INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet LOGICAL

Logique et Calcul

Rocquencourt

THÈME 2A

Table des matières

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.

1 Composition de l'équipe

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 Cristal
Nelly Maloisel [TR INRIA]
Personnel INRIA
Bruno Barras [CR]
Hugo Herbelin [CR]
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]
Jean Goubault-Larrecq [Professeur à l'ENS Cachan]
Gérard Huet [DR INRIA, Projet Cristal]
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
Guan-Shieng Huang
Post-doctorants
Yijia Chen
Xavier Urbain
Doctorants
Frédéric Blanqui [Allocataire MENRT]
Jacek Chrząszcz [Thèse en co-tutelle LRI / Université de Varsovie]
Pierre Corbineau [Élève à l'École Normale Supérieure]
Pierre Courtieu [ATER]
David Delahaye [ATER]
Benjamin Grégoire [Allocataire MENRT, en commun avec le projet Cristal]
Pierre Letouzey [Élève à l'École Normale Supérieure]
Patrick Loiseleur

Micaela Mayero [ATER]
Alexandre Miquel [Allocataire MENRT]
Clément Renard [Élève à l'École Normale Supérieure de Lyon]
Stéphane Vaillant [Allocataire MENRT]
Daria Walukiewicz-Chrząszcz [Thèse en co-tutelle LRI / Université de Varsovie]

Stagiaire

Ashin Thariyan

2 Présentation et objectifs généraux

Le but des recherches menées dans le projet est de construire des *systèmes de traitement de démonstrations mathématiques*, c'est-à-dire des systèmes capables d'opérer des traitements divers sur des connaissances 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 transmettre sur les réseaux, ...

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 télécomunications, 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 mathématique « 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. 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 l'assistant 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 l'objet.

3.3 Les environnements interactifs

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 interactivement, à l'aide de tactiques. Une tactique est un programme, construit par l'utilisateur du système, qui transforme une proposition à démontrer, en un ensemble de nouvelles propositions suffisantes.

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 La liaison des variables

Mots clés : variables liées, indices de De Bruijn, calcul des substitutions explicites.

Le Calcul des Constructions Inductives qui est un λ-calcul comporte un certain nombre de symboles permettant de lier des variables dans les termes. Cette notion de variable liée peut s'étudier dans un cadre plus large que le Calcul des Constructions Inductives. Par exemple, la théorie de la réécriture en présence de variables liées est déjà très développée. Du point de vue logique, on peut également étendre la logique des prédicats avec des symboles permettant de lier des variables dans des termes.

Les questions qui se posent à propos de ces langages vont des formulations de la théorie des ensembles avec un nombre fini d'axiomes à la reformulation du théorème de complétude dans ce cadre. Les outils pour étudier les propriétés de ces langages sont la notion d'indice de De Bruijn et celle de substitution explicite.

3.6 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 $\forall x \exists y\ Q(x, y)$) 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'applications

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 la 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

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

Résumé : *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 élémentaire (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 et d'une version certifiée de l'algorithme de Buchberger utilisé en Calcul Formel.

– la démonstration du théorème de d'Alembert-Gauss.

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.

6 Résultats nouveaux

6.1 Les développements de démonstrations formelles et de tactiques

6.1.1 Les automates temporisés

Participants : Christine Paulin, Yjia Chen, Matthieu Objois, Julien Signoles, Ashin Thariyan.

Mots clés : automates temporisés, protocoles de télécommunications.

Le projet RNRT CALIFE a pour objet la spécification, la preuve et les tests de protocoles de communication modélisés à l'aide d'automates temporisés : les p-automates. Une bibliothèque permettant de représenter des p-automates en Coq a été développée et est intégrée aux contributions de Coq [40].

Cette bibliothèque a été utilisée par Y. Chen, post-doc au LRI puis à France Telecom R & D et par K. Le Liboux dans son stage de DEA à France Telecom R & D pour modéliser des protocoles de multicast développés par France Telecom R & D. Elle a également été utilisée par M. Objois et J. Signoles dans un stage de maîtrise pour développer des versions généralisées d'un protocole de contrôle de conformité.

Christine Paulin a collaboré avec B. Tavernier de CRIL technology pour le développement un traducteur automatique des automates représentés sous une forme graphique vers une représentation Coq. La principale difficulté a été de trouver une représentation systématique adéquate pour les variables manipulées dans l'automate. Les types dépendants de Coq offrent une solution élégante à cette question.

Christine Paulin étudie l'automatisation de la génération d'invariants pour les p-automates dans le cas où les transitions correspondent à des propriétés arithmétiques simples.

6.1.2 L'analyse et la géométrie réelle

Participants : David Delahaye, Olivier Desmettre, Gilles Dowek, Jean Duprat, Micaela

Mayero, Benjamin Werner.

Mots clés : analyse, géométrie, nombre réel.

Micaela Mayero a enrichi la bibliothèque des nombres réels de Coq en y ajoutant des développements sur les suites et les séries.

Micaela Mayero a prouvé la correction de l'algorithme de différentiation du système Odyssée pour les programmes Fortran linéaires (*straight line*).

David Delahaye et Micaela Mayero ont généralisé la tactique Field, développée l'an passé pour les nombres réels, à tous les corps commutatifs.

Olivier Desmettre a développé en Coq un système de modélisation de trajectoires d'avions en se basant sur des éléments déjà développés en PVS. Ceci lui a permis de compléter la bibliothèque des réels de Coq et de proposer des modifications pour améliorer certaines preuves.

Jean Duprat a développé une axiomatisation de Hilbert de la géométrie plane. Contrairement à la géométrie analytique, celle-ci se définit sans utiliser les nombres réels. L'idée est d'utiliser la séparation entre Set et Prop dans Coq pour simuler la règle et le compas comme seuls outils de construction des figures.

6.1.3 La certification de programmes impératifs et à objets

Participants : Jean-Christophe Filliâtre, Xavier Urbain.

Mots clés : preuves de programmes, programmes impératifs, distance d'édition, lignes de Bresenham, langage à objets.

Jean-Christophe Filliâtre a certifié deux programmes impératifs à l'aide du système Coq. Le premier est un programme calculant la distance d'édition entre deux textes (i.e. le plus petit nombre d'insertions et de suppressions permettant de transformer le premier texte en le second). Le second programme implante l'algorithme de tracé de lignes de Bresenham, dont Jean-Christophe Filliâtre a montré l'optimalité (i.e. chaque point tracé est au plus proche de la droite rationnelle exacte). Ces exemples démontrent la robustesse des outils permettant de prouver la correction de programmes impératifs en Coq.

Xavier Urbain a réalisé un état de l'art sur la formalisation des langages à objets et les outils de preuves qui en sont tirés (Jive de l'université de Haagen, Bali de l'université de Münich ou encore Loop développé à Nijmegen). Une réflexion sur ces différentes approches devrait conduire à un prototype permettant un traitement dans Coq des programmes codés en un langages incluant strictement Javacard.

6.1.4 Les sétoïdes

Participants : Clément Renard, Benjamin Werner.

Mots clés : setoïdes, extensionnalité, réécriture.

Clément Renard a réalisé une tactique pour les sétoïdes dans le cadre de son stage de DEA. Cette tactique permet d'utiliser la tactique Rewrite avec une autre relation d'équivalence que

l'égalité propositionnelle de la bibliothèque standard de Coq. Ce développement à été intégré dans les bibliothèques de Coq.

6.1.5 La théorie des graphes

Participant : Jean Duprat.

Mots clés : graphe.

Jean Duprat a développé une bibliothèque pour la théorie des graphes. Dans la poursuite des travaux effectués par Clément Renard, Jean Duprat a écrit une bibliothèque de définitions et théorèmes de base qui est enregistrée comme contribution Coq.

La théorie des graphes est intéressante car elle combine des aspects purement formels de type mathématiques avec des implémentations algorithmiques sans que la cohérence entre ces deux aspects ne soit généralement garantie. L'utilisation de Coq permet la réunion dans un même langage des démonstrations et des algorithmes, ce qui offre naturellement cette garantie. L'utilisation de types inductifs et de types dépendants renverse fréquemment l'ordre usuel entre les définitions et les propriétés mais elle fournit de puissants outils de démonstration.

6.1.6 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 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 entrepris une telle formalisation. Georges Gonthier a proposé une formalisation nouvelle et élégante des graphes planaires dont l'implémentation en Coq est en cours par Benjamin Werner. Georges Gonthier a prouvé entièrement la correction de la partie la plus calculatoire de la preuve (la réductibilité).

6.1.7 Les protocoles cryptographiques

Participant : Jean Goubault-Larrecq.

Mots clés : protocole, cryptographie, sécurité.

Jean Goubault-Larrecq a travaillé sur la vérification de protocoles cryptographiques, en particulier à l'aide de techniques d'automates d'arbres étendus, qu'il développe avec son étudiant

en thèse, Kumar Neeraj Verma. Le but est de pouvoir prouver la sécurité de protocoles utilisant des constructions complexes comme le ou exclusif, les exponentielles à la Diffie-Hellman ou à la RSA.

6.1.8 La réorganisation des bibliothèques

Participants : David Delahaye, Jean Duprat.

Mots clés : bibliothèque.

David Delahaye et Jean Duprat ont effectué une classification sémantique des contributions du système Coq. Une hiérarchie de pages HTML a pu être élaborée et est disponible sur le site de Coq (`http://coq.inria.fr/`).

6.1.9 Les développement dans d'autres systèmes

Participants : Gilles Dowek, Jean-Christophe Filliâtre, Micaela Mayero.

Mots clés : PVS, aéronautique, transfert, nombre réel, programme impératif.

Notre projet entretient une collaboration étroite avec le laboratoire ICASE-NASA Langley à Hampton en Virginie. 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.

Gilles Dowek, en collaboration avec Alfons Geser, Rick W. Butler, Victor Carreño et César Muñoz, a étudié les propriétés de sûreté de plusieurs algorithmes utilisés en aéronautique.

En collaboration avec Rick W. Butler, Victor Carreño et César Muñoz, il a proposé une nouvelle méthode, utilisant une formalisation de la géométrie réelle, pour formuler et démontrer la correction d'algorithmes géométriques. Ils ont appliqué cette méthode pour démontrer certaines propriétés de sûreté de l'algorithme *Airborne Lateral Spacing*, utilisé pour l'assistance des pilotes lors d'atterrissages parallèles [28].

En collaboration avec Alfons Geser et César Mũnoz il a proposé un nouvel algorithme de détection et résolution de conflits aériens [44]. Une preuve de sûreté de cet algorithme a ensuite été donnée.

Jean-Christophe Filliâtre a étendu la tactique permettant de prouver la correction de programmes impératifs qu'il avait développée dans sa thèse en en faisant un outil indépendant du système Coq, et en y ajoutant une génération d'obligations de preuve pour le système PVS. L'outil obtenu a été utilisé pour améliorer les preuves ci-dessus.

Micaela Mayero a porté en PVS la tactique qu'elle avait développée avec David Delahaye qui permet de démontrer automatiquement des égalités portant sur les nombres réels. La tactique obtenue a également été utilisé pour améliorer les preuves ci-dessus.

6.1.10 Algorithmes auto-stabilisants

Participant : Pierre Courtieu.

Mots clés : Auto-stabilisation, terminaison, réécriture.

Dans le cadre d'un projet BQR commun entre les équipes démonstration et parallélisme du L.R.I., Pierre Courtieu a spécifié la notion d'algorithme auto-stabilisant et prouvé une méthode particulière de preuve d'auto-stabilisation. La principale caractéristique d'un algorithme auto-stabilisant s'exécutant sur un réseau est qu'il converge vers un ensemble de configurations stables (dites légitimes) quelque soit la configuration initiale du réseau, ce qui assure des propriétés de tolérance au fautes à l'algorithme considéré. Des techniques de réécriture sur les mots sont employées.

6.2 La formalisation des mathématiques et le développement du système Coq

6.2.1 Modification de Coq

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

Mots clés : Coq.

En mars 2001, le projet a distribué la première version officielle de la nouvelle implantation Coq V7. La version V7.1 qui corrige certains problèmes a été distribuée en septembre.

La nouvelle implantation Coq V7 repose sur une restructuration globale de l'environnement de vérification de preuves qui est devenu plus petit et plus sûr. Elle prépare l'intégration de modules et foncteurs à laquelle travaille J. Chrzaąszcz. Elle intègre le langage de tactiques de haut niveau développé par D. Delahaye ainsi que la nouvelle fonction d'extraction de programmes ML développée par J.-C. Filliâtre et P. Letouzey. H. Herbelin a intégré un mécanisme de définition locale et l'accès par un nom qualifié aux objets définis dans des modules. De nouvelles tactiques ont été introduites et de nombreux développements réalisés dans le projet ou par nos partenaires ont été intégrés à l'ensemble des contributions maintenues par le projet.

La documentation a été restructurée pour tenir compte de l'évolution du système [1].

Suite à la restructuration globale de l'implantation de Coq en 2000, Bruno Barras s'est consacré particulièrement à simplifier et réduire la taille du noyau (partie du code de Coq assurant la cohérence logique de tous les raisonnements effectués) en y éliminant tous les concepts non primitifs. En particulier, les notions d'objets partiels (métavariables et variables existentielles) ont pu être sortis du noyau. Il en résulte un code d'une plus grande sûreté, plus proche de sa description formelle. Il s'agit donc d'un pas supplémentaire vers l'auto-certification (*bootstrap*) de Coq.

Hugo Herbelin a développé et implanté un nouveau modèle plus concis et plus performant pour la gestion des contraintes de stratification des univers logiques sur lesquels repose la logique de Coq.

Jean-Christophe Filliâtre a écrit un outil de « programmation littéraire » pour le système Coq : coqweb. Cet outil permet d'annoter des fichiers sources Coq par des commentaires conte-

nant du LaTeX et de produire à partir de ceux-ci des documents plus agréables à lire que des documents ASCII.

6.2.2 La compilation et les machines abstraites

Participants : Bruno Barras, Benjamin Grégoire, Hugo Herbelin, Pierre-Louis Curien, Benjamin Werner.

Mots clés : compilateur, machine abstraite, réduction.

Bruno Barras a amélioré la machine de réduction de la version courante de Coq, ce qui permet d'effectuer des preuves basées sur l'exécution d'algorithmes complexes, comme le théorème des quatre couleurs. Cette machine de réduction est utilisée pour le test de conversion servant à déterminer si deux expressions sont équivalentes. Bruno Barras a amélioré la stratégie visant à faire converger ces deux expressions en implantant une heuristique indiquant laquelle doit être simplifiée en priorité.

Benjamin Grégoire a réalisé un compilateur et une machine abstraite pour Coq. Le compilateur et la machine abstraite sont fortement inspirés de la technologie d'Ocaml. Le but étant de pouvoir réduire plus rapidement les termes de Coq.

Par ailleurs, Benjamin Grégoire a implanté un nouvel algorithme de mise en forme normale des termes Coq qui utilise le compilateur et la machine abstraite. Enfin, il a réalisé une preuve du compilateur et de la machine abstraite en Coq, pour un λ-calcul avec let.

Hugo Herbelin a donné une formalisation abstraite de la machine virtuelle d'Objective Caml. Le langage ayant servi à la formalisation est le $\overline{\lambda}$-calcul (un calcul inspiré du formalisme logique du calcul des séquents de Gentzen et pertinent pour la description fine de la réduction du λ-calcul).

Pierre-Louis Curien et Hugo Herbelin ont poursuivi leur travaux sur des sémantiques opérationnelles et machines abstraites basées sur le principe de l'interaction (entre programme et environnement). La problématique s'est enrichie cette année des idées de la ludique de Girard. Pierre-Louis Curien a pu reconnaître dans ces travaux une syntaxe intéressante et prometteuse : celle d'un λ-calcul non typé muni d'une notion d'enregistrement, directement issu d'une analyse polarisée de la logique linéaire. En raison des liens entre logique polarisée et recherche de preuves (mis en évidence par J.-M. Andreoli), ces travaux peuvent intéresser les problématiques de LogiCal à plus d'un titre.

6.2.3 Le Calcul des Constructions Inductives et la réécriture

Participants : Frédéric Blanqui, Jacek Chrząszcz, Pierre Courtieu, Jean Goubault-Larrecq, Jean-Pierre Jouannaud, Daria Walukiewicz-Chrząszcz.

Mots clés : Calcul des Constructions Algébriques, réécriture, terminaison structure non libre, 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 étudié les propriétés métathéoriques du Calcul des Constructions Algébriques. Une originalité du système qu'il a étudié est que les règles de réécriture s'appliquent non seulement à des termes, mais également directement à des types et à des propositions. Il a établi un ensemble de conditions très générales qui permettent d'assurer la décidabilité de la vérification de types dans ce cas, c'est-à-dire, savoir si une preuve est correcte ou non. Ce travail a reçu le prix Kleene 2001 et le prix Spécif 2001. Il a également étendu ces résultats à une classe relativement vaste de systèmes de types : les systèmes de types purs (PTS).

Daria Walukiewicz a proposé une autre méthode pour prouver la terminaison d'une extension du Calcul des Constructions avec des règles de réécritures fondées sur le *Higher-order recursive path ordering* (HORPO) de Jouannaud et Rubio. Cette méthode permet de prouver la terminaison de systèmes qui ne vérifient pas le critère précédent, mais son extension à la réécriture des types n'est pas encore achevée.

Plus en amont, Jean-Pierre Jouannaud et Albert Rubio ont poursuivi leur travail de construction de critères de terminaison pour les systèmes de réécriture d'ordre supérieur. Ils ont généralisé leur théorème sur les HORPO (Higher-Order Recursive Path Ordering) en introduisant un ordre sur les types qui permet de comparer par une version étendue du HORPO des termes de type comparable (dans la même direction). De plus, ils ont obtenu une version uniforme de cet ordre, en intégrant l'ordre sur les termes et l'ordre sur les types dans une même définition. Cette définition intégrée pourrait servir de base à un ordre, dont la normalisation forte est une conjecture, sur les termes d'un λ-calcul avec types dépendants, en particulier le Calcul des Constructions.

Jean Goubault-Larrecq a obtenu un théorème de terminaison relativement général, qui généralise notamment toutes les variantes connues de RPO, ainsi que la version de base du HORPO de Jouannaud et Rubio.

Pierre Courtieu s'intéresse à la représentation des structures non libres dans le formalisme sous-jacent du système Coq : le calcul des constructions inductives. Une structure non libre est une structure dans laquelle des termes commençant par des constructeurs différents peuvent être égaux. Ces structures, faciles à définir en théorie des ensembles, ne sont pas aisément exprimables dans le calcul des constructions inductives. Des réponses imparfaites ont été apporté à ce problème par un certain nombre de contributions. Pierre Courtieu en a fourni une nouvelle : les *types normalisés*, une extension du calcul des constructions inductives, dont l'implantation dans une version expérimentale de Coq est prévue. Les types normalisés ont fait l'objet d'une publication à CSL'2001.

Le problème posé par les structures non libre et en particulier par les quotients a été étudié dans le cadre des théorie intentionnelle notamment par Backhouse et al., Hofmann, Barthes, Geuvers et récemment par Samuel Boutin, qui est l'auteur du développement des *types quotients* dans Coq. Le type T/R ainsi obtenu n'est pas un type inductif, ce qui empêche la génération d'un principe d'élimination.

Les *type normalisés* sont une variante des types quotients, dans laquelle on associe au type non pas une relation, mais une fonction de normalisation *nf*. Un terme de ce nouveau type T/nf est un terme t de T associé à une preuve que $t = (nf\ t)$, c.a.d. que t est en forme

normale pour nf.

L'avantage de cette méthode réside dans le fait que chaque classe d'équivalence correspond à un unique terme clos du type normalisé T/nf (à la pertinence des preuves près). Bien entendu tous les quotients ne peuvent pas être représentés de cette manière, il faut que la relation du quotient soit exprimable sous la forme d'un calcul.

6.2.4 Les modules

Participants : Jacek Chrząszcz, Judicaël Courant, Jean-Pierre Jouannaud.

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

Jacek Chrząszcz a étendu le système Coq avec un système de modules inspiré par le travail de Judicaël Courant et Xavier Leroy. L'implantation permet d'introduire des modules d'une facon interactive, y compris des foncteurs. Cette fonctionalité permettra de développer une bibliothèque de théories paramétriques. L'étape suivante consistera en l'implantation des règles de réécriture modulaire.

Jacek Chrzaszcz a étendu le système Coq avec un système de modules inspiré par le travail de Judicaël Courant et Xavier Leroy. L'implantation permet d'introduire des modules d'une facon interactive, y compris des foncteurs. Cette fonctionalité permettra de développer une bibliothèque de théories paramétriques. L'étape suivante consistera en l'implantation des règles de réécriture modulaire.

Judicaël Courant a continué son travail sur les problèmes de réductions dans les systèmes de modules. La simplification du calcul qu'il avait donné dans sa thèse a donné naissance au calcul $\mathcal{MC}_2$, implanté dans un prototype appelé `oeuf`. Ce travail de simplification devrait permettre, à terme, de simplifier la preuve de normalisation ce qui devrait ouvrir la porte à la définition d'extensions de son calcul de modules, telle que la possibilité d'abréger des types de modules ou de définir des foncteurs surchargés (modules paramétrés se spécialisant en fonction de leur argument).

Dans cette perspective, Judicaël Courant a étudié l'interaction de la modularité avec le mécanisme d'univers implicites. Cette étude a montré notamment que le mécanisme d'univers implicites était incompatible avec la vérification séparée des théories. C'est pourquoi Judicaël Courant a préparé une proposition permettant d'avoir des univers explicites polymorphes, compatibles avec la modularité, possédants les propriétés métathéoriques requises (cohérence notamment). Il a commencé à l'implanter dans `oeuf`.

6.2.5 La Déduction Modulo

Participants : Gilles Dowek, Thérèse Hardin, Claude Kirchner, Stéphane Vaillant, Benjamin Werner.

Mots clés : déduction modulo, réécriture, confluence, théorie des ensembles.

Cette idée d'articuler règles de raisonnement et règles de calcul peut se formuler dans un cadre beaucoup plus général que le Calcul des Constructions Inductives. La Déduction Modulo

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

Stéphane Vaillant a donné une formulation de la théorie des ensembles en Déduction Modulo avec un nombre fini d'axiomes et de règles de calcul. Il a commencé la réalisation d'un logiciel de démonstration automatique qui exploite cette théorie, et qui peut fonctionner en mode automatique ou en mode interactif. Stéphane Vaillant a construit une démonstration du théorème de Cantor en mode interactif. Il étudie en ce moment les améliorations à apporter à son système pour que celui-ci trouve cette démonstration en mode automatique.

Gilles Dowek a comparé les algorithmes d'élimination des coupures en présence de règles de déduction non logiques et en présence de règles de calcul. Il a en particulier montré que la notion de coupure par pliage et dépliage proposée par D. Prawitz était un cas particulier de la notion de coupure en Déduction Modulo [22].

Gilles Dowek a donné une nouvelle présentation de la théorie des ensembles stratifiée de Quine en Déduction Modulo. La propriété d'élimination des coupures pour cette théorie a pu être démontrée comme une conséquence du théorème d'élimination des coupures pour la Déduction Modulo, ce qui simplifie la preuve originale de Crabbé en en conservant l'esprit [34]. Gilles Dowek et Benjamin Werner ont montré que l'arithmétique de Peano pouvait se formaliser en Déduction Modulo avec des règles de calcul exclusivement, et que la théorie ainsi obtenue avait une démonstration prédicative d'élimination des coupures [51].

Gilles Dowek a étudié des formalismes plus faibles que la Déduction Modulo, en particulier la Déduction Modulo asymétrique. Il a montré que la propriété de confluence pour un système de réécriture n'était autre que la propriété d'élimination des coupures pour la théorie définie par ce système de réécriture en Déduction Modulo asymétrique. Il a montré que le théorème de Newman était non seulement un théorème d'élimination des coupures, mais aussi de normalisation [33]. Il a également proposé un système encore plus faible, dans lequel les règles de réécriture s'appliquent exclusivement aux occurrences positives (ou négatives) d'une proposition atomique, et montré que toute théorie sans quantificateur pouvait se formuler dans ce système avec des règles de réécritures exclusivement [35].

L'article de G. Dowek, Th. Hardin et C. Kirchner consacré à la présentation de la logique d'ordre supérieur en Déduction Modulo est paru [21].

6.2.6 Le langage des preuves et des tactiques

Participants : David Delahaye, Benjamin Werner.

Mots clés : langage de preuve, langage de tactiques.

David Delahaye a effectué une étude comparée de trois styles de langages de preuves : les langages procéduraux, les langages déclaratifs et ceux utilisant des termes de preuves à la Curry-Howard. Il a mis en évidence la pertinence de chacun de ces styles dans certaines situations concrètes. Il a ensuite proposé un nouveau langage, appelé Lpdt réalisant la fusion de ces trois styles. Il a décrit une sémantique formelle (naturelle) de ce langage, dans le Calcul des Constructions Inductives sans univers, et il a implémenté ce langage dans le système Coq.

David Delahaye a mis en évidence la nécessité d'une nouvelle couche de métalangage dans les outils d'aide à la preuve à la LCF, notamment pour pouvoir traiter facilement les petites

automatisations ponctuelles. Dans le cadre de Coq en particulier, il a présenté Ltac, un nouveau langage de tactiques représentant une sorte de juste milieu entre le langage de Coq et son métalangage (Objective Caml). David Delahaye a alors mis en évidence la lisibilité, la compacité, la facilité de maintenance, ainsi que la portabilité des scripts Ltac, par rapport à leurs équivalents en Objective Caml. Comme un bonus, des automatisations non triviales ont pu être codées comme Tauto, une tactique de recherche de preuves pour la logique propositionnelle intuitionniste, ou Field, une tactique décidant des égalités sur les corps commutatifs. Par ailleurs, deux outils ont été développés pour Ltac : un système de quotations, permettant d'utiliser Ltac en Objective Caml, et un *debugger* spécifique pour le langage de tactiques.

6.2.7 Le Calcul des Constructions Implicites

Participants : Hugo Herbelin, Alexandre Miquel.

Mots clés : argument implicite.

Alexandre Miquel a introduit une variante allégée du Calcul des Constructions avec univers dans laquelle les fonctions sont dépourvues de domaine de définition, et dont le système de types permet de définir de nouveaux types de données par intersection (ce qui généralise la gestion implicite du polymorphisme telle qu'elle est effectuée dans les langages de la famille ML). Au cours de cette étude, Alexandre Miquel a démontré que le Calcul des Constructions Implicite vérifie de bonnes propriétés logiques et calculatoires, telles que la cohérence logique et la propriété selon laquelle tous les calculs que le formalisme permet d'exprimer terminent (propriété de normalisation forte).

6.2.8 La logique avec des variables liées

Participants : Gilles Dowek, Thérèse Hardin (LIP6 et Projet Moscova), Claude Kirchner (Projet Protheo).

Mots clés : variable liée, complétude.

Gilles Dowek, Thérèse Hardin et Claude Kirchner ont entrepris l'étude systématique d'une extension de la logique du premier ordre dans laquelle les termes comportent des variables liées. Ils ont proposé une notion de modèle pour cette logique et démontré un théorème de complétude [32].

6.2.9 L'extraction de programmes

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

Mots clés : extraction de programmes, preuves constructives, preuves de programmes.

Pierre Letouzey et Jean-Christophe Filliâtre ont réalisé une nouvelle implémentation du mécanisme d'extraction de Coq. Cette nouvelle implémentation est basée sur les principes proposés dans le mémoire de DEA de Pierre Letouzey. Parallèllement à l'implémentation, Pierre Letouzey a poursuivi ses travaux visant à établir la correction de cette méthode.

6.2.10 Le Calcul des Constructions et la théorie des ensembles

Participants : Hugo Herbelin, Alexandre Miquel.

Mots clés : Calcul des Constructions, théorie des ensembles.

Alexandre Miquel a étudié l'expressivité de certaines théories des types vis-à-vis de la théorie des ensembles. Une telle étude est importante sur le plan théorique, car elle permet de comprendre jusqu'où il est possible de transporter les mathématiques usuelles (i.e. formalisables en théorie des ensembles) dans les systèmes de traitement des démonstrations basés sur une notion de calcul primitive, tels que le système Coq. Au cours de cette étude, Alexandre Miquel a démontré que le noyau du système Coq privé de son mécanisme de définitions inductives contient une théorie des ensembles plus forte que celle de Zermelo, répondant ainsi à une conjecture posée par T. Coquand en 1986.

Afin de s'assurer de la correction de ces résultats, Alexandre Miquel en a effectué la formalisation dans le système Coq en s'appuyant également sur un outil indépendant de vérification de preuves (basé sur une restriction du formalisme de Coq) qu'il a développé pour cette occasion.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Calife

Un projet exploratoire CALIFE (Environnement pour la Preuve formelle et le Test d'Algorithmes utilisés en Télécommunication) a été labélisé dans le cadre de l'appel d'offre RNRT de la fin 98. Le but de ce projet est le prototypage d'un environnement permettant de valider, de facon rigoureuse, les phases *hautes* du cycle de développement des composants critiques. Cet environnement devra à la fois permettre la spécification d'un algorithme, sa vérification formelle, et la génération de tests permettant de valider une implémentation de cet algorithme. 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.

7.2 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.

7.3 Verificard

Le Projet 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 nationales

8.1.1 Action de recherche concertée SJava

Participants : Jean Goubault-Larrecq, Benjamin Grégoire, Benjamin Werner.

L'action SJava à laquelle participent les projets INRIA LogiCal, Lande, Lemme, Meije et Oasis ainsi que l'équipe « Interprétation Abstraite et Sémantique » du LIENS et les entreprises Gemplus et Trusted Logic étudie la sémantique formelle des langages Java et Javacard ainsi que des techniques d'analyse de programmes pour garantir leur sécurité.

8.2 Actions européennes

8.2.1 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, France Telecom, Nokia, Dassault-Aviation, Trusted Logic et Xerox.

8.2.2 Pologne

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

8.3 Actions internationales

8.3.1 Uruguay

Nous participons à un projet de collaboration avec l'Universidad de la República (Uruguay). Ce projet, déposé auprès du Ministère des Affaires Étrangères, concerne l'intégration de méthodes de démonstrations interactives et par vérification de modèle avec une application à la certification du code embarqué dans des stimulateurs cardiaques.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

9.1.1 Responsabilités éditoriales

Jean-Pierre Jouannaud est editeur de *Constraints*. Jean Goubault-Larrecq est éditeur invité du numéro spécial sur la sécurité informatique du *Journal of Telecommunications and Information Technology*.

Pierre-Louis Curien et Gilles Dowek ont été membres du comité éditorial de l'école de printemps *Semantics of programming languages*. Jean Goubault-Larrecq et Jean-Pierre Jouannaud ont été membres du comité éditorial de la conférence *International Joint Conference on Automated Reasonning*. Gilles Dowek et Christine Paulin ont été membres du comité éditorial de la conférence *Theorem proving in higher-order logics*. Gilles Dowek a été membre du comité éditorial de la conférence *Computer Science Logic*. Jean-Pierre Jouannaud a été membre des commités programme de *Logic in Computer Science* et *International Colloquium on Automata, Languages and Programming*. Jean Goubault-Larrecq a été membre du comité éditorial de la conférence *Logic for Programming, Artificial Intelligence and Reasoning*. Christine Paulin a été membre du comité éditorial des conférences *Rewriting Techniques and Applications* et *Computer Aided Verification*. Gilles Dowek a été membre du comité éditorial du worskop *Strategies*. Jean Goubault-Larrecq a organisé le premier workshop *Logical Aspects of Cryptographic Protocol Verification*.

9.1.2 Jurys

Benjamin Werner a été rapporteur de la thèse de doctorat de Sylvain Boulmé (Paris 6). Gilles Dowek a été rapporteur de la thèse d'Éduardo Bonelli. Il a participé au jury de la thèse de Éduardo Bonelli, Frédéric Blanqui et Micaela Mayero. Christine Paulin a participé au jury des thèses de Dominique Ambroise, Frédéric Blanqui (Université Paris 11) et Pierre Courtieu (Université Paris 11). Jean Duprat et Jean Goubault-Larrecq ont été rapporteurs de la thèse de Guillaume Gillard.

Jean Goubault-Larrecq a participé au jury de la thèse de Nabil El Kadhi et au jury d'Habilitation de Delia Kesner.

Jean Goubault-Larrecq et Hugo Herbelin ont participé au jury de la thèse d'Alexandre Miquel.

9.1.3 Visites

Hugo Herbelin a fait un séjour de 3 mois à l'université de l'Oregon à Eugene aux États-Unis.

Gilles Dowek a passé trois mois en résidence au laboratoire ICASE-NASA Langley. Jean-Christophe Filliâtre et Micaela Mayero ont été invités deux semaines dans ce même laboratoire.

Gilles Dowek a effectué une visite à l'Institut du software à l'Académie des Sciences de Beijing, à l'Université Jiao Tong à Shanghai et à l'Université de Nankin.

David Delahaye, Micaela Mayero et Alexandre Miquel sont partis en post-docs à l'Université technologique de Chalmers (Göteborg, Suède). Frédéric Blanqui est parti en post-doc à l'Université de Cambridge.

9.1.4 Colloques

Gilles Dowek et Alexandre Miquel ont participé à la conférence *Typed Lambda Calculus and Applications* à Cracovie, en Pologne, où ils ont présenté un exposé chacun.

Gilles Dowek, Jean Goubault-Larrecq et Stéphane Vaillant ont participé à la conférence IJCAR. Gilles Dowek a participé au workshop *Stratégies* et au workshop MERLIN où il a présenté un exposé.

Alexandre Miquel et Clément Renard ont participé au premier congrès franco-américain de mathématiques à Lyon. Alexandre Miquel y a présenté un exposé.

Benjamin Werner, Alexandre Miquel et Clément Renard ont participé à un groupe de travail sur Coq et les mathématiques à l'Université de Nice.

Frédéric Blanqui a participé à la conférence *Logic in Computer Science* où il a fait une présentation.

Frédéric Blanqui, Gilles Dowek et Stéphane Vaillant ont participé à la réunion annuelle du groupe de travail « MÉLANGES » dans le cadre du GDR/PRC « Algorithmique, Langage et Programmation ». Ils y ont présenté un exposé chacun.

Jean-Christophe Filliâtre a été invité au séminaire des élèves de l'ENS Lyon où il a donné un exposé.

David Delahaye, Gilles Dowek, Pierre Letouzey et Micaela Mayero ont participé aux *Journées Francophones des Langages applicatifs*. David Delahaye et Micaela Mayero y ont présenté un exposé et Gilles Dowek un exposé invité.

Pierre Letouzey a participé a l'Ecole des Jeunes Chercheurs en Programmation, où il a présenté ses travaux de thèse.

Gilles Dowek a présenté un exposé au séminaire *Philosophie et Mathématiques* [52].

Gilles Dowek a participé au colloque Natural Deduction à Rio, où il a présenté un exposé invité.

Gilles Dowek a participé au *Workshop on Logic, Language, Information and Computation* à Brasilia, où il a présenté un exposé invité et un tutorial.

Jean Goubault-Larrecq, Christine Paulin et Jean-Christophe Filliâtre ont participé à la conférence *Computer Aided Verification*. Jean-Christophe Filliâtre y a présenté un exposé.

Micaela Mayero a participé à la *29ème École de printemps d'informatique théorique et arithmétique des ordinateurs* où elle a fait une démonstration.

Christine Paulin a participé au workshop *Dependent Types in Practical Programming* à Dagstuhl. Elle y a présenté un exposé invité.

Christine Paulin a participé à la conférence *Theoretical Aspects of Computer Software*. Elle y a présenté un exposé invité.

Jean Goubault-Larrecq a participé aux journées *Systèmes infinis* où il a présenté un exposé invité.

Jean Goubault-Larrecq a participé au 14ème *International Computer Security Foundations Workshop* où il a présenté un exposé.

Jean Goubault-Larrecq a participé à la conférence SAS.

Jean Goubault-Larrecq a participé à la première conférence *Algebraic Topological Methods in Computer Science*, où il a présenté un exposé invité.

Jean Goubault-Larrecq et Christine Paulin ont participé à la conférence *Computer Science Logic*, Jean Goubault-Larrecq y a présenté un exposé.

Benjamin Werner a fait un exposé au séminaire de l'Institut Mathématique de Luminy (Université d'Aix-Marseille).

Jean-Pierre Jouannaud a participé à l'organisation d'une rencontre « Paris-Tokyo-Informatique » à Tokyo où il a presenté un exposé.

Jean-Pierre Jouannaud a participé à la conférence RPC à Sendai, où il a donné un exposé invité.

Jean-Pierre Jouannaud et Christine Paulin ont participe à un « Workshop Coq » à Tokyo. Ils ont presenté un exposé invité.

9.1.5 Responsabilités diverses

Bruno Barras est Conseiller Scientifique de la société *Trusted Logic*.

Jean-Pierre Jouannaud est directeur du *Laboratioire d'Informatique de l'École Polytechnique*.

Christine Paulin est membre du *steering comittee* de l'*European Association for Computer Science Logic* et du *Educational European Forum*.

Jean-Pierre Jouannaud est membre du conseil scientifique de *European Association for Computer Science Logic* et de l'*European Association for Theoretical Computer Science*.

Gilles Dowek est membre du Steering Committee du working group *Types*.

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 l'*Association Française d'Informatique Théorique*.

Olivier Desmettre est webmaître des serveurs `http://coq.inria.fr/` et `http://logical.inria.fr/`.

9.2 Enseignement

Jean-Pierre Jouannaud a encadré la thèse de Frédéric Blanqui, qui a soutenu. Laurence Puel a encadré la thèse de Pierre Courtieu, qui a soutenu. Benjamin Werner a encadré la thèse de David Delahaye qui a soutenu. Gilles Dowek a encadré la thèse de Micaela Mayero qui a soutenu. Hugo Herbelin a encadré la thèse d'Alexandre Miquel qui a soutenu. Jean Goubault-Larrecq a encadré la thèse de Nabil El Kadhi qui a soutenu. Jean-Pierre Jouannaud encadre les theses de Jacek Chrząszcz, de Daria Walukiewicz-Chrząszcz et d'Antoine Kremer. Benjamin Werner a encadré le stage de DEA de Clément Renard. Jean Goubault-Larrecq a encadré le stage de DEA de Pierre Corbineau. Christine Paulin encadre la thèse de Pierre Letouzey et de Patrick Loiseleur. Christine Paulin et Pierre Crégut encadrent la thèse de Cuihtlauac Alvarado. Christine Paulin et Jean-François Monin encadrent la thèse de Kristen Le Liboux. Gilles Dowek encadre la thèse de Stéphane Vaillant. Benjamin Werner encadre la thèse de Benjamin Grégoire. Bruno Barras encadre la thèse de Clément Renard. Bruno Barras encadre les travaux d'Olivier Desmettre. Judicaël Courant encadre la thèse de Pierre Corbineau. Jean

Goubault-Larrecq encadre les thèses de Muriel Roger, Kumar Neeraj Verma et Éva Rose. Judicaël Courant le post-doc de Huang Guan-Shieng.

Benjamin Werner, Jean-Pierre Jouannaud, Bruno Barras, Hugo Herbelin, Christine Paulin et Jean Goubault-Larrecq enseignent dans le DEA Programmation. Hugo Herbelin a donné aussi une série d'exposés dans le programme de *Ph. D* de l'Université de l'Oregon à Eugene sur ses travaux avec Pierre-Louis Curien sur la dualité du calcul. Gilles Dowek a donné un cours à l'*European Summer School in Logic, Language and Information* et à l'Université de Nanjing [53]. Bruno Barras et Benjamin Werner ont donné un cours à l'*École des jeunes chercheurs en programmation*. Christine Paulin a organisé une formation de trois jours au système Coq qui a réuni une quinzaine de participants issus de l'industrie et du milieu académique. Bruno Barras, David Delahaye et Benjamin Werner sont intervenus comme formateurs.

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. Judicaël Courant a donné un cours sur les modules au Magistère Informatique et Modélisation de l'Ecole Normale Supérieure de Lyon. Judicaël Courant a encadré le projet de compilation de la maîtrise d'informatique à l'Université Paris-Sud. Gilles Dowek est chargé de cours à l'École Polytechnique. Benjamin Werner assure un cours de Coq à l'ENSTA. Hugo Herbelin a donné un cours dans le programme de *master* de l'Université de l'Oregon à Eugene portant sur les principes des langages de programmation. Le langage Objective Caml a servi de base à la description de la sémantique fonctionnelle des structures de programmation considérées. Benjamin Grégoire a donné des TP de compilation à Orsay. Benjamin Grégoire a donné des TP de Java à l'École Polytechnique.

Micaela Mayero a été ATER à l'université Paris 6 jusqu'au 31 août. A ce titre, elle a donné des TD/TP de programmation en Caml light en licence d'informatique ainsi que des TD/TP de Scheme en DEUG MIAS 1ère année. David Delahaye a été ATER à l'Université Pierre et Marie Curie (Paris 6), jusqu'au 31 août 2001. À ce titre, il a dispensé des TD-TP d'architecture en DEUG MIAS deuxième année, et des TD-TP de programmation (Caml) en licence d'Informatique. Alexandre Miquel a été moniteur à l'Université Paris 12 (Créteil), où il a donné des TD et TP d'algorithmique et de programmation en C en DEUG MIAS deuxième année.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J.-C. Filliâtre, E. Giménez, H. Herbelin, G. Huet, H. Laulhère, C. Muñoz, C. Murthy, C. Parent-Vigouroux, P. Loiseleur, C. Paulin-Mohring, A. Saïbi, B. Werner, « The Coq Proof Assistant, Reference Manual ».

[2] B. Barras, *Auto-validation d'un système de preuves avec familles inductives*, Thèse de doctorat, Université Paris 7, 1999.

[3] T. Coquand, G. Huet, « The Calculus of Constructions », *Information and Computation 76*, 1988, p. 95–120.

[4] J. Courant, « A Module Calculus for Pure Type Systems », *in : TLCA'97*, *LNCS*, Springer-Verlag, p. 112 – 128, 1997.

[5] G. Dowek, T. Hardin, C. Kirchner, « Theorem proving modulo », *rapport de recherche n°3400*, Institut National de Recherche en Informatique et en Automatique, 1998, `http://www.inria.fr/rrrt/rr-3400.html`.

[6] G. Dowek, « La vérification automatique de démonstrations », *in : Technique et Science Informatique : Informatiques, enjeux, tendance et évolutions*, R. Jacquart (éditeur), *19*, 1-2-3, Hermès, 2000, p. 195–202.

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

[8] H. Herbelin, « Games and Weak Head Reduction for Lambda-Calculus + Catch », *in : Proceedings of Typed Lambda Calculi and Applications, 1997*, P. de Groote et J. R. Hindley (éditeur), *LNCS*, *1210*, Springer-Verlag, Nancy, France, April 1997.

[9] C. Paulin-Mohring, « Inductive Definitions in the System Coq - Rules and Properties », *in : Proceedings of the conference Typed Lambda Calculi and Applications*, M. Bezem, J.-F. Groote (éditeurs), *Lecture Notes in Computer Science*, 664, 1993. LIP research report 92-49.

[10] B. Werner, *Une théorie des constructions inductives*, Thèse de doctorat, Université Paris 7, 1994.

Livres et monographies

[11] G. Dowek, *La Lógica*, Siglo Veintiuno Editores, 2001.

Thèses et habilitations à diriger des recherches

[12] F. Blanqui, *Théorie des Types et Réécriture (Type Theory and Rewriting)*, thèse de doctorat, Université Paris-Sud (France), 2001.

[13] P. Courtieu, *Représentation des structures non libres en théorie des types*, thèse de doctorat, Université Paris-Sud, Laboratoire de Recherche en informatique, 2001.

[14] D. Delahaye, *Conception de langages pour décrire les preuves et les automatisations dans les outils d'aide à la preuve : une étude dans le cadre du système* Coq, thèse de doctorat, Université Pierre et Marie Curie (Paris 6), Décembre 2001.

[15] N. El Kadhi, *Vérification Statique des Programmes Cryptographiques*, thèse de doctorat, Université de Tunis II, faculté des sciences de Tunis, juillet 2001.

[16] M. Mayero, *Formalisation et Automatisation de Preuves en Analyse et en Analyse Numérique*, thèse de doctorat, Université Paris 6, Décembre 2001.

[17] A. Miquel, *Le Calcul des Constructions implicite : Syntaxe et Sémantique*, thèse de doctorat, Université Paris 7 – Denis Diderot, Décembre 2001.

Articles et chapitres de livre

[18] F. Blanqui, J.-P. Jouannaud, M. Okada, « Inductive Data Type Systems », *Theoretical Computer Science 227*, 2001.

[19] A. Bouhoula, J.-P. Jouannaud, « Automata-Driven Automated Induction », *Information and Computation 169*, 2001, p. 1–22.

[20] P.-L. Curien, « Symmetry and interactivity in programming », *Bulletin of Symbolic Logic*, 2001, à paraître.

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

[22] G. Dowek, « About folding-unfolding cuts and cuts modulo », *Journal of Logic and Computation 11*, 3, 2001, p. 419–429.

[23] G. Dowek, « Higher-order unification and matching », *in : Handbook of Automated Reasoning*, Elsevier, 2001, p. 1009–1062.

[24] J.-C. Filliâtre, « Formal Proof of a Program : Find », *Science of Computer Programming*, 2001, à paraître.

[25] H. Herbelin, « Explicit Substitutions and Reducibility », *Logic and Computation 11*, 3, 2001.

[26] L. Moreau, J. Duprat, « A construction of distributed reference counting », *Acta Informatica 37*, 2001, p. 563–595.

Communications à des congrès, colloques, etc.

[27] F. Blanqui, « Definitions by rewriting in the Calculus of Constructions », *in : LICS'01*, 2001.

[28] R. Butler, V. Carreño, G. Dowek, C. Muñoz, « Formal Verification of Conflict Detection Algorithms », *in : Proceedings of the 11th Working Conference on Correct Hardware Design and Verification Methods CHARME 2001*, *LNCS*, *2144*, p. 403–417, Livingston, Scotland, UK, 2001. A long version appears as report NASA/TM-2001-210864.

[29] H. Comon, J.-P. Jouannaud, « First order constraints in shallow equational theories », *in : Proc. International Workshop on Constraints*, 2001.

[30] P. Courtieu, « Normalized types », *in : Proceedings of CSL2001*, *LNCS*, 2001. to appear.

[31] D. Delahaye, M. Mayero, « `Field` : une procédure de décision pour les nombres réels en `Coq` », *in : Journées Francophones des Langages Applicatifs, Pontarlier*, INRIA, Janvier 2001.

[32] G. Dowek, T. Hardin, C. Kirchner, « A Completeness theorem for an extension of first-order logic with binders », *in : Merlin, Technical Report 2001/26, Mathematics and Computer Science, University of Leicester*, p. 49–63, 2001.

[33] G. Dowek, « Confluence as a cut elimination property », *in : Workshop on Logic, Language, Information and Computation*, 2001. Invited talk.

[34] G. Dowek, « The Stratified Foundations as a theory modulo », *in : Typed Lambda Calculi and Applications*, *Lecture Notes in Computer Science*, *2044*, Springer-Verlag, 2001.

[35] G. Dowek, « What is a Theory ? », *in : International Symposium on Theoretical Aspects of Computer Science*, 2002. Invited talk.

[36] J.-C. Filliâtre, S. Owre, H. Ruess, N. Shankar, « ICS : Integrated Canonization and Solving (Tool presentation) », *in : Proceedings of CAV'2001*, G. Berry, H. Comon, A. Finkel (éditeurs), *Lecture Notes in Computer Science*, *2102*, Springer-Verlag, p. 246–249, 2001.

[37] J. Goubault-Larrecq, « Well-Founded Recursive Relations », *in : Computer Science Logic*, *Lecture Notes in Computer Science*, *2142*, Springer-Verlag, p. 484–497, 2001, `http://www.lsv.ens-cachan.fr/Publis/PAPERS/Gou-csl2001.ps`.

[38] J.-P. Jouannaud, A. Rubio, « Higher-Order Recursive Path Orderings "à la carte" », *in : First Japanese Conference on Logic and Computation*, 2001. Invited Lecture.

[39] A. Miquel, « The Implicit Calculus of Constructions : Extending Pure Type Systems with an Intersection Type Binder and Subtyping », *in : Typed Lambda Calculi and Applications*, *Lecture Notes in Computer Science*, *2044*, Springer-Verlag, 2001.

[40] C. Paulin-Mohring, « Modelisation of Timed Automata in Coq », *in : International Symposium on Theoretical Aspects of Computer Software (TACS2001)*, 2001. Invited paper.

[41] X. Rival, J. Goubault-Larrecq, « Experiments with Finite Tree Automata in Coq », *in : Theorem Proving in Higher Order Logics*, *Lecture Notes in Computer Science*, *2152*, Springer-Verlag, p. 362–377, 2001, `http://www.lsv.ens-cachan.fr/Publis/PAPERS/RivGou-tphol01.ps`.

[42] M. Roger, J. Goubault-Larrecq, « Log Auditing through Model Checking », *in : Computer Security Foundations Workshop*, IEEECSP, p. 220–236, 2001, `http://www.lsv.ens-cachan.fr/Publis/PAPERS/RogGou-csfw01.ps`.

Rapports de recherche et publications internes

[43] J. Courant, « $\mathcal{MC}_2$: A Module Calculus for Pure Type Systems », *Research Report n°1292*, LRI, septembre 2001, `http://www.lri.fr/~jcourant/papers/01/mc2rr.ps`.

[44] G. Dowek, C. Muñoz, A. Geser, « Tactical Conflict Detection and Resolution in a 3-D Airspace », *rapport de recherche n°NASA/CR-2001-210853 ICASE Report No. 2001-7*, ICASE-NASA Langley, ICASE Mail Stop 132C, NASA Langley Research Center, Hampton VA 23681-2199, USA, April 2001.

[45] J. Duprat, « A Coq Toolkit for Graph Theory », *rapport de recherche n°2001-15*, LIP, 2001.

[46] J. Goubault-Larrecq, E. Goubault, « On the Geometry of Intuitionistic S4 Proofs », *rapport de recherche n°LSV-01-8*, Lab. Specification and Verification, ENS de Cachan, Cachan, France, novembre 2001, 107 pages, `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/rr-lsv-2001-8.rr.ps`.

[47] J. Goubault-Larrecq, « Higher-Order Automata, Pushdown systems, and Set Constraints », *rapport de recherche n°LSV-01-9*, Lab. Specification and Verification, ENS de Cachan, Cachan, France, novembre 2001, 15 pages, `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/rr-lsv-2001-9.rr.ps`.

[48] C. Renard, « Un peu d'extensionnalité en Coq », *Mémoire de dea*, INRIA Rocquencourt, septembre 2001.

[49] S. Vaillant, « A finite first order presentation of set theory », *Research Report n°4344*, INRIA, 2001, `http://www.inria.fr/rrrt/rr-4344.html`.

Divers

[50] P.-L. Curien, « Introduction à la ludique », 2001, `http://www.pps.jussieu.fr/~curien`.

[51] G. Dowek, B. Werner, « Peano's arithmetic as a theory modulo », soumis à publication, 2001.

[52] G. Dowek, « À propos de quelques démonstrations pas très convaincantes », Séminaire de Philosophie et Mathématiques, 2001.

[53] G. Dowek, « Introduction to proof theory », Course notes at ESSLLI, 2001.

[54] H. Herbelin, « Compiling pattern-matching in Coq », manuscript, 2001.

[55] H. Herbelin, « Floating universes in the Calculus of Inductive Constructions », manuscript, 2001.