INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet LogiCal

Logique et Calcul

Futurs

THÈME 2A

Table des matières

1. Composition de l'équipe

Le projet LogiCal est un projet commun qui rassemble des chercheurs de l'INRIA-Rocquencourt et du Laboratoire de Recherche en Informatique de l'Université de Paris XI.

Responsable scientifique
Gilles Dowek [DR INRIA]

Co-responsable scientifique
Christine Paulin [Professeur à l'Université de Paris XI]

Responsable permanent
Benjamin Werner [CR INRIA]

Assistante de projet en commun avec le projet Verso
Gina Grisvard [TR INRIA]

Personnel INRIA
Bruno Barras [CR]
Hugo Herbelin [CR]
Claude Marché [CR, détaché de l'enseignement supérieur]

Ingénieur associé
Olivier Desmettre [Ingénieur Développement Logiciels]

Conseiller Scientifique
Pierre-Louis Curien [DR CNRS, PPS]

Collaborateurs extérieurs
Jean Duprat [Maître de conférences à l'ENS-Lyon]
Gérard Huet [DR INRIA]

Personnel Paris XI
Judicaël Courant [Maître de Conférences à l'Université de Paris XI]
Jean-Pierre Jouannaud [Professeur à l'Université de Paris XI]

Personnel CNRS
Jean-Christophe Filliâtre [CR]

Chercheur Invité
Areski Nait Abdallah [University of Western Ontario]

Post-doctorants
Xavier Urbain [INRIA]
Frédéric Blanqui [École polytechnique]

Doctorants
Jacek Chrzaszcz [Thèse en co-tutelle LRI / Université de Varsovie]
Pierre Corbineau [Élève à l'École Normale Supérieure]
Benjamin Grégoire [Allocataire MENRT, en commun avec le projet Cristal]
Olivier Hermant [Bourse DGA]
Antoine Kremer
Pierre Letouzey [Élève à l'École Normale Supérieure]
Patrick Loiseleur
Alexandre Miquel [ATER à l'Université de Paris XI]
Julien Narboux [Élève à l'École Normale Supérieure de Cachan]
Nicolas Oury [Élève à l'École Normale Supérieure de Lyon]
Clément Renard [Allocataire-moniteur à l'Université de Paris XI]
Julien Signoles [Allocataire MENRT]

Stéphane Vaillant [Allocataire MENRT]
Daria Walukiewicz-Chrzaszcz [Thèse en co-tutelle LRI / Université de Varsovie]

2. Présentation et objectifs généraux

Le but des recherches menées dans le projet est de construire des *systèmes de traitement de démonstrations mathématiques*. Ces systèmes peuvent vérifier l'absence d'erreurs dans une démonstration, ils peuvent aider les utilisateurs à construire des démonstrations interactivement, en rechercher de manière automatique, les archiver, les exporter vers d'autres logiciels, ...

Utiliser un système informatique pour traiter des démonstrations mathématiques permet de se convaincre avec un grand degré de certitude que ces démonstrations ne comportent pas d'erreurs. On peut en particulier se convaincre ainsi de l'exactitude des arguments justifiant la correction de matériels et de logiciels. Cela est particulièrement important dans les domaines applicatifs où un défaut de fonctionnement met la vie humaine, la santé ou l'environnement en péril et dans celles qui mettent en jeu des sommes d'argent importantes : l'informatique médicale, les transports, les télécommunications, le commerce électronique, l'informatique en réseau, ... Utiliser un système de traitement de démonstrations permet également de construire des démonstrations de grande taille, par exemple des démonstrations utilisant des polynômes formés de plusieurs centaines de monômes. Enfin, cela participe à la quête d'une nouvelle forme d'exactitude et de rigueur dans la rédaction mathématique : le point où rien n'est sous-entendu, et où le lecteur peut donc être remplacé par un programme.

Le principal axe de nos travaux est le développement du système Coq qui a aujourd'hui une communauté importante d'utilisateurs industriels et académiques. Nous croyons cependant que le développement d'un système ne peut pas s'effectuer sans une réflexion en aval sur les usages spécifiques que l'on fait de ce système dans certains domaines (quand on fait de la géométrie réelle, des preuves de programmes impératifs ou objets, des preuves de protocoles cryptographiques, ...) et en amont sur les questions relatives à la formalisation des mathématiques (sur la représentation des démonstrations, sur l'intégration d'un langage de programmation dans un formalisme mathématique, sur la notion de variable liée, ...). Ces recherches s'articulent autour de deux notions clés : celle de raisonnement logique et celle de calcul. Ce sont ces deux notions qui donnent son nom au projet LogiCal.

3. Fondements scientifiques

3.1. La formalisation des mathématiques

Mots clés : *langage mathématique*, *langage de programmation*, *logique des prédicats*, *théorie des ensembles*, *démonstration constructive*, *algorithme*.

Un langage traditionnel pour formaliser les mathématiques est la théorie des ensembles, exprimée dans la logique des prédicats du premier ordre. Cependant ce cadre ne répond pas parfaitement à nos besoins. Il a en effet été développé au début du vingtième siècle, pour étudier mathématiquement les propriétés du raisonnement mathématique. Pour cela, la possibilité d'y formaliser les mathématiques « en principe » est suffisante. Les questions qui se posent à nous aujourd'hui ne sont plus celles de la formalisation « en principe » mais celles de la formalisation « en faits ».

Cela amène à étudier des variantes de la théorie des ensembles qui proposent un langage riche et compact pour exprimer les objets mathématiques (par exemple les fonctions) en particulier des langages comportant des symboles lieurs comme le λ-calcul. Des outils pour étudier les propriétés de ces langages sont la notion d'indice de De Bruijn et celle de substitution explicite.

Le fait de devoir écrire des démonstrations mathématiques dans un très grand détail amène aussi à étudier des formalismes qui articulent raisonnement et calcul afin de ne pas devoir écrire les démonstrations des propositions dont la vérité peut s'établir par un simple calcul.

S'intéresser à des développements mathématiques portant sur des programmes et leurs propriétés amène à s'intéresser à l'aspect effectif des mathématiques et en particulier au rapport entre la notion de démonstration constructive et celle d'algorithme, afin que de la démonstration constructive d'existence d'une fonction, on puisse tirer un algorithme la calculant. Plus généralement, nous défendons la thèse qu'il n'y a pas de frontière nette entre le langage mathématique et les langages de programmation et qu'un langage mathématique moderne doit contenir comme sous-langage un langage de programmation.

3.2. Le Calcul des Constructions Inductives

Mots clés : *Calcul des Constructions Inductives*.

Le système **Coq** est une implémentation d'un formalisme appelé le Calcul des Constructions Inductives.

Ce formalisme permet de développer des preuves dans un calcul des prédicats d'ordre supérieur et en cela s'apparente aux logiques utilisées dans les assistants de preuve HOL ou PVS. Ces logiques sont bien adaptées au raisonnement abstrait et permettent en particulier une représentation naturelle des types de données structurés et des définitions par point-fixe de prédicats.

Cependant, le Calcul des Constructions Inductives se distingue de la logique d'ordre supérieur introduite par Church par plusieurs points que nous exploitons dans le système **Coq** :

- Les preuves sont représentés par des λ-termes qui sont des objets à part entière. Vérifier une preuve se fait par un algorithme de vérification du type du λ-terme associé.
- Le langage permet de définir des fonctions comme des algorithmes (en utilisant un sous-ensemble d'un langage de programmation fonctionnel). Le raisonnement sur ces fonctions peut alors, dans certains cas, se faire par calcul plutôt que de manière équationnelle.
- La logique utilisée est constructive (elle n'utilise pas le tiers-exclu ni le raisonnement par l'absurde). Ceci permet d'interpréter une preuve d'existence d'un objet comme un algorithme permettant de calculer cet objet.

3.3. Les environnements de développement interactif de preuves

Mots clés : *tactique*.

Le système **Coq** permet de construire des théories mathématiques en introduisant des définitions, des hypothèses, en énonçant des théorèmes et enfin en donnant une démonstration de ces théorèmes. L'utilisateur peut construire ses démonstrations semi-automatiquement, à l'aide de procédures de décision et de *tactiques*.

Une tactique est un programme, livré avec le système ou construit par l'utilisateur, qui transforme une proposition à démontrer en un ensemble de nouvelles propositions suffisantes. L'implantation des tactiques et des procédures de décision repose sur des algorithmes d'unification, de gestion de bases de théorèmes et de calcul (normalisation, réécriture).

La certification des preuves repose sur des algorithmes de typage et de calcul. Celle des méthodes de décision repose sur des techniques dite de « réflexion » consistant à internaliser la preuve de correction de la méthode.

3.4. La Déduction Modulo

Mots clés : *logique*, *calcul*, *déduction modulo*.

Le Calcul des Constructions Inductives utilise une règle de conversion qui identifie des propositions équivalentes modulo un ensemble de règles de calcul, en ce sens que toute preuve de l'une devient automatiquement une preuve de l'autre, et que l'équivalence des deux propositions n'a donc pas besoin d'être démontrée. Cette articulation entre raisonnement et calcul peut se formuler dans un cadre beaucoup plus large que le Calcul des Constructions Inductives. Ce cadre est une alternative à la logique des prédicats du premier ordre appelé *la Déduction Modulo* dans lequel une théorie se définit par un ensemble d'axiomes et de règles de calcul.

La Déduction Modulo permet, entre autres choses, de concevoir des algorithmes de démonstration automatique qui exploitent cette opposition entre raisonnement et calcul. Elle permet également d'entrevoir une théorie unifiée de l'élimination des coupures. Une question ouverte, concernant la Déduction Modulo, est celle de la caractérisation des théories qui peuvent s'exprimer avec des règles de calcul uniquement.

3.5. Les démonstrations et les programmes

Mots clés : *isomorphisme de Curry-Howard*, *réalisabilité*, *logique intuitionniste*, *Calcul des Constructions Inductives*.

La principale originalité du Calcul des Constructions Inductives est que les démonstrations y sont des objets au même titre que les nombres, les fonctions ou les ensembles. Ainsi un entier pair est représenté par un couple formé d'un entier et d'une démonstration que cet entier est pair. Une autre originalité de ce formalisme est que chaque terme exprimant un objet d'un type de données, peut se réduire sur une valeur de ce type de données.

Ces propriétés combinées permettent de concevoir la spécification d'un programme comme une relation liant la valeur d'entrée et la valeur de sortie de ce programme. En effet, si $Q(x, y)$ est une telle relation, une démonstration dans le Calcul des Constructions Inductives de la totalité de cette relation (c'est-à-dire de la proposition $\forall x \exists y \quad Q(x, y)$) est une fonction qui, à tout objet x associe un objet y et une démonstration de $Q(x, y)$. À partir de la démonstration de totalité, il est possible d'exprimer dans le calcul à la fois un programme fonctionnel et sa démonstration de correction. La fonction f appliquée à l'entrée x se réduira pour fournir la sortie y.

Mais les démonstrations ne sont pas, en général, des programmes efficaces, et il est nécessaire, en pratique, d'éliminer certaines parties de ces démonstrations non pertinentes pour le calcul : cette étape s'appelle l'extraction de programme. Les programmes extraits peuvent alors être traduits dans un langage de programmation ordinaire tel que ML, puis compilés en langage machine. Dans cette approche, la spécification du programme est une formule mathématique et le programme certifié est obtenu à partir de la démonstration de cette formule.

Cette interprétation est particulièrement adaptée à la preuve de programmes fonctionnels. Elle s'étend aujourd'hui à d'autres styles de programmation, tels la programmation impérative et la programmation objets.

4. Domaines d'application

Mots clés : *santé*, *transports*, *télécommunications*, *commerce électronique*, *informatique en réseau*.

Les systèmes de traitement de démonstrations, et plus généralement les outils de méthodes formelles, sont utiles dans les domaines où la sûreté (c'est-à-dire l'absence d'erreurs involontaires) et la sécurité (c'est-à-dire la protection contre les attaques malveillantes) des systèmes informatiques sont centrales. En particulier, les applications où un défaut de fonctionnement met la vie humaine, la santé ou l'environnement en péril et celles qui mettent en jeu des sommes d'argent importantes. Nos domaines d'application sont naturellement l'informatique médicale, les transports, les télécommunications, le commerce électronique, l'informatique en réseau, ...

Le système **Coq** est utilisé pour modéliser des protocoles cryptographiques de commerce électronique, des politiques de sécurité dans le cadre d'applications Java sur cartes à puce, des études d'algorithmes de contrôle de conformité dans les réseaux de télécommunications, des démonstrations de compilateur de langages réactifs, la certification d'algorithmes de calcul formel, ...

Ces développements sont pour la plupart menés à bien par nos partenaires, en particulier nos partenaires industriels : Trusted Logic, France Telecom, Gemplus, Schlumberger-Sema, ...

5. Logiciels

5.1. Coq

Participants : Bruno Barras, David Delahaye, Jean-Christophe Filliâtre, Benjamin Grégoire, Hugo Herbelin, Pierre Letouzey, Christine Paulin.

Le système **Coq** développé dans le projet est un système de traitement de démonstrations mathématiques qui permet de développer interactivement des spécifications et des démonstrations.

La principale originalité du système **Coq** est le formalisme utilisé qui comporte :

- une notion primitive de définitions mutuellement inductives permettant des spécifications de haut niveau soit dans un style fonctionnel en déclarant des types concrets et en définissant des fonctions par des équations représentant un calcul, soit dans un style déclaratif en spécifiant des relations à l'aide de clauses ;
- une interprétation des démonstrations comme des programmes certifiés, mise en œuvre dans une compilation des démonstrations sous forme de programmes ML mais aussi des outils pour associer un programme à une spécification et engendrer automatiquement des obligations de démonstration permettant de justifier sa correction ;
- une notion primitive de définitions co-inductives permettant de représenter directement des structures infinies rationnelles et de construire des démonstrations sur de tels objets sans passer par la notion classique de bisimulation.

Au niveau de l'architecture du système, les principales fonctionnalités sont :

- une boucle d'interaction permettant la définition d'objets mathématiques et informatiques et l'énoncé de lemmes,
- le développement interactif de preuves à l'aide d'un large ensemble extensible de tactiques se divisant en tactiques élémentaires (offrant un contrôle fin de la structure de la preuve et donc du programme sous-jacent) et tactiques de décision ou semi-décision,
- un système de bibliothèques modulaires et des outils de recherche dans les bibliothèques,
- un mécanisme intégré d'évaluation partielle ou totale des programmes écrits dans le langage de **Coq**,
- la possibilité de développer des tactiques comme des programmes Ocaml sophistiqués qui peuvent ensuite être chargés et utilisés dans l'environnement,
- l'isolement du code assurant la correction des preuves dans un noyau dont la petite taille permet d'accroître la confiance en sa correction (avec le projet en cours d'auto-certification), ainsi que l'abstraction de l'interface et des structures de données du noyau, garantissant que seul celui-ci peut construire des théories certifiées.

Parmi les développements les plus significatifs réalisés à l'aide de **Coq**, on peut mentionner :

- la modélisation du protocole d'authentification CSET utilisé en commerce électronique et la démonstration de propriétés de ce protocole,
- une démonstration de la correction du compilateur du langage réactif Lustre utilisé dans l'environnement industriel Scade,
- une démonstration du noyau critique de l'environnement **Coq**,
- plusieurs modélisations des propriétés du π-calcul,
- le développement de bibliothèques d'algèbre, d'analyse et de géométrie,
- une version certifiée de l'algorithme de Buchberger utilisé en Calcul Formel,
- la démonstration du théorème de d'Alembert-Gauss,
- la démonstration du théorème d'approximation de Taylor.

Le système **Coq** est disponible à l'URL http://coq.inria.fr/. Écrit en Ocaml et Camlp4, il fonctionne sur la plupart des stations de travail Unix et également sous Windows et MacOS X.

Coq est utilisé sur une centaine de sites. Nous avons des utilisateurs intensifs dans le milieu industriel (France Telecom R & D, Dassault-Aviation, Trusted Logic, Gemplus, Schlumberger-Sema, ...) dans le milieu académique en Europe (Ecosse, Hollande, Espagne, Italie, Portugal) et en France (Bordeaux, Lyon, Marseille, Nancy, Nantes, Nice, Paris, Strasbourg).

Une liste électronique (mailto:coq-club@pauillac.inria.fr) permet l'échange entre les personnes intéressées par le système.

5.2. Why

Participant : Jean-Christophe Filliâtre.

L'outil **Why** prend en entrée un programme annoté par une spécification et des propriétés (notamment variant et invariant de boucles) et produit des obligations de preuve pour différents systèmes de développement de démonstrations mathématiques, actuellement **Coq** et PVS, mais également bientôt pour une procédure de décision développée dans le projet CASSIS à l'INRIA Lorraine. **Why** prend en entrée plusieurs sortes de programmes : des programmes ML, d'une part, mêlant traits impératifs, traits fonctionnels et exceptions ; et des programmes C d'autre part. **Why** inclut par ailleurs un calcul de plus petite précondition.

Le système **Why** est disponible à l'URL http://why.lri.fr/

5.3. Krakatoa

Participants : Claude Marché, Christine Paulin, Xavier Urbain.

L'outil de traduction KRAKATOA fournit, sur l'entrée d'un programme JAVA/JAVACARD (et donc impératif) annoté en JML, un programme ayant la même sémantique mais destiné au générateur d'obligations de preuve WHY, c'est-à-dire fonctionnel avec quelques traits impératifs : exceptions et références.

Krakatoa est disponible à l'URL http://www.lri.fr/~marche/krakatoa

6. Résultats nouveaux

6.1. Les développements de démonstrations formelles et de tactiques

6.1.1. Les automates temporisés

Participante : Christine Paulin.

Mots clés : *automates temporisés*, *protocoles de télécommunications*.

Le projet RNRT CALIFE qui s'achève en décembre 2002 a pour objet la spécification, la preuve et les tests de protocoles de communication modélisés à l'aide d'automates temporisés : les p-automates. Cette structure est modélisée par une théorie **Coq** intégrée aux contributions. Un prototype réalisé par B. Tavernier de CRIL technology permet d'éditer graphiquement les automates qui sont ensuite traduits vers des model-checkers ou des assistants à la démonstration. Le travail de cette année a consisté à améliorer le modèle de traduction des automates vers **Coq** afin de prendre en compte les constructions modulaires d'automates et en particulier d'affiner le traitement des variables locales et globales lors de la synchronisation. Le prototype a pu être utilisé avec succès pour modéliser dans **Coq** une preuve paramétrique du protocole CSMA CD d'accès multiple avec détection de collision pour un nombre arbitraire d'émetteurs.

6.1.2. Les systèmes réactifs temps-réel asynchrones

Participants : Jean-Pierre Jouannaud, Antoine Kremer.

Mots clés : *système réactif temps-réel asynchrone*.

Antoine Kremer développe le système FATALIS pour la manipulation des systèmes réactifs temps-réel asynchrones. Les programmes Fatalis sont exprimés dans un fragment simple de la logique linéaire étendu par des contraintes de temps exprimées dans la théorie (classique) de Presburger. Un prototype existe qui permet de faire des preuves de sûreté dans le cas fini (par model-checking) ou infini (dans **Coq**).

6.1.3. L'analyse et la géométrie réelle

Participant : Olivier Desmettre.

Mots clés : *analyse*, *géométrie*, *nombre réel*.

Olivier Desmettre a terminé la modélisation en **Coq** de l'algorithme AILS (détection des risques de collision entre avions) et la preuve de sa correction. Par rapport à la version initialement proposée, ce développement utilise une axiomatique plus restreinte.

Olivier Desmettre a enrichi la bibliothèque des réels dans la perspective de proposer une librairie d'analyse réelle facilement utilisable par la communauté mathématique. Pour cela, Olivier Desmettre a porté son effort de développement dans quatre domaines :

- théorie des suites et des séries (complétude de $\mathbb{R}$, comparaison de convergence de suites et de séries, critères de convergence - essentiellement le critère de d'Alembert -, séries alternées)
- fonctions trigonométriques (entièrement définies désormais)
- analyse réelle et topologie (théorème des valeurs intermédiaires, théorème des accroissements finis, définitions de la racine carrée, du logarithme et de la fonction puissance, lemme de Borel-Lebesgue, caractérisation des compacts de $\mathbb{R}$, théorème de Bolzano-Weierstrass, propriétés des fonctions continues sur un compact),
- théorie de l'intégration (intégrales de Newton et de Riemann, propriétes usuelles, théorème fondamental de l'analyse),

Olivier Desmettre a également développé des tactiques pour automatiser certains raisonnements dans les réels.

Olivier Desmettre a participé à la preuve de la conjecture de Bertrand en collaboration avec Laurent Théry (INRIA Sophia-Antipolis).

6.1.4. La géométrie euclidienne

Participants : Jean Duprat, Julien Narboux, Hugo Herbelin.

Jean Duprat s'intéresse aux aspects constructifs de la géométrie plane. Pour cela, il écrit une définition axiomatique de l'ensemble des points du plan, et dans la lignée des travaux de Von Plato, il choisit un ensemble d'axiomes de décision plus faibles que le tiers exclus, mais permettant de retrouver les objets et les propriétés de la construction de Hilbert. Sur cette théorie, la règle et le compas sont définis comme constructeurs de figures. La construction de la figure apparaît alors comme la preuve constructive d'un problème en géométrie euclidienne. Le support de ce travail est le système **Coq**, qui permet de profiter de la distinction entre **Prop** et **Set**.

Julien Narboux commence un thèse sur la formalisation et la mécanisation du raisonnement géométrique sous la direction de Hugo Herbelin. Classiquement on distingue deux méthodes de preuve en géométrie : la méthode analytique qui consiste à coder le problème sous forme algébrique grâce à la notion de coordonnée et la méthode synthétique qui manipule les objets géométriques directement. Julien Narboux se propose dans cette thèse d'étudier l'approche synthétique de la preuve en géométrie. Cette approche a le double intérêt de ne pas faire appel à la construction des réels qui reste difficile à appréhender dans les assistants de preuve actuels, et de se rapprocher de l'intuition géométrique.

Le travail a commencé par une formalisation des axiomatiques de Hilbert et de Tarski en **Coq**. La preuve de l'equivalence entre les axiomatiques de Tarski, Hilbert et Von Plato est en cours.

6.1.5. La certification de programmes impératifs

Participant : Jean-Christophe Filliâtre.

Mots clés : *preuves de programmes*, *programmes impératifs*.

Jean-Christophe Filliâtre a consacré l'essentiel de son activité au développement d'un outil de vérification formelle, appelé **Why** [33]. Cet outil étend le codage réalisé par Jean-Christophe Filliâtre pendant sa thèse — la tactique `Correctness` du système **Coq**. Toutefois, l'outil **Why** se démarque plus nettement du système **Coq**, dans la mesure où il n'est plus lié à un seul assistant de preuve mais peut au contraire produire des obligations de preuve pour plusieurs systèmes : pour **Coq** et PVS actuellement, mais également bientôt pour

une procédure de décision développée dans le projet CASSIS à l'INRIA Lorraine. Cette indépendance vis-à-vis de **Coq** a notamment été motivée par des applications à des preuves d'algorithmes de contrôle aérien developpés en PVS par la NASA.

D'autre part, l'outil **Why** prend en entrée plusieurs sortes de programmes : des programmes ML, d'une part, mêlant traits impératifs, traits fonctionnels et exceptions ; et des programmes C d'autre part. Enfin, l'outil **Why** est impliqué dans la preuve de programmes Java par l'intermédiaire d'un outil indépendant (Krakatoa, développé à l'Université Paris Sud) traduisant des programmes Java annotés en JML vers la syntaxe d'entrée de l'outil **Why**. Cette combinaison est notamment mise en œuvre dans le cadre du projet européen Verificard.

Jean-Christophe Filliâtre s'est intéressé à l'algorithme de Koda et Ruskey pour la génération de certains codes de Gray (correspondant aux idéaux d'ensembles partiellement ordonnées dont les diagrammes de Hasse sont des forêts), inspiré par un travail de Knuth à paraître dans le quatrième volume de *The Art of Computer Programming*. Jean-Christophe Filliâtre a montré comment cet algorithme, habituellement codé de manière impérative, pouvait être codé à la fois de manière concise et efficace dans un langage d'ordre supérieur (i.e. où les fonctions sont des objets de première classe), là où les implantations en C proposées jusque là étaient laborieuses et complexes à analyser. Ce travail a été présenté aux Journées Francophones des Langages Applicatifs [22]. Jean-Christophe Filliâtre a ensuite collaboré avec François Pottier (INRIA Rocquencourt, projet Cristal) pour proposer deux nouvelles améliorations de ce codage. Ce travail commun est à paraître au *Journal of Functional Programming*.

6.1.6. La certification de programmes Java

Participants : Claude Marché, Christine Paulin, Xavier Urbain, Jean Duprat.

Mots clés : *preuves de programmes*, *Java*, *JavaCard*, *interprétation fonctionnelle*.

Dans le cadre du projet VERIFICARD, Claude Marché et Christine Paulin ont participé au développement des bibliothèques **Coq** pour la modélisation des objets. Deux modèles ont été développés. Le premier modèle (dit modèle par valeurs) utilise une définition coinductive des valeurs qui permet de prendre en compte la possible cyclicité des objets dans la mémoire. Le second modèle (dit modèle avec tas global) introduit un modèle classique de représentation du tas et interprète un objet comme une adresse dans le tas.

Claude Marché, Christine Paulin et Xavier Urbain ont travaillé sur l'outil de traduction KRAKATOA. Ce dernier fournit, sur l'entrée d'un programme JAVA/JAVACARD (et donc impératif) annoté en JML, un programme ayant la même sémantique mais destiné au générateur d'obligations de preuve WHY, c'est-à-dire fonctionnel avec quelques traits impératifs : exceptions et références. Ce travail a été présentée lors du workshop Verisafe à Nice en septembre 2002, puis a fait l'objet d'une soumission au journal JLAP. Un premier prototype de KRAKATOA a permis de traiter des exemples de programmes simples. Une version de distribution est prévue pour janvier 2003.

Claude Marché a défini, pour les modèles par valeurs et avec tas global, les règles de traduction en **Coq** des instructions Java. Il a implanté ces règles dans KRAKATOA.

À partir d'une étude de cas fournie dans le cadre du projet européen VerifiCard, Claude Marché et Christine Paulin réalisent une comparaison entre les deux modèles.

Xavier Urbain a travaillé à la définition de méthodes permettant de distinguer les programmes JAVA/JAVACARD contenant des *alias*, seuls susceptibles de rendre la traduction fonctionnelle incorrecte. Des critères (vérifiables statiquement) dits de *linéarité* ont été proposés afin de pouvoir sélectionner, parmi le modèles COQ de représentation des objets qui ont été définis, le plus pertinent : par valeurs, dans le cas sans alias, ou avec représentation du tas, si des alias sont détectés.

Xavier Urbain s'est consacré aux études de cas proposées par les partenaires industriels du projet VERIFICARD. Il est en particulier parti en séjour au sein de l'équipe LOOP de l'université de Nijmegen (pays-bas) afin de comparer les possibilités respectivement offertes par l'outil LOOP développé dans cette équipe et KRAKATOA.

6.1.7. Le théorème des 4 couleurs

Participants : Vincent Danos [PPS], Georges Gonthier [Projet Moscova], Benjamin Werner.

Le théorème dit des « 4 couleurs » est un résultat spéctaculaire de la théorie des graphes. Le résultat, conjecturé dès 1852, énonce que 4 couleurs sont suffisantes pour colorier une carte plane sans que deux pays voisins aient la même couleur. Cette conjecture a résisté plus d'un siècle aux efforts des mathématiciens jusqu'à ce qu'Appel et Haken proposent une preuve en 1976. Cette preuve a causé un émoi certain dans la communauté mathématique car elle comporte une part importante et apparemment irréductible de calcul ; plus précisément, cette partie est si importante qu'elle échappe aux capacités humaines et ne peut être vérifiée que par un ordinateur.

Parce que le système **Coq** combine harmonieusement déduction logique et calcul, il est un candidat naturel pour une formalisation complète de cette preuve. Georges Gonthier, Benjamin Werner et Vincent Danos ont poursuivi la formalisation commencée l'an dernier.

6.1.8. La logique à information partielle

Participant : Areski Nait Abdallah.

Mots clés : *méthode des tableaux de Beth*, *calcul des séquents*, *logique à information partielle*.

Areski Nait Abdallah a travaille sur les liens entre **Coq** et la logique en information partielle. Il a developpé une implémentation de la méthode des tableaux de Beth pour la logique en information partielle. Ce développement comporte notamment :

- un sous-système qui est une implémentation des tableaux de Beth (une sorte de calcul des séquents) pour le calcul propositionnel classique,
- une librairie d'algorithmes sur les arbres codés comme des ensemble de listes d'occurrences,
- des applications de la logique en information incomplète à la détection de pannes dans des circuits électriques.

Il s'agit de la première implémentation des tableaux de Beth en **Coq**, à la fois pour la logique classique, et pour la logique en information partielle.

6.1.9. Les développements dans d'autres systèmes

Participants : Frédéric Blanqui, Gilles Dowek.

Mots clés : *Isabelle*, *PVS*, *transfert*, *aéronautique*, *protocoles cryptographiques*, *sécurité*.

L'assistant à la démonstration Isabelle est développé par les équipes de Larry Paulson à Cambridge et de Tobias Nipkow à Münich, et repose sur une logique différente de celle utilisée dans **Coq**.

Frédéric Blanqui a formalisé un critère général pour s'assurer que certaines données utilisées dans un protocole cryptographique restent bien confidentielles. Cela a donné lieu au développement d'une bibliothèque de théorèmes dans Isabelle qui va être distribuée en mars 2003 avec la prochaine distribution d'Isabelle.

Notre projet entretient aussi une collaboration étroite avec le laboratoire ICASE-NASA Langley à Hampton en Virginie (États-Unis). Ces collaborations portent sur le développement de preuves en PVS (système alternatif à **Coq**) ainsi que sur le transfert vers PVS de tactiques développées en **Coq**.

Lors d'un séjour de deux mois dans le laboratoire ICASE-NASA Langley, Gilles Dowek, en collaboration avec César Muñoz a étudié des propriétés de passage à l'échelle pour algorithmes de résolutions de conflits aérien. Ils ont montré que l'algorithme KB3D, développé en collaboration avec Alfons Geser en 2001, utilisé par deux avions pouvait être rendu collaboratif si on lui imposait une certaine discipline dans le choix des solutions, et ce malgré l'abscence de concertation et de communication.

Ils ont ensuite montré une propriété de terminaison pour une restriction de cet algorithme utilisée avec un nombre arbitraire d'avions. Cette propriété montre que même si la résolution d'un conflit avec un avion peut en créer d'autres, tous les conflits finissent par être résolus après un temps fini.

6.2. La formalisation des mathématiques et le développement du système Coq

6.2.1. Le développement du logiciel Coq

Participants : Bruno Barras, Jean-Christophe Filliâtre, Hugo Herbelin, Pierre Letouzey.

Mots clés : *Coq*.

Le projet a distribué les versions V7.2, V7.3 et V7.3.1 de **Coq** en janvier, mai et octobre 2002 respectivement. Ces versions ont été l'occasion d'étendre et de raffiner diverses fonctionnalités déjà existantes, ainsi que d'introduire de nouveaux outils de développement et des tactiques.

Pierre Letouzey a modifié et amélioré l'outil d'extraction (voir section 6.2.9). Olivier Desmettre a enrichi la bibliothèque que des réels (voir section 6.1.3). Micaëla Mayero et David Delahaye ont développé une interface avec Maple. Hugo Herbelin a développé des tactiques permettant le « raisonnement vers l'avant », style couramment utilisé dans les preuves des textes mathématiques.

Bruno Barras s'est attaché à réparer certaines erreurs dans les algorithmes d'unification, qui sont un outil de base pour la création de tactiques automatiques. Ici, le but est de préparer le terrain en vue de l'implantation de l'algorithme d'unification sur lequel travaille Clément Renard.

Coq est désormais disponible par cvs anonyme à l'adresse http://coqcvs.inria.fr/. On peut notamment y télécharger la version de développement et la version corrigeant les bogues connus de la dernière version officiellement distribuée.

La documentation a été restructurée pour tenir compte de l'évolution du système [28].

Une nouvelle distribution majeure de **Coq** est en cours de préparation (version 8). Ses trois caractéristiques les plus importantes seront

- Un système de modules (voir section 6.2.4)
- Une nouvelle syntaxe concrète (voir section 6.2.5)
- L'intégration d'un compilateur vers du code à réduction efficace (voir section 6.2.2)

6.2.2. La compilation du test d'égalité

Participants : Bruno Barras, Benjamin Grégoire, Benjamin Werner.

Mots clés : *compilateur*, *machine abstraite*, *réduction*.

Benjamin Grégoire a continué de développer le compilateur pour **Coq** ainsi qu'un algorithme l'utilisant pour effectuer le test de conversion. Actuellement toutes les constructions de **Coq** sont compilables. Les résultats obtenus sur des exemples très calculatoires sont très encourageants, le gain est d'un facteur 50 sur certains exemples et la quantité de mémoire utilisée est réduite. La méthodologie employée a donné lieu à un article écrit avec Xavier Leroy dans ICFP.

Benjamin Grégoire a terminé la preuve d'un compilateur réaliste et d'un algorithme de normalisation basé sur la même technique que l'algorithme de conversion qu'il a implanté dans **Coq**. Le compilateur transforme les termes d'un λ-calcul avec `let` vers la machine abstraite de Ocaml étendue en une machine réduisant sous les lieurs. Les optimisations, comme la représentation minimale des environnements ou des appels terminals, ont pu être prouvées.

Benjamin Grégoire et Bruno Barras ont prouvé qu'il n'est pas nécessaire de convertir les annotations de types du Calcul des Constructions Inductives lors du test d'égalité : les systèmes avec et sans annotations de types sont équivalent.

Benjamin Werner s'est interessé au moyen d'adapter la théorie des types pour favoriser l'exécution de programmes à l'intérieur de celle-ci. En reprenant une proposition de Christine Paulin, il a défini une version de la théorie de **Coq** où l'identification entre les objets ne tient plus compte que de la partie de ces objets qui est effectivement calculatoirement significative. En d'autres termes, on applique un mécanisme d'extraction avant de faire le test d'égalité. Une telle théorie semble prometteuse.

Alexandre Miquel a collaboré avec Thierry **Coq**uand sur les propriétés de la fonction syntaxique d'incarnation (issue des travaux récents de J.-Y. Girard), qui permet notamment de définir un algorithme de test d'égalité typée dans plusieurs systèmes de types (avec ou sans types dépendants).

6.2.3. Le Calcul des Constructions Inductives et la réécriture

Participants : Frédéric Blanqui, Pierre Courtieu, Jean-Pierre Jouannaud, Daria Walukiewicz-Chrzaszcz.

Mots clés : *Calcul des Constructions Algébriques*, *réécriture*, *terminaison*, *structures non libres*, *quotients*.

Le *Calcul des Constructions Algébriques* a deux ambitions : étendre la règle de conversion à des règles de réécriture introduites par les utilisateurs, ce qui permet de définir des fonctions plus simplement et de définir des procédures de décision, et voir l'élimination des coupures de récurrence comme un cas particulier de règles utilisateur. Cela suppose que ces règles soient testées pour vérifier qu'elles ne compromettent pas la cohérence logique du système et la décidabilité du typage. La principale propriété requise pour cela est une propriété de terminaison.

Frédéric Blanqui a montré que le Calcul des Constructions Inductives à la base du système **Coq** peut être vu comme un Calcul des Constructions Algébriques particulier. Il a aussi étudié l'utilisation de récriture modulo certaines théories équationnelles comme l'associativité et la commutativité, très utiles en pratique.

Daria Walukiewicz-Chrzaszcz a travaillé sur certaines propriétés du calcul qu'elle avait formulé dans les années précédentes et qui consiste en les règles du typage du Calcul des Constructions et de la réécriture engendrée par les règles satisfaisant HORPO. En particulier, elle a détaillé la preuve de subject-reduction, elle a prouvé la décidabilité de HORPO et elle a corrigé la preuve de normalisation en renforçant l'ordre qui servait à faire l'induction dans un des lemmes.

Plus en amont, Jean-Pierre Jouannaud et Albert Rubio ont tout juste terminé et soumis au JACM un ambitieux travail intitulé « Higher-Order Recursive Path Orderings à la carte ». Cet article contient plusieurs aspects novateurs : un cadre très général pour la récriture d'ordre supérieur et pour la récriture normalisée d'ordre supérieur (de Tobias Nipkow), les algèbres d'ordre supérieur polymorphes ; deux définitions, manquantes y compris dans les cadres plus simples existant à ce jour, d'ordre de réduction d'ordre supérieur polymorphe, et d'ordre de réduction normalisé d'ordre supérieur polymorphe, accompagnées des résultats qui montrent leur intérêt pour faire des preuves de terminaison ; des généralisations à ce cadre de l'ordre récursif sur les chemins de N. Dershowitz permettant de résoudre automatiquement la terminaison de la plupart des exemples rencontrés en pratique dans les codages logiques utilisant la récriture d'ordre supérieur.

Jean-Pierre Jouannaud, Femke Van Ramsdoonk et Albert Rubio ont commencé un travail sur la récriture d'ordre supérieur « à la Nipkow », pour laquelle les règles sont déclenchées par filtrage d'ordre supérieur. Ce travail donne une nouvelle définition de cette récriture, qui s'appuie sur des signatures de symboles de fonctions et de variables *avec arités*, ce qui permet d'éviter les complications techniques liées aux η-expansions. Le cadre utilisé, celui des algèbres d'ordre supérieur polymorphes, fait apparaître un problème intéressant d'unification de « patterns (à la Miller) polymorphes » dans ce cadre.

Claude Marché et Xavier Urbain ont poursuivi un travail autour des preuves automatiques de terminaison. Un article a été soumis au *Journal of Symbolic Computation*.

6.2.4. Les modules

Participants : Jacek Chrzaszcz, Julien Signoles, Judicaël Courant, Jean-Pierre Jouannaud.

Mots clés : *PTS*, *modules*, *modularité*, *bibliothèques de preuves*, *théories mathématiques*, *défonctorisation*.

Jacek Chrzaszcz a travaillé sur l'implantation des modules dans le système **Coq**. Il a commencé par une extension du mécanisme de gestion des objets de haut niveau de **Coq** (comme règles de grammaires, paramètres par défaut, bases de données de tactiques automatiques, etc.) pour qu'il soit cohérent avec les modules et en particulier les modules paramétrés (foncteurs). L'implantation des modules a été intégrée dans la branche principale de l'archive **Coq** au mois d'août. La première version de la documentation utilisateur a aussi été réalisée.

Judicaël Courant a continué son travail sur les problèmes de réductions dans les systèmes de modules. L'étude de la preuve de normalisation de $\mathcal{MC}_2$ a des retombées sur les calculs avec types singletons : elle permet en effet de donner et de justifier pour ceux-ci un algorithme d'inférence et de vérification de type extrêmement simple [19].

Judicaël Courant a étudié l'interaction de la modularité avec le mécanisme d'univers implicites [18]. Cette étude a montré notamment que le mécanisme d'univers implicites était incompatible avec la vérification séparée des théories et propose des univers explicites polymorphes, compatibles avec la modularité, possédants

les propriétés métathéoriques requises (cohérence notamment). Cette proposition est implantée dans un système prototype appelé `oeuf`.

Durant son stage de DEA, Julien Signoles a travaillé sur un calcul statique des applications de modules paramétrés (appelé *défonctorisation*). L'étude théorique [27] a débouché sur une implantation d'un défonctoriseur pour Objective Caml. En thèse, il travaille sur une extension d'Objective Caml avec types dépendants.

6.2.5. L'interfaçage entre Coq et l'utilisateur

Participants : Bruno Barras, Olivier Desmettre, Hugo Herbelin.

Mots clés : *présentation des preuves*, *interopérabilité*, *syntaxe*.

Hugo Herbelin a isolé l'arbre de syntaxe abstraite des commandes, tactiques et termes de **Coq**, facilitant grandement la communication de **Coq** avec ses interfaces (notamment Pcoq développé par l'équipe Lemme et l'outil de rendu des preuves en langue naturelle via XML-MathML développé dans le cadre de l'action européenne MoWGLI).

Hugo Herbelin a réécrit l'interpréteur de syntaxe concrète de **Coq** et introduit un mécanisme d'extensibilité par des notations symboliques à la fois simple et puissant, directement accessible aux utilisateurs de **Coq**.

Bruno Barras a participé à l'effort visant à rendre **Coq** plus simple à utiliser. Sa principale activité a été la refonte complète de la syntaxe concrète. Les deux objectifs sont :

- D'une part faciliter l'interaction avec l'utilisateur humain. La principale difficulté aura été de proposer un système de notations plus cohérent, sans pour autant perdre la possibilité d'étendre dynamiquement cette syntaxe, de façon à adapter la notation concrète aux conventions mathématiques.

- D'autre part permettre le développement d'interfaces et d'outils de présentation des théories formalisées de façon indépendante de **Coq** (par exemple Pcoq et MoWGLI). Comme ces outils interagissent avec **Coq** par l'intermédiaire de la syntaxe concrète, une plus grande cohérence de notations rend ces outils plus robustes.

Bruno Barras a réalisé un prototype d'analyseur grammatical pour cette proposition. Olivier Desmettre a développé un traducteur pour la transition entre l'ancienne et la nouvelle syntaxe.

Hugo Herbelin a participé à l'intégration de l'outil d'export des définitions et preuves de **Coq** dans un format XML qui, après une série de traductions s'achevant dans le format MathML, peut être rendu en langue naturelle dans un navigateur Web tel que Mozilla.

Jean-Christophe Filliâtre a fait évoluer son outil COQWEB de « programmation littéraire » vers un nouvel outil de documentation de **Coq** : COQDOC. Cet outil permet d'annoter des fichiers sources **Coq** par des commentaires contenant des balises soit LaTeX, soit HTML, soit ASCII dans un format propre à COQDOC et de produire à partir de ceux-ci des documents PostScript ou hypertexte.

6.2.6. La Déduction Modulo

Participants : Gilles Dowek, Stéphane Vaillant, Olivier Hermant, Benjamin Werner.

Mots clés : *déduction modulo*, *réécriture*, *confluence*, *théorie des ensembles*, *arithmétique de Peano*.

La Déduction Modulo est une extension de la logique des prédicats obtenue en identifiant des propositions convertibles pour une certaine congruence.

Gilles Dowek et Benjamin Werner ont poursuivi l'étude des théories qui peuvent s'exprimer en déduction modulo uniquement avec des règles de calcul. Ils ont en particulier montré que c'était le cas de l'arithmétique de Peano. Cette formulation de l'arithmétique de Peano repose sur trois idées. D'une part celle d'exprimer l'injectivité de la fonction successeur (troisième axiome de Peano) en donnant une définition calculatoire de son inverse à gauche (la fonction prédécesseur). Ensuite, celle d'exprimer la non-confusion des constructeurs (quatrième axiome de Peano) en définissant de manière calculatoire l'image de ces constructeurs (le singleton contenant le nombre 0 et l'ensemble des entiers non nuls). Enfin, celle d'exprimer le principe de récurrence définissant de manière calculatoire le prédicat « être un entier » dans une forme faible de la logique du second ordre qui est elle-même exprimable de manière calculatoire dans la logique des prédicats.

Gilles Dowek et Benjamin Werner ont ensuite montré que la propriété d'élimination des coupures pour cette théorie était un corollaire de leur théorème général d'élimination des coupures pour la déduction modulo. Ce

résultat donne une tradution du système T de Gödel dans le lambda-calcul, et donc une expression des entiers dans le lambda-calcul qui permette un prédécesseur en temps constant. Cette expression des entiers est la même que celle proposée par M. Parigot dans son travail sur les types récursifs. Plus généralement, il semble qu'outre le type des entiers, les types récursifs et les types inductifs (comme ceux du système **Coq**) soient exprimables en déduction modulo. Ce résultat a été présenté au colloque Types et est soumis à publication [32].

Dans un travail indépendant, Gilles Dowek a montré qu'en logique proprositionnelle classique, n'importe quel ensemble d'axiome cohérent pouvait se transformer en une théorie formée exclusivement de règles de calcul, et que de plus cette théorie a la propriété d'élimination des coupures. Ce travail s'inspire d'idées dues à S. Negri et J. Von Plato (en particulier l'idée d'utiliser une forme conjonctive-disjonctive des axiomes), mais introduit aussi un certain nombre d'idées nouvelles, comme celle de l'utilisation de la propriété de cohérence (et donc de l'existence d'un modèle) pour fabriquer les règles de calcul. Ce travail a été présenté lors d'un exposé invité à Stacs 2002 [20].

Olivier Hermant a effectué son stage de DEA de mars a juillet. Il a construit une méthode permettant de traduire une dérivation en résolution modulo (méthode ENAR) en une preuve dans le calcul des séquents modulo qui n'utilise pas la règle de coupure.

Olivier Hermant a commencé sa thèse en octobre, et cherche une démonstration à base de modèles de l'élimination des coupures dans le calcul des séquents modulo. Quelques résultats ont déjà été obtenus dans le cas de la logique des prédicats. Ce travail est lié à son stage de DEA, et il met en œuvre des techniques qu'il y a développées.

6.2.7. Démonstration automatique

Participants : Bruno Barras, Pierre Corbineau, Judicaël Courant, Gilles Dowek, Clément Renard, Stéphane Vaillant.

Mots clés : *théorie des ensembles*, *unification d'ordre supérieur*.

Dans le cadre de sa recherche sur une formalisation de la théorie des ensembles en déduction modulo, Stéphane Vaillant a réalisé un logiciel de démonstration automatique.

Celui-ci est fondé sur une version de la méthode ENAR (calcul de résolution en déduction modulo) spécialisé à une présentation de la théorie des ensembles qu'il a précédemment donnée. Cette présentation peut être vue comme une implémentation au premier ordre d'une version de la théorie des ensembles exprimée avec un symbole lieur.

Dans un premier temps Stéphane Vaillant a réalisé un programme où l'application des règles d'inférence est effectué par l'utilisateur, mais où la recherche de la satisfiabilité des contraintes (unification modulo un calcul de substitution explicite) est effectuée automatiquement. De cette façon il a construit une preuve du théorème de Cantor.

Il a ensuite réalisé un programme où les règles d'inférences sont appliquées automatiquement. Ce programme permet de démontrer des lemmes simples de théorie des ensembles.

Clément Renard s'est documenté sur l'unification d'ordre supérieur dans le lambda-calcul simplement typé et a commencé à étudier les problèmes posés par les types dépendants qui conduisent à considérer des équations dont les deux membres peuvent ne pas avoir le même type et donc à introduire un ordre sur les équations. Il s'est également intéressé à une restriction décidable de l'unification d'ordre supérieur : les patterns d'ordre supérieur introduit par D. Miller et a commencé l'écriture d'un prototype d'unification pour cette restriction ce qui a permis de soulever de nouvelles difficultés liées au polymorphisme et de tester certaines stratégies.

Clément Renard a également quelque peu simplifié les structures utilisées pour l'unification dans **Coq**. Il a commencé à s'intéresser à l'inférence de type à la ML afin d'essayer de comprendre comment adapter ces techniques dans le cadre d'un calcul avec polymorphisme et types dépendants comme le Calcul des Constructions Inductives. Il a commencé à formaliser des règles d'unification qui devraient permettre, à terme, d'avoir une inférence de type partielle pour **Coq**.

6.2.8. Logique avec des variables liées

Participants : Gilles Dowek, Thérèse Hardin [LIP6 et Projet Moscova], Claude Kirchner [Projet Protheo], Ying Jiang.

Mots clés : *variable liée*, *complétude*, *logique minimale*, *décidabilité*.

Gilles Dowek, Thérèse Hardin et Claude Kirchner ont développé leur travail sur les modèles de la logique avec des symboles lieurs, ils ont en particulier proposé un contre-modèle pour l'axiome d'extensionalité. Ce travail a été publié [21].

En collaboration avec Ying Jiang, Gilles Dowek a proposé une nouvelle preuve de décidabilité du fragment positif de la logique minimale intuitionniste, résultalt démontré en 1968 par G. Mints. Cette nouvelle démonstration utilise un mécanisme de gestion de la portée des variables alternatif à la condition de fraîcheur des variables du calcul des séquents. La portée de ce nouveau mécanisme en démonstration automatique reste à explorer.

Gilles Dowek et Ying Jiang ont également remarqué que ce résultat s'étendait à toutes les théories modulo qui s'expriment avec des règles de calcul uniquement comme la théorie des types simples (logique d'ordre supérieur) ou le système F. Cela donne donc une nouvelle démonstration de décidablité de l'habitation d'un type positif dans le système F. Ce travail est soumis a publication [31].

6.2.9. L'extraction de programmes

Participants : Jean-Christophe Filliâtre, Pierre Letouzey, Nicolas Oury, Christine Paulin.

Mots clés : *extraction de programmes*, *preuves constructives*, *preuves de programmes*, *Haskell*, *Scheme*, *Ocaml*.

Pierre Letouzey a poursuivi l'étude et implantation du mécanisme d'*extraction*, qui permet à partir de preuves **Coq**, d'obtenir automatiquement du code source Caml dont le comportement calculatoire est celui des preuves initiales. Trois axes ont été développés :

- une partie théorique : il a fallu bâtir un système théorique d'extraction nouveau pour régler certaines difficultés. Une preuve de correction de ce système théorique a été obtenue.
- une partie implantation : l'outil d'extraction implémenté l'année précédente dans le système **Coq** a été modifié et amélioré pour correspondre à la partie théorique. Cet outil a aussi été complété pour produire du code source vers d'autres langages, à savoir Haskell et Scheme. Cette implantation est désormais distribuée dans la version
  7.3 de l'assistant **Coq**, et documentée.
- une partie expérimentation : grâce à la base actuelle des contributions des utilisateurs de **Coq**, Pierre Letouzey a pu tester l'extraction sur de multiples développements **Coq**.

Pierre Letouzey a décris ces trois axes plus en détail dans l'article [34] qui vient d'être soumis pour publication aux actes de la conférence TYPES'2002. Concernant l'extraction, les résultats obtenus correspondent quasiment aux objectifs fixés : la partie théorique est désormais bien déblayée et l'implantation est stabilisée. En particulier Pierre Letouzey a dernièrement levé la principale limitation de l'extraction, qui était la possibilité dans certains cas extrêmes de produire du code non typable. Cette nouveauté sera disponible dans la prochaine version de **Coq**. Mais tout ceci a pris beaucoup plus de temps que prévu initialement, repoussant d'autant l'étude du mécanisme réciproque d'importation de code Caml dans **Coq** afin d'en prouver les propriétés. Cette partie, initialement prévue pour être le cœur de la thèse de Pierre Letouzey, sera donc l'objet de l'année 2003.

Durant son stage de DEA, Nicolas Oury a travaillé sur la possibilité d'intégrer des tableaux persistants — i.e. à comportement fonctionnel — à l'extraction de **Coq**. Cela a amené à définir un schéma général de substitution de code à l'extraction [26]. En thèse, il travaille sur la puissance respective de la convertibilité et des axiomes d'égalité en **Coq**.

6.2.10. Modèles du calcul

Participant : Pierre-Louis Curien.

Mots clés : *sémantique dénotationnelle*.

Pierre-Louis Curien a montré [30] un lien nouveau entre modèles de jeux et sémantique dénotationnelle « traditionnelle ». On savait déjà[39] qu'ajouter des points symbolisant les erreurs dans les domaines interprétant les programmes permet d'observer des différences souvent qualifiées d'intensionnelles, telles que des différences dans l'ordre d'exécution. En 2002, Laird a réussi à donner une sémantique dénotationnelle très simple basée sur de tels domaines avec erreurs, qui ne sont rien d'autre que des treillis satisfaisant quelques axiomes. Pierre-Louis Curien a montré que le modèle des algorithmes séquentiels[38] en forme une sous-catégorie pleine, c'est-à-dire que l'on retrouve bien les mêmes objets, que l'on peut donc appréhender soit comme des fonctions, soit plus concrètement comme des arbres rassemblant des traces d'exécution.

6.2.11. Le Calcul des Constructions et la théorie des ensembles

Participants : Alexandre Miquel, Benjamin Werner.

Mots clés : *Calcul des Constructions*, *théorie des ensembles*, *axiome du choix*.

Alexandre Miquel a poursuivi son travail de recherche sur les liens entre théorie des types et théorie des ensembles, afin de dégager le contenu calculatoire de cette dernière, notamment de l'axiome du choix et du remplacement.

On considère en général que la manière la plus simple de modéliser une théorie des types est de construire un modèle ensembliste : chaque type est interprété par un ensemble et chaque habitant du type par un élément de cet ensemble.

Bien qu'utile pour établir certaines proprietés importantes, cette construction est souvent considérée comme peu interessante. Néanmoins, Alexandre Miquel a découvert que cette construction était plus problématique que ce qui était admis. Avec Benjamin Werner, ils ont exposé la difficulté et proposé une manière de la contourner.

7. Contrats industriels

7.1. Calife

Nous partcipons au projet exploratoire CALIFE (Environnement pour la Preuve formelle et le Test d'Algorithmes utilisés en Télécommunication) (référence http://www.loria.fr/calife) qui a été labélisé dans le cadre de l'appel d'offre RNRT de la fin 98 et se termine en janvier 2003.

Le but de ce projet était le prototypage d'un environnement permettant de valider, de facon rigoureuse, les phases *hautes* du cycle de développement des composants critiques. Les partenaires de ce projet sont CRIL, France Telecom R & D, l'INRIA-Rocquencourt, le LaBRI (Bordeaux), le LORIA, le LRI (Orsay) et le LSV (ENS Cachan). Le projet LogiCal est impliqué principalement dans deux actions : la première vise à améliorer l'automatisation des démonstrations par l'utilisation de techniques de démonstration réflexive, la seconde vise à mettre en place un prototype de système de traitement de démonstrations mathématiques reposant sur un calcul modulaire et modulo.

Parmi les résultats de ce projet on peut noter :

- Le développement d'un formalisme spécifique d'automates temporisés paramétrés appelés les p-automates. Les bibliothèques **Coq** modélisant cette classe d'automates et leurs propriétés ont été réalisées. Elles ont été utilisées pour modéliser les protocoles internet PIM et PGM de diffusion multicast. la version la plus récente du modèle utilise le nouveau mécanisme de modules intégré à Coq.

- La construction d'une plateforme permettant une édition graphique d'automates temporisés. Ces automates peuvent alors être automatiquement traduits vers les formats d'entrée de model-checkers (Kronos, Hytech, ...) ou d'assistants de démonstration (**Coq**, Isabelle). Cet outil a été expérimenté avec succès pour faire des preuves automatiques ou interactives d'une forme simplifiée du protocole CSMA CD de détection de conflit dans le protocole Ethernet.
- La réalisation de tactiques **Coq** effectuant des réécritures en utilisant le moteur de l'outil Elan développé dans le projet Protheo au LORIA.

7.2. Averroes

Nous participons au projet AVERROES qui a démarré en octobre 2002. Labelisé dans le cadre du Réseau National des Technologies Logicielles (RNTL), il fait suite au projet CALIFE et réunit les mêmes partenaires. Ce projet a pour objectif le développement de méthodes formelles capables de vérifier de manière fiable des propriétés apparaissant dans des problématiques industrielles. Il étend le cadre du projet CALIFE en s'intéressant non seulement aux propriétés fonctionnelles mais également à des propriétés stochastiques ou à la consommation de ressources des protocoles.

7.3. France-Telecom

Nous avons une collaboration suivie avec Pierre Crégut et Jean-François Monin de France Télécom à Lannion. Un contrat, venant en complément du contrat RNRT Calife pour le financement de thèses et post-doc et portant sur le passage à l'échelle des techniques de démonstration a été établi avec le LRI.

7.4. Verificard

Le projet européen Verificard étudie des questions de sécurité et de sûreté pour la nouvelle génération de cartes à puces.

Il regroupe l'INRIA, l'Université de Nijmegen, l'Université de Munich, l'Université de Hagen, le Swedish Institute of Computer Science et les entreprises Gemplus et Schlumberger-Sema.

8. Actions régionales, nationales et internationales

8.1. Actions européennes

8.1.1. Working Group TYPES

Le *Working Group* « TYPES » porte sur le développement assisté par ordinateur de démonstrations et de programmes.

Il regroupe des équipes de Helsinki, Chambéry, Paris, Lyon, Rocquencourt, Sophia Antipolis, Orsay, Darmstadt, Freiburg, München, Birmingham, Cambridge, Durham, Edinburgh, Manchester, London, Sheffield, Padova, Torino, Udine, Nijmegen, Utrecht, Bialystok, Warsaw, Minho, Chalmers, ainsi que les entreprises Prover Technology, France Telecom, Nokia, Dassault-Aviation, Trusted Logic et Xerox.

8.1.2. Pologne

Nous avons une collaboration avec l'université de Varsovie qui se traduit par des thèses en co-tutelle (Jacek Chrzaszcz et Daria Walukiewicz-Chrzaszcz).

8.1.3. Le Consortium MoWGLI

Le *Consortium* « MoWGLI » (Mathematics on the Web, Get it by Logic and Interface) porte sur le développement d'une bibliothèque hypertexte de théories mathématiques, basée autour d'une notation de document et de formule mathématique au format XML (OmDoc et MathML), ainsi que la conception d'outil d'analyse de recherche, et d'interfaces de manipulation des théories.

Il regroupe des équipes de Berlin, Bologne, Nimègue, Saarbrücken, Sophia-Antipolis, ainsi que l'entreprise Trusted Logic.

9. Diffusion des résultats

9.1. Animation de la communauté scientifique

9.1.1. Responsabilités éditoriales

Gilles Dowek a été membre du comité de programme de la *Conférence on Automated Deduction* (CADE). Gilles Dowek et Christine Paulin ont été membres du comité de programme de la conférence *Theorem proving in higher-order logics* (TPHOLs). Claude Marché a été membre du comité de programme des 14èmes Journées Francophones des Langages Applicatifs. Christine Paulin a été membre du comité de programme du Workshop on Types in Language Design and Implementation. Christine Paulin a été membre du comité de programme de la conférence *Rewriting Techniques and Applications* (RTA).

Jean-Christophe Filliâtre a organisé les prochaines Journées Francophones des Langages Applicatifs (Chamrousse, janvier 2003). Hugo Herbelin a co-organisé l'école d'été « Proofs-as-programs » à Eugene, Oregon, États-Unis. Jean-Pierre Jouannaud a organisé la journée annuelle du LIX (septembre 2002 : « la sécurité des logiciels »).

9.1.2. Jurys

C. Paulin a participé au jury de la thèse de Patrick Loiseleur et de la la thèse de Cuihtlauac Alvarado. C. Paulin a été rapporteur de la thèse de Antonia Balaa. Gilles Dowek a été rapporteur de la thèse de Nguyen Quang Huy.

9.1.3. Visites

Gilles Dowek a passé deux mois en résidence au laboratoire ICASE-NASA Langley en Virginie (États-Unis) Benjamin Werner a rendu visite à Martén Abadi à l'Université de Californie à Santa Cruz. Alexandre Miquel est parti en post-doc à l'Université technologique de Chalmers (Göteborg, Suède). Frédéric Blanqui est parti en post-doc à l'Université de Cambridge. Frédéric Blanqui a rendu une visite d'une semaine à Ralph Matthes et Helmut Schwichtenberg à Munich.

9.1.4. Colloques

Areski Nait Abdallah, Pierre Letouzey, Clément Renard et Stéphane Vaillant ont particité à l'école de Printemps de sémantique organisé par l'équipe PPS de Jussieu (Agay, mars 2002).

Areski Nait Areski a particité à ESSLLI (Trente, Août 2002),

Areski Nait Abdallah, Daria Walukiewicz-Chrzaszcz, Pierre Letouzey, Clément Renard, Jacek Chrzaszcz ont participé à l'école d'été Types 2002 « Theory and practice of formal proofs » à Giens, France en septembre 2002.

Frédéric Blanqui a participé au Workshop « Termination and Type Theory » à Göteborg, Suède, en novembre 2002.

Bruno Barras, Judicaël Courant, Olivier Desmettre, Gilles Dowek, Jean Duprat, Jean-Christophe Filliâtre, Hugo Herbelin, Olivier Hermant, Alexandre Miquel, Areski Nait Abdallah, Pierre Letouzey, Christine Paulin, Clément Renard et Benjamin Werner ont participé à la rencontre annuelle TYPES 2002 à Nimègue aux Pays-Bas. Judicaël Courant, Gilles Dowek et Benjamin Werner, Jean Duprat, Hugo Herbelin, Pierre Letouzey, ainsi que Alexandre Miquel y ont présenté un exposé.

Judicaël Courant a participé à la conférence TPHOLs où il a présenté une communication.

Judicaël Courant a participé à la conférence CADE.

Judicaël Courant a présenté son travail sur les singletons dans le cadre du workshop ITRS de FLOC 2002 à Copenhague.

Gilles Dowek a participé à LPAR 2002.

Gilles Dowek a participé à STACS 2002 où il a fait un exposé invité.

Gilles Dowek a participé au Meeting de l'ASL 2002 où il a fait un exposé invité.

Jean Duprat a participé au Workshop Cracovie-Chambéry-Lyon, à Cracovie (Pologne) du 26 au 29 juin 2002 dont l'objectif était de mettre en place une collaboration scientifique sur la logique et ses applications entre ces trois sites universitaires.

Hugo Herbelin et Benjamin Werner ont participé à la réunion du projet MoWGLI à Berlin en mars 2002. Hugo Herbelin a participé aux réunions du projet MoWGLI qui se sont déroulées à Eindhoven en juillet 2002 et à Saarbrücken en décembre 2002.

Pierre-Louis Curien, Hugo Herbelin, Alexandre Miquel et Benjamin Werner ont chacun donné un cours à l'école d'été « Proofs-as-programs » de Eugene, Oregon, États-Unis.

Benjamin Werner a donné un cours à l'école d'été TYPES « Theory and Practice of Formal Proofs » à Giens en septembre 2002. Jean Duprat a participé à l'organisation de cette école.

Jean-Pierre Jouannaud a participé à la conférence ICALP (Malagua, juillet 2002), et aux symposiums LICS (Copenhague, juillet 2002), et ISS (Tokyo, novembre 2002).

Pierre Letouzey a participé aux Journées Francophones des Langages Applicatifs (Anglet, janvier 2002).

Claude Marché et Xavier Urbain ont participé au workshop VeriSafe à Nice en septembre 2002, workshop conjoint des projets européens SecSafe et VerifiCard.

Xavier Urbain a participé à la réunion annuelle Verificard au CIRM du 7 au 9 janvier 2002.

Bruno Barras et Jean-Christophe Filliâtre ont présenté un exposé au séminaire QSL à l'INRIA Lorraine en mai 2002.

Gilles Dowek a fait un exposé « Qu'est-ce qu'une démonstration constructive ? » au colloque « Algorithmique et programmation » à Luminy.

Alexandre Miquel a participé au Workshop « Logic & Interactions » qui s'est tenu en février à l'Institut de Mathématiques de Luminy, et y a présenté un exposé.

9.1.5. Responsabilités diverses

Bruno Barras est consultant en méthodes formelles auprès de la société Trusted Logic.

Frédéric Blanqui est membre du jury du prix de thèse SPECIF.

Jean-Pierre Jouannaud est directeur du Laboratoire d'Informatique de l'École polytechnique.

Christine Paulin est membre du *steering committee* de l'*European Association for Computer Science Logic*.

Jean-Pierre Jouannaud est l'un des trois membres du jury du prix décerné annuellement par l'*European Association for Theoretical Computer Science* pour « life time achievements ».

Gilles Dowek est *CADE trustee*.

Jean-Pierre Jouannaud est président de l'*Association Française d'Informatique Théorique*. Gilles Dowek est président du *Comité des thèses* de cette association.

Olivier Desmettre est responsable des serveurs de diffusion mail et http de l'équipe : http://coq.inria.fr/ et http://logical.inria.fr/.

9.1.6. Distinctions

Jean-Pierre Jouannaud a reçu le prix de la Coopération Scientifique France-Chine 2002.

9.2. Enseignement

C. Paulin a encadré la thèse de Patrick Loiseleur « spécifications et preuves de programmes distribués : une approche basée sur les réseaux d'objets CORBA » qui a été soutenue en septembre 2002. Christine Paulin et Pierre Crégut ont encadré la thèse de Cuihtlauac Alvarado « Réflexion pour la réécriture dans le Calcul des Constructions Inductives » qui a été soutenue en décembre.

Bruno Barras encadre la thèse de Clément Renard. Judicaël Courant encadre les thèses de Pierre Corbineau et Julien Signoles. Jean-Christophe Filliâtre encadre la thèse de Nicolas Oury. Hugo Herbelin encadre la thèse de Julien Narboux. Jean-Pierre Jouannaud encadre les theses de Jacek Chrzaszcz, de Daria Walukiewicz-Chrzaszcz et d'Antoine Kremer. Christine Paulin encadre la thèse de Pierre Letouzey. Christine Paulin et

Olivier Ly de Schlumberger encadrent la thèse de June Andronick. Benjamin Werner encadre la thèse de Benjamin Grégoire.

Bruno Barras encadre le travail d'Olivier Desmettre. Judicaël Courant a encadré le stage de DEA de Julien Signoles. Gilles Dowek encadre la thèse de Stéphane Vaillant. Jean-Christophe Filliâtre a encadré les stages de DEA de Nicolas Oury et de Jérôme Creci. Gilles Dowek a encadré le stage de Olivier Hermant. Alexandre Miquel a encadré le stage de maîtrise de Germain Faure.

En tronc commun du DEA « Programmation, sémantique et preuves », Jean-Pierre Jouannaud a enseigné le cours « Les termes de premier ordre » et Benjamin Werner le cours « Preuves constructives ». Dans ce même DEA, Bruno Barras, Hugo Herbelin et Christine Paulin ont enseigné le cours d'option « Calcul des Constructions Inductive » et Claude Marché le cours d'option « Terminaison ».

Christine Paulin a organisé une formation **Coq** de trois jours en février 2002 au LRI qui a réuni une dizaine de personnes. Les cours ont été assurés par Bruno Barras, Jean-Christophe Filliâtre, Hugo Herbelin, Christine Paulin et Benjamin Werner.

Jean-Pierre Jouannaud, Christine Paulin et Judicaël Courant sont enseignants à l'Université de Paris XI. Christine Paulin est responsable de la Licence et de la Maîtrise d'Informatique. Elle enseigne en DEUG MIAS et maîtrise d'informatique. Jean-Pierre Jouannaud enseigne en licence d'informatique le cours d'informatique théorique, et en maîtrise d'informatique un TD de génie logiciel. Judicaël Courant a assuré le cours de logique en licence d'informatique.

Alexandre Miquel est ATER et Clément Renard moniteur à l'université d'Orsay. Clément Renard a donné des TP de compilation en maîtrise ainsi que des TD et TP de principes d'interprétation des langages en deuxième année de DEUG.

Gilles Dowek, Jean-Christophe Filliâtre et Jean-Pierre Jouannaud enseignent à l'École polytechnique. Gilles Dowek est chargé de cours. Jean-Pierre Jouannaud enseigne en majeure 2 de l'École polytechnique un cours de vérification de systèmes réactifs temps réel, qui peut également être pris comme TER théorique par les étudiants de maîtrise d'informatique de l'université Paris Sud.

Benjamin Werner assure un cours de **Coq** à l'ENSTA.

Gilles Dowek a publié deux livres de vulgarisation : *Peut-on croire les sondages ?* [11] et *Voulez-vous jouer avec les maths ?* [12].

10. Bibliographie

Bibliographie de référence

[1] B. BARRAS. *Auto-validation d'un système de preuves avec familles inductives.* Thèse de Doctorat, Université Paris 7, 1999.

[2] T. COQUAND, G. HUET. *The Calculus of Constructions.* in « Information and Computation », volume 76, 1988, pages 95-120.

[3] J. COURANT. *A Module Calculus for Pure Type Systems.* in « TLCA'97 », série LNCS, Springer-Verlag, pages 112 - 128, 1997.

[4] P. CURIEN, H. HERBELIN. *The duality of computation.* in « Proceedings of the International Conference of Functional Programming 2000 », série LNCS, volume 1210, Springer-Verlag, April, 2000.

[5] G. DOWEK. *La vérification automatique de démonstrations.* éditeurs R. JACQUART., in « Technique et Science Informatique : Informatiques, enjeux, tendance et évolutions », numéro 1-2-3, volume 19, Hermès, 2000, pages 195-202.

[6] G. DOWEK, T. HARDIN, C. KIRCHNER. *Theorem proving modulo.* rapport technique, numéro 3400, Institut National de Recherche en Informatique et en Automatique, 1998, http://www.inria.fr/rrrt/rr-3400.html.

[7] J.-C. FILLIÂTRE. *Preuve de programmes impératifs en théorie des types.* Thèse de Doctorat, Université Paris-Sud, July, 1999.

[8] C. PAULIN-MOHRING. *Inductive Definitions in the System Coq - Rules and Properties.* in « Proceedings of the conference Typed Lambda Calculi and Applications », série Lecture Notes in Computer Science, numéro 664, éditeurs M. BEZEM, J.-F. GROOTE., 1993, LIP research report 92-49.

[9] B. WERNER. *Une théorie des constructions inductives.* Thèse de Doctorat, Université Paris 7, 1994.

[10] THE COQ DEVELOPMENT TEAM. *The Coq Proof Assistant, Reference Manual.* http://coq.inria.fr/doc/main.html.

Livres et monographies

[11] G. DOWEK. *Peut-on croire les sondages ?.* Le Pommier, 2002.

[12] G. DOWEK. *Voulez-vous jouer avec les maths ?.* Le Pommier, 2002.

Articles et chapitres de livre

[13] F. BLANQUI, J.-P. JOUANNAUD, M. OKADA. *Inductive Data Type Systems.* in « Theoretical Computer Science », numéro 1-2, volume 272, 2002, pages 41-68.

[14] J.-C. FILLIÂTRE, F. POTTIER. *Producing All Ideals of a Forest, Functionally.* in « Journal of Functional Programming », February, 2002, http://www.lri.fr/~filliatr/ftp/publis/kr-fp.ps.gz, à paraître.

[15] C. MARCHÉ, C. PAULIN, X. URBAIN. *The Krakatoa tool for JML/Java Program Certification.* in « soumission JLAP », 2002.

[16] A. MIQUEL, B. WERNER. *The not so simple proof-irrelevant model of CC.* in « soumission Types Proceedings 2002 », 2002.

[17] D. WALUKIEWICZ-CHRZASZCZ. *Termination of rewriting in the Calculus of Constructions.* in « Journal of Functional Programming », 2002, à paraître.

Communications à des congrès, colloques, etc.

[18] J. COURANT. *Explicit Universes for the Calculus of Constructions.* in « Theorem Proving in Higher Order Logics : 15th International Conference, TPHOLs 2002 », série Lecture Notes in Computer Science, volume 2410, Springer-Verlag, éditeurs V. A. CARREÑO, C. A. MUÑOZ, S. TAHAR., pages 115-130, Hampton, VA, USA, août, 2002, http://www.lri.fr/~jcourant/papers/02/ExplUniv.ps.gz.

[19] J. COURANT. *Strong Normalization with Singleton Types.* in « Second Workshop on Intersection Types and Related Systems », série Electronic Notes in Computer Science, volume 70, Elsevier Science BV, éditeurs S. V. BAKEL., Copenhaguen, Danemark, juillet, 2002, http://www.lri.fr/~jcourant/papers/02/singlrr.ps.gz.

[20] G. DOWEK. *What is a theory ?.* in « Symposium on Theoretical Aspects of Computer Science », volume 2285, Springer-Verlag, éditeurs H. ALT, A. FERREIRA., pages 50-64, 2002.

[21] G. DOWEK, T. HARDIN, C. KIRCHNER. *Binding logic : proofs and models.* in « Logic for Programming, Artificial Intelligence, and Reasoning », série Lecture Notes in Computer Science, volume 2514, Springer-Verlag, éditeurs M. BAAZ, A. VORONKOV., pages 130-144, 2002.

[22] J.-C. FILLIÂTRE. *La supériorité de l'ordre supérieur.* in « Journées Francophones des Langages Applicatifs », pages 15-26, Anglet, France, Janvier, 2002, http://www.lri.fr/~filliatr/ftp/publis/sos.ps.gz.

[23] B. GRÉGOIRE, X. LEROY. *A compiled implementation of strong reduction.* in « International Conference on Functional Programming 2002 », ACM Press, pages 235-246, 2002.

[24] J.-P. JOUANNAUD, R. TREINEN. *Constraints and Constraint Solving : An Introduction.* in « Constraints in Computational Logics : Theory and Applications », série Lecture Notes in Computer Science, volume 2002, Springer-Verlag, éditeurs H. COMON, C. MARCHÉ, R. TREINEN., pages 1-46, 2002.

Rapports de recherche et publications internes

[25] O. HERMANT. *Déduction modulo et élimination des coupures : une approche syntaxique.* Mémoire deDEA, Politecnico di Milano, juillet, 2002.

[26] N. OURY. *Équivalence vis-à-vis de l'observation et extraction de programmes.* Rapport de DEA, Septembre, 2002.

[27] J. SIGNOLES. *Calcul statique des applications de foncteurs en présence d'effets de bord.* Rapport de DEA, septembre, 2002.

[28] THE COQ DEVELOPMENT TEAM. *The Coq Proof Assistant Reference Manual V7.2.* rapport technique, numéro 255, INRIA, France, mars, 2002, http://www.inria.fr/rrrt/rt-0255.html.

Divers

[29] F. BLANQUI. *Rewriting modulo in Deduction modulo.* 2002, Submitted.

[30] P. CURIEN. *Sequential algorithms as bistable maps.* 2002, manuscript.

[31] G. DOWEK, Y. JIANG. *Eigenvariables, bracketing and the decidability of positive minimal intuitionistic logic.* 2002, manuscript.

[32] G. DOWEK, B. WERNER. *Peano's arithmetic as a theory modulo.* 2002, manuscript.

[33] J.-C. FILLIÂTRE. *The Why verification tool.* 2002, http://why.lri.fr/.

[34] P. LETOUZEY. *A New Extraction for Coq.* 2002, http://www.lri.fr/~letouzey/download/NewExtraction.ps.gz, soumis aux actes de la conférence TYPES'02.

[35] C. MARCHÉ, C. PAULIN, X. URBAIN. *The Krakatoa Tool for JML/Java Program Certification.* 2002, manuscript.

[36] C. MARCHÉ, X. URBAIN. *Modular & Incremental Proofs of AC-Termination.* 2002, manuscript.

[37] B. WERNER. *A Proof-Irrelevant Type Theory.* 2002, manuscript.

Bibliographie générale

[38] G. BERRY, P.-L. CURIEN. *Sequential algorithms on concrete data structures.* in « Theoretical Computer Science », numéro 20, 1982, pages 265-321.

[39] R. CARTWRIGHT, P.-L. CURIEN, M. FELLEISEN. *Fully abstract models of observably sequential languages.* in « Information and Computation », numéro 111, volume 2, 1994, pages 297-401.