hiérarchie des classes de complexité, les plus connues sont P (les algorithmes déterministes en temps polynomial) et NP (les algorithmes non déterministes en temps polynomial). La classe NP possède de nombreux problèmes complets, c'est-à-dire "les plus difficiles" de la classe.
Dans les problèmes de comptage, contrairement aux problèmes de décision, on cherche à connaître le nombre de solutions et non seulement l'existence d'une solution. Il existe aussi une hiérarchie des classes de comptage : les plus connues sont FP (le nombre de solutions est calculé en temps polynomial par un algorithme déterministe) et #P (le nombre de solutions est calculé en temps polynomial par un algorithme non déterministe). Il existe aussi des problèmes #P-complets.
En filtrage et unification équationnels, nous sommes intéressés non seulement par la question de l'unifiabilité de deux termes dans la théorie (problème de décision), mais aussi par le nombre de filtres ou unificateurs principaux (problème de comptage). Pour les théories finitaires (celles qui ont toujours un nombre fini d'unificateurs), il est naturel de poser la question de la complexité des problèmes de comptage. Or pour prouver la #P-difficulté d'un problème de comptage en unification équationnelle, nous avons besoin de trouver une réduction parcimonieuse (c'est-à-dire qui préserve le nombre d'éléments comptés) à partir d'un

[5] *From regular expressions to deterministic automata* G. Berry and R. Sethi - TCS, 48:117–126, 1986.

problème #P-complet.
La #P-complétude de plusieurs problèmes de filtrage équationnel a été prouvée en collaboration avec Phokion Kolaitis [22].

Langages de motifs et systèmes de réécriture de mots: cette recherche menée par G. Kucherov en collaboration avec M. Rusinowitch (projet PROTHÉO) porte sur les *langages de motifs*. Ce sont des langages de mots qui contiennent des occurrences de motifs, la notion de motif pouvant avoir plusieurs définitions possibles. Dans [10], les motifs considérés sont des *mots avec variables*. Un motif est donc contenu dans un mot si le premier a une instance qui est un facteur du dernier. Il a été prouvé qu'il n'existe pas d'algorithme pour tester si toutes les instances d'un motif donné contiennent les motifs d'un ensemble donné. Notons que cette propriété se ramène à la propriété de *réductibilité inductive* pour les *systèmes de réécriture de mots avec variables*. Cette propriété joue un rôle important dans la théorie de la réécriture et ses applications.
Cette étude a été poursuivie et la complexité du problème ci-dessus a été étudiée pour la classe des motifs *linéaires*, c'est-à-dire ceux qui ne contiennent qu'une occurrence de chaque variable. Il a été prouvé que le problème dans ce cas est co-NP-complet. Le problème de finitude de l'ensemble des mots ne contenant pas les motifs d'un ensemble donné est également co-NP-complet [24].
Ces travaux théoriques ont également débouché sur de nouveaux algorithmes ayant des applications pratiques intéressantes (voir section 3.2.2).

3.2.2 Développement d'algorithmes en calcul symbolique

Génération aléatoire: le système GAÏA de génération aléatoire de structures combinatoires a été complètement réécrit en collaboration avec le projet ALGO (notamment Eithne Murray) afin d'en faire un logiciel robuste et fiable [15]. Ce système doit être intégré à la prochaine version de MAPLE, sous le nom COMBSTRUCT. Le cas étiqueté a fait l'objet d'une étude particulière [9]. Des travaux antérieurs réalisés par P. Zimmermann sont parus cette année : l'un concerne une étude sur le problème des n reines, réalisée avec Igor Rivin et Ilan Vardi, lors de la visite de ce dernier au sein du projet ALGO [12]; l'autre décrit le

programme GFUN de manipulation de fonctions holonomes, réalisé avec Bruno Salvy [14].

Algorithmes de recherche de motifs dans les séquences: les travaux théoriques de G. Kucherov sur les langages de motifs ont donné lieu à de nouvelles idées dans le domaine de la recherche de motifs dans les séquences. Plus précisément, il s'agit de la reconnaissance la plus efficace possible de l'occurrence dans une (longue) séquence (une chaîne de caractères) d'un motif appartenant à un ensemble donné. Ici un motif doit être compris comme un certain nombre de (petites) séquences, séparées par un symbole spécial autorisé à s'identifier avec n'importe quelle chaîne de caractères. Ce problème est intéressant sous plusieurs aspects. Par exemple, il généralise le problème du filtrage dynamique (*dynamic dictionary matching*) qui a suscité beaucoup de travaux ces dernières années. Outre son intérêt théorique, un algorithme efficace aura des applications pratiques importantes en biologie moléculaire dans l'analyse des séquences génétiques.
G. Kucherov a développé en collaboration avec M. Rusinowitch (projet PROTHÉO) un algorithme qui résout le problème en temps linéaire par rapport à la taille totale des motifs et à la séquence à analyser. L'algorithme est basé sur la structure de données appelée DAWG (*Directed Acyclic Word Graph*) par opposition aux autres algorithmes de filtrage dynamique qui utilisent soit l'arbre de suffixes (*suffix tree*) soit un automate du type Aho-Corasick.
Une implantation de l'algorithme est envisagée pour être testée sur des données biologiques réelles. Plus généralement, on s'attend à ce que ce travail serve de point de départ pour mettre en place des travaux plus systématiques et plus larges sur l'application de méthodes syntaxiques à l'analyse du génome.

3.3 Preuve de programmes et de circuits

Participants : Boutheina Chetali, Stefan Krischer, Pierre Lescanne

L'application des compétences acquises en démonstration automatique est un axe important de la recherche dans EURÉCA. Outre les deux applications que sont la preuve de programmes parallèles et de circuits, il faut aussi inclure les travaux cités précédemment sur les utilisations de COQ.

3.3.1 Preuve de programmes parallèles

Dans la lignée du travail réalisé dans le cadre de la vérification formelle des programmes concurrents spécifiés en UNITY [34], B. Chetali s'est intéressée à la preuve d'un protocole de communication, à savoir le protocole du *Bit Alterné*. Elle s'est proposée de *vérifier* la preuve du protocole, telle qu'elle a été proposée par Chandy et Misra [6].
Ce travail s'est organisé en deux grandes étapes. Tout d'abord,une étape de conception; il a fallu concevoir une preuve manuelle qui a conduit à développer les grandes lignes d'une preuve détaillée et complète. Puis, la preuve mécanique dans LARCH PROVER a constitué une étape importante dans le processus de preuve du fait du caractère *humain*, donc sujet à erreurs, de la preuve manuelle. Le but de ce travail était de montrer la faisabilité de la preuve de correction mécanique d'un protocole de communication. Cela a aussi permis de valider notre choix tant pour le prouveur LP que pour l'environnement de spécification UNITY.
Du point de vue méthodologique, ce travail a mis en évidence la différence entre les preuves de *vivacité* et celles de *sécurité*.

3.3.2 Vérification des circuits digitaux

S. Krischer a soutenu le 18 mars 1994 sa thèse de doctorat intitulée *Méthodes de vérification de circuits digitaux*. Il a ensuite poursuivi un travail de programmation sur son logiciel FANCY de vérification de machines à nombre fini d'états, dites machines de Mealy. Ceci a conduit à la version 1.1 de FANCY qui dispose maintenant d'un nouvel algorithme de test d'équivalence d'automates, la méthode de *co-factorisation* proposée par Touati et al. au colloque ICCAD'90. Ainsi FANCY reconnaît automatiquement des machines identiques en utilisant un format canonique. De plus, l'interface utilisateur de FANCY a été améliorée dans le sens de la convivialité.
S. Krischer est aujourd'hui professeur assistant à l'Université de Trêves (Allemagne), d'où il entretient une collaboration active avec l'INRIA-Lorraine.

[6] *Parallel Program Design: A Fundation* Addison-Wesley (1988)

4 Actions industrielles

À la suite au stage effectué par B. Chetali dans le laboratoire de la direction de la Recherche et des Programmes de Bull, dans l'équipe de Dominique Bolignano, des visites et échanges permanents ont lieu. La collaboration porte actuellement sur l'optimisation de la formalisation de UNITY en LP, en particulier le problème des affectations, et le développement de quelques idées sur la méthodologie des preuves de propriétés de vivacité.

5 Actions nationales et internationales

P. Lescanne est vice-président de SPECIF (Société des Personnels Enseignants et Chercheurs de France), directeur du GDR-PRC *Programmation* et directeur du centre Charles Hermite (Centre Lorrain de Compétence en Modélisation et Calcul à Haute Performance).

N. Hermann était membre du Comité d'organisation de la 12^{eme} *Conférence Internationale sur la Déduction Automatique* (CADE-12), organisée à Nancy en juin 1994.

R. Amadio est responsable pour le site de Nancy du projet CNRS inter-PRC *Modèles logiques du calcul* (coordinateur: M. Parigot, sites principaux: Marseille, Nancy, Orsay, Paris VII). Avec G. Boudol (INRIA-Sophia), il anime le pôle lambda-calcul et parallélisme de ce projet.
R. Amadio est également associé à la Basic Research Action *Concurrency and Functions: Evaluation and Reduction* (Confer) (responsable: J.J. Lévy, sites principaux: CWI, ECRC, Imperial College, INRIA-Rocquencourt, INRIA-Sophia, LIENS-Paris, Universités d'Édimbourg et de Pise). Dans le cadre de ce projet, R. Amadio a visité le projet FACILE à l'ECRC pour une période de six mois et le projet Formal Methods à SICS pour une période de trois semaines.

Le projet a accueilli dans le cadre de ses séminaires (communs avec le projet PROTHÉO) plusieurs chercheurs dont Leo Bachmair, Alan Bundy, Claude Marché, Wayne Snyder.

6 Diffusion des résultats

6.1 Diffusion de produits

Les logiciels développés par le projet, FANCY, ORME et GAÏA, sont diffusés sous ftp (`ftp.loria.fr`, répertoire `pub/loria/eureca`).

De plus, P. Zimmermann s'est consacré avec Bruno Salvy et Claude Gomez à la rédaction d'un livre sur l'utilisation du calcul formel. À l'inverse d'un manuel de référence, ce livre est construit autour des principales classes de problèmes rencontrés dans le domaine scientifique. Pour chaque classe, il est indiqué quels types de problèmes les systèmes actuels savent résoudre, et quelles sont leurs limites. Tout cela est illustré par des exemples en MAPLE.

6.2 Actions d'enseignement

6.2.1 Enseignement

Plusieurs membres du projet ont participé aux enseignements du DEA d'informatique de l'UHP-Nancy I :
• *Environnements de Preuves et Réécriture* (module de techniques avancées): N. Hermann
• *Terminaison* (module de spécialisation): A. Cichon et P. Lescanne
• *Logique 2* (module de techniques avancées): A. Cichon.

En outre, A. Cichon a assuré la partie logique du cours *Mathématiques en Informatique* en licence d'informatique et en première année à l'Ecole Supérieure d'Informatique et Applications de Lorraine et la partie logique du cours *Mathématiques et Algorithmique* en maîtrise d'informatique.

6.2.2 Séminaires et formation permanente

A. Weiermann a donné une série de cinq séminaires sur les beaux préordres et la terminaison.

P. Zimmermann a organisé avec Claude Gomez et Bruno Salvy un cours de trois jours à l'INRIA-Rocquencourt sur le calcul formel (*Calcul formel pour les applications - Utilisation du système Maple*).

6.2.3 Jurys de thèse

A. Cichon a été rapporteur des habilitations de Nicolas Hermann (Nancy) et Didier Galmiche (Nancy). Il a également été rapporteur des thèses de Joseph Rouyer (Nancy), Adel Bouhoula (Nancy) et Elias Tahhan-Bittar (UCB, Lyon).

N. Hermann a été rapporteur de la thèse de doctorat d'Olivier Gasquet (Université Paul Sabatier, Toulouse).

P. Lescanne a été rapporteur des thèses de doctorat d'Adel Bouhoula, Naïma Brown et Abdelillah Mokkedem (Nancy). Il a participé aux jurys des habilitations de Roberto Amadio et Nicolas Hermann, ainsi qu'aux jurys de thèse de Stefan Krischer et Joseph Rouyer (Nancy).

6.3 Participation aux manifestations

La plupart des membres du projet a participé aux conférences CADE-12, à Nancy, et LICS, à Paris.

A. Cichon a participé au EC Workshop on *Proof Theory and Computation* à Leeds (Angleterre).

N. Hermann a présenté à CADE-12 son travail sur la #P-complétude et le filtrage équationnel [22].

S. Krischer a présenté FANCY lors de *La Science en Fête* et au colloque international *Theorem provers for circuit design* (TPCD'94) à Bad Herrenalb, en Allemagne.

G. Kucherov a participé aux colloques suivants :
- *Symposium on Applied Computing* à Phoenix (États-Unis) où il a présenté le travail [10],
- la réunion du groupe LANFOR (groupe national de travail sur le thème de langages formels) où il a exposé son travail sur les langages d'arbres,
- *International Colloquium on Trees in Algebra and Programming* (CAAP) à Édimbourg (Écosse) où il a présenté le travail [23],
- séminaire franco-russe *Complexity, Probability and their Applications* à l'Université Lomonosov de Moscou où il a fait un exposé,
- *International Colloquium on Automata, Languages and Programming (ICALP)* et *International Workshop on Conditional Term Rewriting Systems (CTRS)* à Jérusalem (Israël), où il a présenté le travail [24].

Dans le cadre de la mise en œuvre des travaux sur le génome G. Kucherov a également participé au Workshop *Combinatorial Methods of Genome Rearrangement* à l'Université de Californie de Los Angeles en mars 1994 et à la réunion du groupe de travail national français *Recherche de Motifs dans les Séquences* en septembre 1994.

P. Lescanne a assisté à la conférence *Principles of programing Languages* à Portland (Oregon) et au Workshop *Atlantique* qui suivait, pour y présenter ses travaux.

P. Zimmermann a effectué une démonstration du logiciel GAÏA lors du colloque Gascom à Bordeaux en janvier 1994. Il a également été invité à faire une présentation de ses travaux sur la génération aléatoire aux journées annuelles de la DMV (*Deutsche Mathematische Vereinigung*) à Duisbourg en septembre 1994.

6.4 Activités extérieures

R. Amadio a donné les exposés suivants:
- *Translating Core Facile*, à la Ludwig-Maximilian Universität de Munich, à l'Atelier Confer Munich et au SICS Stockholm,
- *Localities and Failures*, à l'Atelier Confer Munich et au SICS Stockholm,
- *Some results on a concurrent and distributed λ-calculus*, à l'Université de Nice,
- *A Hennessy-Milner logic for higher-order bisimulation*, à l'Atelier Confer Londres.

Z. Benaïssa, B. Chetali et F. Leclerc ont présenté leurs recherches aux journées annuelles du GDR Programmation.

B. Chetali a présenté ses travaux au groupe Basic Research Compass.

A. Cichon a donné un séminaire au département de mathématiques de l'Université de Muenster (Allemagne).

G. Kucherov a donné des séminaires à l'Université de l'État de New-York à Stony Brook, à l'Université de Marne-la-Vallée (Institut Gaspard Monge) et à l'Université Technique de Berlin. Il a également visité l'Université de l'État de New-York à Albany (P.Narendran, D.Rosenkrantz) et le *SRI* International à Palo Alto (J. Meseguer).

P. Lescanne a donné des séminaires à l'Université de Stanford, de Santa Cruz et à l'IRIT de Toulouse.

6.5 Réunions diverses

Journées au vert : dans le prolongement de l'expérience qui s'est établie depuis quelques années, les membres du projet se sont réunis les 26 mai et 28 novembre, pour que chacun présente ses progrès de recherche et ceux de ses collaborateurs directs. En tout, une trentaine d'exposés a eu lieu, ce qui a permis à tous les participants de se faire une image instantanée de l'activité du projet et a suscité de nouvelles collaborations, au delà de celles qui existaient déjà. De plus, cette réunion constitue un excellent exercice pour les jeunes chercheurs peu habitués à présenter leurs travaux car cela leur impose à échéances régulières de faire le point et de synthétiser leurs progrès récents.

7 Publications

Thèses

[1] R. Amadio, *Trois Lambda-Calculs*, Habilitation à diriger des recherches, Université Henri Poincaré Nancy 1, 1994.

[2] N. Hermann, *Divergence des systèmes de réécriture et schématisation des ensembles infinis de termes*, Habilitation ' diriger de recherches, Université Henri Poincaré Nancy 1, mars 1994.

[3] S. Krischer, *Méthodes de vérification de circuits digitaux*, thèse de doctorat, Institut National Polytechnique de Lorraine, mars 1994.

[4] J. Rouyer, *Développements d'algorithmes dans le calcul des constructions*, thèse de doctorat, Université Henri Poincaré Nancy 1, mars 1994.

[5] E. Tahhan Bittar, *Bornes Supérieures Inductives pour la Terminaison d'Algorithmes*, thèse de doctorat, Université Claude-Bernard, Lyon 1, octobre 1994.

Articles et chapitres de livre

[6] V. M. Antimirov, P. D. Mosses, «Rewriting Extended Regular Expressions (Short Version)», *in : Developments in Language Theory – At the Crossroads of Mathematics, Computer Science and Biology*, G. Rozenberg et A. Salomaa (réd.), World Scientific, Singapore, 1994, p. 195–209.

[7] V. M. Antimirov, P. D. Mosses, «Rewriting Extended Regular Expressions», *Theoretical Computer Science 141*, 1995, À paraître.

[8] W. Buchholz, A. Cichon, A. Weiermann, « A Uniform Approach to Fundamental Sequences and Hierarchies », *Mathematical Logic Quarterly*, 40, 1994, p. 273–286.

[9] P. Flajolet, P. Zimmermann, B. V. Cutsem, « A Calculus for the Random Generation of labelled Combinatorial Structures », *Fundamental Studies of Theoretical Computer Science 132*, p. 1–35.

[10] G. Kucherov, M. Rusinowitch, « On the ground reducibility problem for word rewriting systems with variables », *Information Processing Letters*, 1994, À paraître.

[11] P. Lescanne, « On Termination of one Rule Rewrite Systems », *Theoretical Computer Science 132*, 1-2, 1994, p. 395–401.

[12] I. Rivin, I. Vardi, P. Zimmermann, « The n-Queens Problem », *American Mathematical Monthly 101*, 7, 1994, p. 629–639.

[13] J. Rouyer, P. Lescanne, « Verification and Programming of First-order Unification in the Calculus of Constructions with Inductive Types », *Journal of Logic and Computation*, 1994, À paraître.

[14] B. Salvy, P. Zimmermann, « Gfun: A Maple Package for the Manipulation of Generating and Holonomic Functions in One Variable », *ACM Transactions on Mathematical Software 20*, 2, juin 1994, p. 163–177.

[15] P. Zimmermann, « Gaïa: a package for the random generation of combinatorial structures », *MapleTech 1*, 1, 1994, p. 38–46.

Communications à des congrès, colloques, etc.

[16] R. Amadio, S. Prasad, « Localities and Failures », *in : Proc. FST-TCS 94, Madras*, SLNCS, 1994. À paraître- version priliminaire parue dans ECRC-TR-94-18,Munich, disponible sous ftp.ecrc.de.

[17] V. M. Antimirov, « Partial Derivatives of Regular Expressions and Finite Automata Constructions », STACS'95, 1994. À paraître.

[18] D. Briaud, « An explicit *Eta* rewrite rule », *in : Int. Conf. on Typed Lambda Calculus and Applications*, 1995.

[19] F. Bronsard, U. S. Reddy, R. W. Hasker, « Induction using Term Orderings », *in : Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, A. Bundy (réd.), *Lecture Notes in Computer Science (in Artificial Intelligence)*, *814*, Springer-Verlag, p. 102–117, juin 1994.

[20] W. Charatonik, L. Pacholski, « Negative Set Constraints with Equality », *in : Ninth Annual IEEE Symposium on Logic in Computer Science*, p. 128–136, July 1994.

[21] W. Charatonik, L. Pacholski, «Set Constraints with Projections are in NEXPTIME», *in : 35th Annual IEEE Symposium on Foundations of Computer Science*, p. 642–653, November 1994.

[22] N. Hermann, P. Kolaitis, «The complexity of counting problems in equational matching», *in : Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, A. Bundy (réd.), *Lecture Notes in Computer Science (in Artificial Intelligence)*, *814*, Springer-Verlag, p. 560–574, juin 1994.

[23] D. Hofbauer, M. Huber, G. Kucherov, «Some Results on Top-context-free Tree Languages», *in : Proceedings of the 19th International Colloquium on Trees in Algebra and Programming*, S. Tison (réd.), *Lecture Notes in Computer Science*, Springer-Verlag, 1994.

[24] G. Kucherov, M. Rusinowitch, «Complexity of Testing Ground Reducibility for Linear Word Rewriting Systems With Variables», *in : Proceedings of the 4th International Workshop on Conditional (and Typed) Term Rewriting Systems*, Jirusalem (Israel), 1994. À paraître dans la sirie Lecture Notes in Computer Science de Springer Verlag.

[25] F. Leclerc, «Termination proof of term rewriting systems with the multiset path ordering. A complete development in the System Coq», *in : Proceedings of the International Conference Typed Lambda Calculus and Applications*, *Lecture Notes in Computer Science*, Springer-Verlag, 1995.

[26] P. Lescanne, J. Rouyer-Degli, «Explicit Substitutions with de Bruijn's Levels», *in : Proceedings 6th Conference on Rewriting Techniques and Applications, Kaiserslautern (Germany)*, J. Hsiang (réd.), 1995. À paraître.

[27] P. Lescanne, «From $\lambda\sigma$ to $\lambda\upsilon$, a journey through calculi of explicit substitutions», *in : Proceedings of the 21st POPL conference, Portland (Or., USA)*, H. Boehm (réd.), ACM, p. 60–69, 1994.

Rapports de recherche et publications internes

[28] O. Aït-Mohamed, «Vérification de l'équivalence du π-calcul dans HOL», *Research Report n° 2412*, Institut National de Recherche en Informatique et Automatique, Novembre 1994.

[29] R. Amadio, O. Ait-Mohamed, «An Analysis of π-calculus bisimulations», *Technical Report n° 94-2*, European Computer-Industry Research Center, Munich, 1994, disponible sous ftp.ecrc.de.

[30] R. Amadio, P. Curien, «Selected Domains and Lambda Calculi», *Technical Report n° 161*, INRIA-Lorraine, 1994.

[31] R. Amadio, M. Dam, «Reasoning about higher-order processes», *Research Report n° 94-18*, SICS, 1994, disponible sous http://www.sics.se/.

[32] R. Amadio, « Translating Core Facile », *Technical Report n° ECRC-94-3*, ECRC, Munich, 1994, disponible sous ftp.ecrc.de.

[33] B. Chetali, « Faulty channels and correct protocols: A formal proof of a concurrent program using the Larch Prover », *rapport de recherche*, Centre de Recherche en Informatique de Nancy, 1994, Soumis.

[34] B. Chetali, « Formal Verification of Concurrent Programs: How to specify UNITY using the Larch Prover », *rapport de recherche*, Centre de Recherche en Informatique de Nancy, 1994, Soumis au 10^{eme} International Workshop ADT + COMPASS 94.

[35] E. A. Cichon, M. Rusinowitch, S. Selhab, « Cut Elimination in Sequent Calculus and Rewriting », *rapport de recherche*, Centre de Recherche en Informatique de Nancy, Mars 1994.

[36] P. Lescanne, J. Rouyer-Degli, « The Calculus of Explicit Substitutions $\lambda\upsilon$ », *Rapport interne*, Centre de Recherche en Informatique de Nancy, 1994.

[37] P. Lescanne, « An introduction to ORME », *rapport de recherche n° 94-R-164*, CRIN, octobre 1994.

Divers

[38] V. M. Antimirov, « Rewriting Regular Inequalities », Octobre 1994, Soumis.

[39] Z. Benaissa, « A suitable environments for an efficient calculus of explicit subsitutions », Octobre 1994, Soumis.

[40] E. Cichon, E. Tahhan-Bittar, « Ordinal Recursive Bounds for Higman's Theorem », Soumis ' TCS.

8 Abstract

Research in Computer Science in Nancy is conducted jointly under the auspices of two institutions: (1) the CRIN (Centre de Recherche en Informatique de Nancy), a laboratory associated with the CNRS and comprising Nancy's main university teaching establishments, and (2) the Inria (Institut National de Recherche en Informatique et Automatique). Researchers work in formal groups. *Euréca* is the chosen name of such a group headed by Pierre Lescanne, and comprises research and teaching staff employed, in particular, by the CNRS, the University and the INRIA-Lorraine.

The group's research activities are linked by a central theme: *Proof, Symbolic Computation and Logic*. There are three main lines of research:

1. The λ - calculus, Mathematical Logic and Term Rewriting

λ - calculi with explicit substitution are being investigated as are the semantics of parallelism in λ - calculi. Works is also in progress on the calculus of constructions and on inductive and deductive reasoning.

Applications of ordinal structures to termination proofs of rewrite systems, proof-theoretic analysis of termination and characterisations of solution sets for unification problems are being investigated. String rewriting systems with variables and tree languages are also being studied.

2. Algorithmics

Research is under way on the complexity analysis of algorithmic problems arising in symbolic computations as well as on the development of efficient algorithms for those problems. Specifically, the complexity of the inclusion problem for general regular languages and for some subclasses (*pattern languages*) is being studied. Also, the problem of counting the number of most general matchers and unifiers with respect to different equational theories is being investigated using the notion of counting complexity classes. Various algorithms of random generation of combinatorial data structures have been implemented in the GAÏA package. Finally, new efficient algorithms for some general pattern matching problems are being developed, and which are expected to be applicable to the analysis of biological sequences.

3. Proof and Verification

Researchs in two directions is under way, namely, proofs of program correctness and verification of digital circuits and concurrent programs.

Table des matières