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-Rcoquencourt et du Laboratoire de Recherche en Informatique de l'Université de Paris XI.

1 Composition de l'équipe

Responsable scientifique

Gilles Dowek [CR, INRIA]

Co-responsable scientifique

Christine Paulin [Professeur, Université de Paris XI]

Responsable permanent

Benjamin Werner [CR, INRIA]

Assistante de projet (commun avec Cristal)

Nelly Maloisel [Adjointe Administrative INRIA]

Personnel INRIA

Bruno Barras [CR, INRIA Rocquencourt]

Hugo Herbelin [Maître de Conférence Paris X, détache à l'INRIA]

Personnel Paris XI

Judicaël Courant [Maître de Conférences, Paris XI]

Jean-Christophe Filliâtre [ATER, Paris XI]

Jean-Pierre Jouannaud [Professeur, Paris XI]

Collaborateurs extérieurs

Philippe Audebaud [Maître de conférences, ENS-Lyon]

Jean Duprat [Maître de conférences, ENS-Lyon]

Jean Goubault-Larrecq [ENS Cachan]

Gérard Huet [Projet Cristal]

Professeur Invité

Mitsuhiro Okada [Université de Keio]

Doctorants

Frédéric Blanqui [Allocataire MENRT, Paris XI]

Jacek Chrzaszcz [Thèse en co-tutelle LRI, Université de Varsovie]

Pierre Courtieu [demi-ATER, Paris XI]

David Delahaye [demi-ATER, INRIA]

Nabil El Khadi [Bourse INRIA, G.I.E. Dyade]

Benjamin Grégoire [Allocataire MENRT]

Pierre Letouzey [Élève à l'École Normale Supérieure, Paris XI]

Patrick Loiseleur [Paris XI]

Micaela Mayero [demi-ATER, INRIA]

Alexandre Miquel [Allocataire MENRT, INRIA]

Éva Rose [Bourse INRIA, G.I.E. Dyade]

Stéphane Vaillant [Allocataire MENRT, INRIA]

Daria Walukiewicz-Chrzaszcz [Thèse en co-tutelle LRI, Université de Varsovie]

2 Présentation et objectifs généraux

Le projet LogiCal mène des recherches sur les *systèmes de traitement de démonstrations mathématiques*, c'est-à-dire des systèmes capables de vérifier et plus généralement de traiter des démonstrations mathématiques.

Construire et utiliser un système de traitement de démonstrations mathématiques répond à la quête d'une certaine forme 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. Sur un plan plus concret, vérifier automatiquement une démonstration permet d'atteindre un nouveau degré de certitude quant à sa correction. Cela est particulièrement utile pour les arguments qui justifient la correction de matériels et de logiciels qui se révèlent souvent à l'utilisation plein d'erreurs, y compris certains systèmes pourtant largement diffusés.

Un des axes majeurs de nos travaux est le développement et la valorisation du système Coq. Ce système, qui a aujourd'hui une communauté importante d'utilisateurs industriels et académiques, est une implémentation d'une variante de la théorie des ensembles : *Le Calcul des Constructions Inductives*.

La conception d'un système de traitement de démonstrations mathématiques mobilise des compétences larges en informatique et en mathématiques. Ces comptétences concernent l'architecture de ces systèmes, l'axiomatisation des mathématiques, l'organisation de vastes bibliothèques de résultats, l'automatisation de la recherche de démonstrations, la manière de représenter mathématiquement des programmes et de formaliser leurs propriétés, ... En amont de l'activité de développement, notre projet s'investit dans la recherche sur ces sujets afin de pouvoir être acteur de l'émergence des fonctionnalités des systèmes de demain. 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.

Le projet LogiCal fait partie du projet Calife du *Réseau National de Recherche en Télécommunications*, de l'Action de Recherche Coopérative SJava du *working group* européen Types et du projet européen Verificard.

3 Fondements scientifiques

3.1 Logique et calcul

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

Une des premières questions qui se posent quand on veut bâtir un système capable de vérifier, et plus généralement traiter, des démonstrations mathématiques est celui du choix d'une formalisation des mathématiques. Une théorie traditionnellement utilisée pour formaliser les mathématiques est la théorie des ensembles, exprimée dans le calcul des prédicats du premier ordre. Mais, si cette théorie est satisfaisante dans bien des cas, elle ne l'est pas complètement quand on cherche à réaliser un système de traitement de démonstrations mathématiques. Cela s'explique d'une part par le fait que l'on veut écrire des développements mathématiques qui puissent être vérifiés par un programme, d'autre part par le fait que parmi ces développements, on accorde beaucoup d'importance à ceux traitant des programmes et de leur propriétés.

Vouloir écrire des développements mathématiques vérifiables par un programme impose de les décrire dans un très grand détail. 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 leur propriétés amène à s'intéresser à l'aspect effectif des mathématiques et en particulier au rapport entre démonstrations constructives et algorithmes, 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 (voire un langage de programmation comportant

des traits impératifs ou objets ...).

3.2 Calcul des Constructions Inductives

Mots clés : calcul des constructions inductives.

Nous développons un système de traitement de démonstrations appelé Coq et fondé sur le *Calcul des Constructions Inductives*. Ce formalisme permet de définir des algorithmes comme de simples fonctions, il permet de définir des prédicats inductivement par un ensemble de clauses. Il contient donc un langage de programmation fonctionnel et un langage de programmation logique comme sous-langages.

Ce formalisme articule raisonnement et calcul, en particulier parce qu'un terme obtenu en appliquant une fonction définie algorithmiquement à un argument se calcule en un terme plus simple sans que l'égalité entre ces deux termes doive être démontrée. Une autre originalité de ce formalisme est que les démonstrations sont des objets à part entière, au même titre que les nombres, les ensembles ou les fonctions. Plus précisément, les démonstrations sont des fonctions définies algorithmiquement dans la tradition de Brouwer, Heyting et Kolmogorov.

3.3 Environnement interactif de démonstrations

Mots clés : tactique.

Le système Coq permet de construire des théories mathématiques en introduisant des définitions, des hypothèses, en énonçant des propositions et enfin en donnant une démonstration de ces propositions. La correction de cette démonstration est alors vérifiée par le système.

De plus, le système Coq fournit une aide à l'utilisateur dans la rédaction même de ses démonstrations en lui permettant de les construire interactivement avec des tactiques, c'est-à-dire des programmes qui construisent, à partir d'une proposition à démontrer, un ensemble de nouvelles propositions suffisantes pour démontrer la proposition initiale.

3.4 Preuves et 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. Par exemple, le terme $2 + 5$ se réduit sur la valeur 7.

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 en fait une fonction qui, à tout objet a associe un objet b et une démonstration de $Q(a, b)$. À partir de cette

démonstration, 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 a se réduira pour fournir la sortie b.

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 vue comme une formule mathématique et le programme certifié est obtenu à partir de la démonstration de cette formule.

Cette interprétation est particulièrement adaptée à la programmation fonctionnelle mais fournit également des techniques de démonstration de programmes impératifs.

4 Domaines d'applications

Mots clés : informatique médicale, transports, télécomunications, 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 cruciaux.

Essentiellement cela concerne les applications où un défaut de fonctionnement met en péril la vie humaine, l'environnement ou des sommes d'argent importantes. Les domaines d'application sont donc naturellement l'informatique médicale, les transports, les télécomunications, le commerce électronique et plus généralement l'informatique en réseau.

Le système Coq a été ou est actuellement utilisé pour la modélisation de protocoles cryptographiques utilisés en commerce électronique, la modélisation de politiques de sécurité dans le cadre d'applications Java sur cartes à puce, des études d'algorithmes de contrôle de conformité dans les réseaux ATM en vue de leur normalisation, des démonstrations de compilateur de langages réactifs, la certification d'algorithmes de calcul formel, ...

Ces développements sont pour la plupart menés à bien hors de notre projet par nos partenaires, en particulier nos partenaires industriels (Trusted Logic, France Telecom, ...)

5 Logiciels

Participants : Bruno Barras, David Delahaye, Jean-Christophe Filliâtre, Hugo Herbelin, Christine Paulin [Correspondante].

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 :

- un système de bibliothèques mathématiques modulaires permettant de compiler et recharger rapidement des théories mathématiques;
- la possibilité d'introduire des notations spécifiques par l'utilisation de grammaires et fonctions d'impression interprétées;
- la possibilité de développer des tactiques comme des programmes Ocaml sophistiqués qui peuvent ensuite être chargés et utilisés dans l'environnement.

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

- la modélisation et la démonstration du protocole d'authentification CSET utilisé en commerce électronique,
- une démonstration 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.

Coq est utilisé sur une centaine de sites. Nous avons des utilisateurs intensifs dans le milieu industriel (France Telecom Lannion, Dassault-Aviation, Trusted Logic, CP8, Schlumberger et au sein de l'action VIP du GIE Dyade), dans le milieu académique en Europe (Ecosse, Hollande,

Espagne, Italie, Portugal) et en France (Bordeaux, Lyon, Marseille, Nancy, Nantes, Nice, Paris 6, 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 Développement du système Coq

6.1.1 Restructuration du système : Coq Version 7

Participants : Hugo Herbelin, Jean-Christophe Filliâtre, David Delahaye.

Mots clés : Coq, environnement de démonstrations certifiées.

Développé depuis de plus de dix ans, le système Coq avait atteint une taille importante. Il était devenu nécessaire de repenser son organisation et en particulier les fonctionnalités de son noyau afin de permettre différentes extensions et évolutions - en particulier l'intégration d'un système de modules paramétriques et l'auto-certification ("boot-strap") du noyau. Un chantier important de rénovation du système a été entamé à l'automne 1999 et s'est déroulé tout au long de l'année 2000. Cet effort s'est concentré sur plusieurs points : penser une nouvelle architecture qui place le vérificateur de démonstrations au cœur du système, nettoyer le code afin de le rendre plus clair, plus sûr et mieux documenté, remplacer les invariants dynamiques par des contraintes de typage, choisir des structures de données simples et adaptées au besoin de chaque module du système. Cet effort a aussi été l'occasion de réécrire l'algorithme de compilation des définitions par filtrage, d'ajouter une construction primitive pour les définitions locales, de permettre l'usage de noms longs pour l'accès aux définitions dans les modules, d'introduire la localisation des erreurs de l'utilisateur.

Cet effort a abouti à une distribution d'une nouvelle version Coq (la version 7) à la fin de l'année 2000. Cette nouvelle version est à la fois plus sûre, plus légère, plus conviviale et plus rapide que la version 6.

6.1.2 Langages de démonstrations et de tactiques

Participants : David Delahaye, Benjamin Werner.

Mots clés : système de traitement de démonstrations mathématiques, langage de démonstrations, langage de tactiques, métalanguage.

David Delahaye a porté en Coq V7 le langage de démonstrations qu'il avait développé en Coq V6. Il s'est tout particulièrement penché sur le problème des métavariables afin de les fusionner correctement avec celles du système et celles provenant du langage de tactiques. En ce qui concerne les parties purement procédurales, il a décidé d'imposer qu'elles résolvent les buts sur lesquels elles s'appliquent afin de ne pas perdre possibilité de lire les démonstrations. Le procédural est donc réservé à des buts que l'on sait prouver. Enfin, il a modifié plus en avant le langage afin de couvrir les termes de Coq plus en profondeur.

David Delahaye a également porté en Coq V7 le langage de tactiques $\mathcal{L}_{tac}$ qu'il avait développé en Coq V6. Il a complété ce langage en fournissant du filtrage sur les sous-termes et en permettant un retour arrière sur les occurrences. Il a aussi mis au point un système d'échec (tactique Fail) à plusieurs niveaux et, afin de pouvoir mettre au point ce langage et les tactiques de manière générale, il a développé un debugger spécifique plus sensible aux nouveaux aspects tels que le retour arrière. Dans l'optique de pouvoir utiliser les avantages de $\mathcal{L}_{tac}$ dans ML, David Delahaye a développé une interface à base de quotation qui simplifie considérablement l'écriture des tactiques en ML tout en restant complet au sens de Turing.

Des exemples conséquents ont été développés afin de montrer la pertinence de $\mathcal{L}_{tac}$. Par exemple, un travail commun avec Micaela Mayero a débouché sur le codage de la tactique Field (voir section 6.2.1). David Delahaye a redéveloppé totalement les tactiques Tauto et Intuition en utilisant $\mathcal{L}_{tac}$ et le système de quotation précédemment élaboré. Par rapport au code d'origine, il a obtenu un gain de taille considérable passant de 2000 lignes de code à 100 lignes. En temps, un gain avoisinant un facteur 3 en moyenne a été constaté.

6.1.3 Extraction

Participants : Pierre Letouzey, Christine Paulin.

Durant son stage de DEA[44], encadré par Christine Paulin, Pierre Letouzey a développé une nouvelle méthode d'extraction de programmes à partir de démonstrations pour le Calcul des Constructions Inductives. La méthode utilisée actuellement dans Coq ne prend pas en compte la hiérarchie d'univers et n'accepte que partiellement les définitions inductives. Pierre Letouzey a mis au point un nouvel algorithme d'extraction qui repose sur des règles de réduction appliquées à des termes annotés par leur type. Contrairement à ce qui était fait précédemment, une trace des termes logiques qui ne servent pas au calcul reste dans le terme extrait. Il a étudié précisément les propriétés de ce calcul. Sa méthode devrait servir de base à l'implantation de l'extraction de programmes en Coq V7.

En parallèle de ce travail d'implémentation, Pierre Letouzey a débuté en Octobre 2000 une thèse intitulée "Un environnement de programmation Caml-Coq", cette thèse est destinée à permettre l'écriture de programmes Caml comprenant des spécifications en commentaires puis la démonstration en Coq des obligations de preuves.

6.2 Développement en Coq

6.2.1 Tactique Field

Participants : David Delahaye, Micaela Mayero, Benjamin Werner, Gilles Dowek.

Mots clés : corps commutatifs, nombres réels, réflexion, langage de tactiques.

David Delahaye et Micaela Mayero ont développé une tactique - appelée Field - qui permet de décider d'égalités sur les corps commutatifs modulo des démonstrations d'inégalités. Ce travail provenait d'un besoin important d'une procédure de décision pour les nombres réels. Cette procédure est générale et fonctionne pour tous les corps commutatifs. Les développements basés sur les nombres rationnels pourront donc l'utiliser. Du point de vue du code proprement

dit, cette tactique a montré l'utilité du nouveau langage de tactiques intégré à Coq V7 ($\mathcal{L}_{tac}$). Notamment, il a permis de constater l'adéquation de $\mathcal{L}_{tac}$ dans l'écriture de tactiques réflexives. Le développement est relativement conséquent et avoisine les 900 lignes de code.

6.2.2 Formalisation des nombres réels

Participants : Micaela Mayero, Gilles Dowek.

Mots clés : Coq, nombres réels, limite, dérivée, spécification.

Micaela Mayero a développé une bibliothèque de démonstrations de théorèmes concernant les nombres réels et les fonctions d'une variable réelle : par exemple, le théorème selon lequel la dérivée d'une composition est le produit des dérivées des fonctions composées.

Afin de définir les fonctions transcendantes, Micaela Mayero a formalisé une grande partie de la théorie mathématique concernant les suites de Cauchy et les séries entières. La fonction exponentielle est en passe d'être terminée.

Micaela Mayero a également réalisé, pour les nombres réels, une grammaire ainsi qu'un pretty-printer qui sont intégrés à Coq V7. Dans le but d'augmenter l'automatisation des démonstrations faites sur les réels, une nouvelle tactique appelée Field (voire section 6.2.1). Grâce à ces travaux, et pour répondre à des demandes d'utilisateur pour la Coq V7, Micaela Mayero a «nettoyé» les démonstrations concernant les réels afin d'en diminuer la taille des scripts ainsi que celle des y λ-termes de démonstration.

6.2.3 Preuve d'un dérivateur Fortran

Participants : Micaela Mayero, Gilles Dowek.

Mots clés : méthodes formelles, analyse numérique, dérivation, Odyssée, Coq, Fortran.

Suite à la définition de la sémantique de Fortran effectuée en Coq l'an passé, Micaela Mayero a effectué la démonstration du dérivateur pour les fonctions polynomiales à une variable. Un développement plus important est en cours afin de prouver la correction sur un ensemble plus vaste de programmes (fonctions).

6.2.4 Sécurité

Participants : Jean Goubault-Larrecq, Nabil El Kadhi, Muriel Roger, Éva Rose, Kumar Neeraj Verma, Xavier Rival.

Mots clés : protocoles cryptographiques, audit de logs, analyse statique, sécurité.

Au sein du GIE Dyade, Jean Goubault-Larrecq et son équipe VIP ont continué à développer les trois thèmes liés à la sécurité qu'il avait identifiés en 1999 : d'abord, une technique d'audit de logs basée sur la logique temporelle, sur laquelle Muriel Roger avait passé son stage de DEA, et qui a été améliorée depuis, prenant en compte des problèmes algorithmiques (efficacité) et les besoins des clients (Bull OSS, Bull Trustway, les participants du projet européen ITEA/TASSC, Alcatel/Next&So); cette technique, qui avait déjà fait l'objet d'un brevet Dyade, fera l'objet

d'une publication étendue en 2001. L'audit de logs fait par ailleurs l'objet d'une collaboration avec Mireille Ducassé, de l'IRISA, projet Lande, et de l'INSA Nantes.

Ensuite, une technique d'analyse de protocoles cryptographiques à l'aide de variantes d'automates d'arbre; le prototype de juin 1999 a été étendu pour traiter des principaux en multi-session parallèle, et a fait l'objet de diverses présentations, donc un article à un workshop international [29]. Il est aussi à la source d'un projet de collaboration entre le GIE Dyade, l'IRISA/INSA et Bull Trustway sur la vérification d'une implémentation du protocole SSL. À long terme, il est souhaité que cet outil fournisse aussi une démonstration Coq aussi automatisée que possible de son analyse. C'est dans cette optique que Xavier Rival a développé plusieurs bibliothèques de calculs d'automates d'arbres en Coq [46].

Enfin, l'étude d'un cadre d'analyse statique en vue de vérifier des propriétés de confidentialité d'applets. Nabil El Kadhi a effectué plusieurs publications et communications à ce sujet [26, 25, 27], et soutiendra sa thèse à l'université Tunis II en 2001. Un prototype a été réalisé en Caml, qui a servi à valider les idées de base sur des protocoles classiques comme ceux de Needham-Schroeder et de Yahalom. Ces travaux forment le substrat d'un prototype plus industriel développé par Pierre Boury (Dyade/Bull) dans le cadre du contrat européen ITEA/TASSC (Transactional Added-value Services for SmartCards).

D'autre part, Éva Rose a continué ses travaux de thèse (directeur : Jean Goubault-Larrecq) sur la "lightweight bytecode verification", une technique permettant la vérification de bytecode Java et JavaCard sur des terminaux à faibles ressources. Elle a en particulier étendu son approche à la gestion des exceptions, ainsi qu'à la représentation compacte des certificats légers. Ceci sera présenté dans un rapport de recherche INRIA à venir, qui formera aussi l'essentiel de sa thèse. Ces derniers sont maintenant utilisés au sein de la KVM de Sun (`http://java.sun.com/products/cldc/`). Elle a d'autre part proposé une extension simple à Java permettant un raffinement de la protection des accès aux champs des classes par typage [35].

D'un autre côté, Kumar Neeraj Verma, étudiant de B.Tech. à l'IIT Delhi, qui avait déjà effectué un stage de deux mois chez Dyade sur la réalisation de bibliothèque de BDDs (diagrammes de décision binaire) certifiés, par réflexion en Coq, est revenu passer son DEA à l'INRIA [36]. Il a poursuivi son travail en réalisant un model-checker symbolique complet, prouvé, du mu-calcul modal. Il est maintenant en thèse avec Jean Goubault-Larrecq.

6.2.5 Comptage de références.

Participant : Jean Duprat.

Mots clés : comptage de références, algorithme distribué, sûreté, vivacité, démonstrations formelles, système de traitement de démonstrations mathématiques, Coq.

Le comptage de références est une technique classique utilisée par exemple par les systèmes d'exploitation pour réaliser des ramasse-miettes. Luc Moreau a proposé un algorithme distribué basée sur une machine abstraite. Luc Moreau et Jean Duprat ont utilisé le système Coq pour prouver la vivacité et la sûreté de cet algorithme. Cette implémentation fait partie des contributions de Coq. Jean Duprat a présenté ce travail à TYPES 1999. Ce travail est décrit dans un article [16] à paraître dans *Acta Informatica*.

6.2.6 Bases constructives de la théorie des graphes.

Participant : Jean Duprat.

Mots clés : théorie des graphes, constructions, système de traitement de démonstrations, Coq.

La théorie des graphes est actuellement présentée selon deux facettes : l'utilisation de la théorie des ensembles pour les définitions et théorèmes, l'utilisation d'un langage impératif pour décrire les algorithmes et calculer leur complexité. La cohérence entre ces deux mondes fait généralement beaucoup appel à l'intuition du lecteur. L'utilisation du calcul des constructions permettrait d'unifier spécifications et calculs dans un même monde d'où il sera possible d'extraire des programmes certifiés. Certains travaux ont montré la faisabilité en prouvant un théorème spécifique de théorie des graphes. L'objectif est maintenant de spécifier les bases de la théorie afin de faciliter et d'unifier les démonstrations de théorèmes. Jean Duprat a commencé ce travail avec le concours d'un stagiaire de première année de Magistère Informatique et Modélisation de l'Ecole Normale Supérieure de Lyon, Clément Renard en 1999. Ce travail se poursuit à l'heure actuelle et a été présenté à TYPES 2000 par Jean Duprat.

6.2.7 Algèbres de processus

Participant : Jean Duprat.

Mots clés : pi-calcul, mu-logique, système de traitement de démonstrations mathématiques, Coq.

Ce travail s'amorce sur une étude de cas de contrôleur d'accès proposée par Yves Ledru dans une algèbre de processus. Noël Bernard et Yves Dumond proposent de spécifier le problème en pi-calcul et d'exprimer les propriétés avec la mu-logique. Ainsi, une vérification formelle des propriétés du systèmes pourrait être automatisée. Jean Duprat s'est joint à eux pour tenter d'écrire cela en Coq. Ce travail a débuté en 2000 et doit être poursuivi.

6.2.8 Preuves de code mobile, bytecode typé, compilation de Coq

Participants : Benjamin Grégoire, Benjamin Werner.

Mots clés : Code auto-certifié, Assembleur typé, compilation.

Benjamin Grégoire s'est intéressé a la technologie du code auto-certifié ("Proof-Carrying Code" de P. Lee et G. Necula) et au langage assembleur typé ("Typed Assembly Language" de G. Morisett, K. Crary, D. Walker et N. Glew). Il a cherché a comprendre quel rapport existe entre ces deux techniques de certification de programme (bytecode). Le résultat de ces réflexions est qu'il semble possible de compiler le language Coq vers du bytecode typé. Cela permettrait de fournir une nouvelle méthode de démonstration de programme, à cheval entre l'utilisation de code auto-certifié et d'un assembleur typé, en héritant directement de tous les outils de démonstrations de programmes déjà présents dans Coq.

6.2.9 Automates temporisés

Participants : Emmanuel Freund, Christine Paulin.

Emmanuel Freund durant un stage de maîtrise et Christine Paulin ont développé des bibliothèques Coq de manipulation d'automates temporisés. Ils ont représenté les p-automates [40] qui ont été élaborés dans le cadre du projet RNRT Calife. Ce développement comprend une axiomatisation du temps qui reprend une partie des axiomes de la théorie des réels développée par Micaela Mayero et une définition des systèmes de transitions sur laquelle repose la représentation des p-automates. Ce travail est décrit dans [40]. Il a été réalisé en parallèle avec une modélisation dans Isabelle-HOL [39]. La représentation en machine des p-automates a permis d'affiner leur définition. D'autre part les types dépendants de Coq se sont révélés utiles pour modéliser la synchronisation d'une famille arbitraire de p-automates.

6.2.10 Algorithmes probabilistes

Participants : Philippe Audebaud, Christine Paulin.

Mots clés : algorithmes probabilistes, sémantique axiomatique, spécification, démonstration.

En collaboration avec Richard Lassaigne (Équipe de Logique de l'Université Paris 7), Philippe Audebaud et Christine Paulin ont poursuivi l'étude de la mécanisation des spécifications et preuves d'algorithmes probabilistes. L'objectif de ce travail est de réunir des compétences diverses afin d'aboutir à la spécification et la preuve d'algorithmes probabilistes classiques comme le Minimal Cut sur les graphes ou le test de primalité sur les entiers.

Comme proposé par Dexter Kozen [Koz81], les programmes probabilistes sont interprétés comme des transformateurs de mesures. Philippe Audebaud a repris ce travail, pour proposer un système d'inférence en sémantique axiomatique, qui étend très exactement le système de Hoare. Ce travail s'appuie sur un travail commun avec Elena Zucca [AZ99] qui mettait en avant la thèse selon laquelle il est possible de construire un tel système d'inférence pour tout langage possédant une sémantique dénotationnelle (directe ou seulement par continuations). De fait, le système proposé est immédiatement extensible en présence d'exceptions et de blocs d'instructions étiquetées. La nécessité de prendre en compte, sur le plan syntaxique, des formules atomiques de la forme $\text{Prob}[A] \leq a$ a posé des contraintes très fortes sur le modèle proposé. Tout cela est discuté plus en détail dans un rapport de recherche en cours de finition.

Christine Paulin étudie plus particulièrement une traduction des programmes fonctionnels probabilistes en théorie des types à l'aide d'une transformation monadique.

[Koz81] D. Kozen, « Semantics of Probabilistic Programs », *Journal of Computer ans System Sciences 22*, 1981, p. 328–350.

[AZ99] P. Audebaud, E. Zucca, « Deriving Proof Rules from Continuation Semantics », *Formal Aspects of Computing 11*, 4, 1999, p. 426–447, `ftp://ftp.ens-lyon.fr/pub/users/LIP/paudebau/Papers/FAC99.ps.gz`.

6.3 Réduction, réécriture, automatisation

6.3.1 Compilation des démonstrations

Participants : Benjamin Grégoire, Hugo Herbelin, Alexandre Miquel, Benjamin Werner.

Mots clés : démonstrations, compilation.

L'une des originalités du système Coq est que son formalisme intègre une notion de calcul. Plus exactement, les objets de Coq correspondent essentiellement à un noyau fonctionnel pur et terminant de ML. De plus, le raisonnement sur ces objets se fait modulo la relation d'exécution de ces programmes (modélisée par la $\beta\delta\iota$-réduction); pour certains problèmes, on peut exploiter cette caractéristique pour réduire d'un ordre de grandeur la taille des démonstrations par rapport à ce qui est possible dans un formalisme plus traditionnel.

Ceci a conduit à un style nouveau de démonstrations, exploitant de manière de plus en plus intensive le mécanisme d'exécution interne au système Coq au point de mettre celui-ci à mal. Hugo Herbelin, Alexandre Miquel, Benjamin Grégoire et Benjamin Werner ont commencé à étudier la mise en œuvre de technologies issues de la compilation des langages fonctionnels pour un système de démonstrations. La proximité de spécialistes de la compilation dans les projets Cristal et Moscova a évidemment favorisé ce travail.

Alexandre Miquel et Benjamin Grégoire ont proposé des premiers prototypes de compilateurs au performances comparables à celles de Caml, mais capables d'effectuer de la *réduction forte*, c'est-à-dire d'exécuter partiellement des programmes incomplets. Il reste à traiter certains aspects du langage (filtrage en particulier) pour interfacer ces prototypes avec Coq.

6.3.2 Sémantique du calcul

Participant : Hugo Herbelin.

Mots clés : appel par nom, appel par valeur, correspondance de Curry-Howard, calcul des séquents, dualité, machines abstraites.

Hugo Herbelin, conjointement avec Pierre-Louis Curien (PPS, Université Paris 7), a développé un λ-calcul (un mini-langage de programmation) interprétant calculatoirement les démonstrations du système formel dit "calcul des séquents de Gentzen", un formalisme clé de la théorie de la démonstration. Ce λ-calcul, nommé $\overline{\lambda}$-calcul, met en évidence de profondes propriétés du mécanisme d'évaluation des langages : une dualité entre terme et contexte (c'est-à-dire entre un programme et son environnement d'utilisation) ainsi qu'une dualité entre appel par nom et appel par valeur. Ce travail a été l'objet d'une communication à la conférence ICFP 2000 [18].

Par ailleurs ce calcul s'est révélé être un excellent outil de description des machines abstraites comme celle mise en œuvre dans le compilateur Objective Caml. Ceci a été l'objet d'une note privée sur la ZINC, la machine abstraite "byte-code" d'Objective Caml.

6.3.3 Réflexion avec traces

Participant : Benjamin Werner.

La réflexion est une technique de démonstration qui exploite l'expressivité calculatoire de la théorie des types et qui permet la construction de démonstrations par réécriture particulièrement efficaces. Elle a fait l'objet, au sein du projet Coq, de la thèse de Samuel Boutin, ainsi que d'une implémentation dans Coq, ensuite améliorée par Patrick Loiseleur.

Pour étendre le domaine d'application de cette technique, ainsi que pour réduire le temps de calcul nécessaire, Benjamin Werner a suggéré d'expliciter les étapes de réécriture dans le terme de démonstration, sous forme de « trace ».

Par ailleurs, ce travail donne lieu à une collaboration avec le projet Prothéo (Unité de Recherche de Nancy), dont le logiciel ELAN semble bien adapté à la construction des traces et France-Telecom R& D à Lannion; deux thèses sont en cours sur le sujet dans ces deux laboratoires.

6.3.4 Lambda-calcul et réécriture

Participants : Frédéric Blanqui, Jean-Pierre Jouannaud, Mitsuhiro Okada, Daria Walukiewicz-Chrząszcz.

Mots clés : réécriture d'ordre supérieur, terminaison, confluence, Calcul des constructions, réécriture, normalisation forte.

La notion de calcul du Calcul des Constructions Inductives n'est pas extensible. Ainsi, l'utilisateur du système Coq peut calculer certaines expressions afin de simplifier ses démonstrations, mais il ne peut pas définir lui-même les règles de calcul utilisées pour cela. Frédéric Blanqui et Daria Walukiewicz-Chrząszcz étudient des extensions du Calcul des Constructions Inductives qui donnent cette facilité à l'utilisateur. Une des questions essentielles dans ce domaine est la conservation de la terminaison, qui est garante de la cohérence du système.

Frédéric Blanqui a étendu à la réécriture d'ordre supérieur un critère de terminaison, le "Schéma Général", qu'il avait déjà développé dans le cas de la réécriture du premier ordre avec Jean-Pierre Jouannaud et Mitsuhiro Okada. La réécriture d'ordre supérieur, à la différence de la réécriture du premier ordre, permet de manipuler des expressions avec des lieurs, comme le λ du λ-calcul, ou tout simplement d'avoir des paramètres fonctionnels qu'on puisse appliquer à des arguments. Elle utilise donc une notion de filtrage plus complexe que la réécriture du premier ordre : le filtrage d'ordre supérieur. Frédéric Blanqui a également montré que le schéma géneral pouvait s'appliquer à différents formalismes de réécriture d'ordre supérieur. Ces résultats ont été présentés à la conférence internationale RTA 2000 [17].

Suivant une piste différente, Daria Walukiewicz-Chrząszcz a travaillé sur la normalisation forte de la réécriture sur les termes du Calcul des Constructions. Il y a deux ans, elle a défini un système de types, consistant en les règles du Calcul des Constructions et la règle définissant la réécriture. Pour conserver la propriété de normalisation forte du système, les règles de réécriture devaient satisfaire une condition décrite à l'aide d'un ordre bien-fondé HORPO (Higher-order recursive path ordering). Pendant l'année écoulée, elle a fini la démonstration de normalisation du calcul et augmenté la puissance du HORPO pour pouvoir accepter les règles

d'élimination des petit types inductifs. Finalement, elle a vérifié que HORPO et la méthode de démonstration de normalisation qu'elle utilise sont modulaires et donc qu'ils sont adaptés à l'environnement interactif. Elle a présenté ces résultats [37] au workshop Logical Frameworks and Meta-Languages en juillet 2000 à Santa Barbara.

6.3.5 Substitution explicite et normalisation forte

Participant : Hugo Herbelin.

Mots clés : calcul des séquents, normalisation forte, réductibilité, substitutions explicites.

La réductibilité est une technique permettant de prouver des propriétés d'un langage-objet par introjection dans le méta-langage. Hugo Herbelin a développé une variante de réductibilité basée sur les séquents et non sur les types comme il est fait d'habitude. Cette définition se prête bien à la prise en compte d'une substitution explicite dans le langage-objet. En exploitant cette définition, Hugo Herbelin a donné une démonstration de normalisation forte du $\overline{\lambda}$-calcul avec substitutions explicites [30].

6.3.6 Normalisation des démonstrations modulo

Participants : Gilles Dowek, Benjamin Werner.

Mots clés : démonstrations, réécriture.

La présentation d'une logique dans une forme "modulo", c'est-à-dire où les axiomes ont été remplacés par des règles de réécriture, permet de mettre en évidence l'aspect calculatoire des démonstrations et en particulier la notion de coupure.

Gilles Dowek a comparé deux algorithmes de démonstration automatique pour la déduction modulo : la résolution du premier ordre (où les règles sont remplacées par des axiomes) et la résolution modulo proposée en 1998 dans un travail en collaboration avec Thérèse Hardin et Claude Kirchner, où les règles de réécritures sont pleinement exploitées [5]. Il a montré que la résolution modulo pouvait être vue comme une restriction de la résolution dont la complétude dépendait de la propriété d'élimination des coupures du calcul des séquents modulo ce système de réécriture. Ces résultats ont été présentés à la conférence FroCoS 2000 [23].

En 1999, Gilles Dowek et Benjamin Werner ont montré qu'une théorie modulo avait la propriété d'élimination des coupures quand elle avait un pré-modèle, qui est un modèle multivalué dont les valeur de vérité sont des ensembles de démonstrations et ils ont mis en évidence des liens entre la propriété d'élimination de ces coupures et des propriétés combinatoires du système de réécriture. En particulier, l'élimination des coupures est toujours valide pour diverses classes de systèmes de réécriture terminants et confluents [DW99].

La connaissance de ces théories modulo qui vérifient et ne vérifient pas l'élimination des coupures a progressé dans deux direction. D'une part, Gilles Dowek et Benjamin Werner ont

[DW99] G. Dowek, B. Werner, « Proof normalization modulo », *in : Types for proofs and programs 98*, *Lecture Notes in Computer Science*, *1657*, Springer-Verlag, p. 62–77, 1999.

démontré que la terminaison et la confluence n'étaient pas des conditions suffisantes pour la cohérence et l'élimination des coupures. Le contre-exemple est présenté dans [41]. D'autre part, exploitant un résultat de Marcel Crabbé, Gilles Dowek [24] a montré que les Fondations stratifiées de Quine pouvaient être présentées comme une théorie modulo et que cette théorie avait la propriété d'élimination des coupures. Cette démonstration ouvre de nouvelles voies quant à l'exploration des liens qui existent entre l'existence d'un modèle pour une théorie modulo, et l'existence d'un pré-modèle (qui est une condition suffisante pour avoir l'élimination des coupures).

Par ailleurs, Gilles Dowek a montré que la notion de coupure modulo, généralisait la notion de coupure avec des règles de pliage et de dépliage due à Dag Prawitz [11].

6.3.7 Démonstration automatique en théorie des ensembles

Participants : Stéphane Vaillant, Gilles Dowek, Thérèse Hardin.

Mots clés : théorie des ensembles, démonstration automatique, substitutions explicites, déduction modulo.

La théorie des ensembles peut s'exprimer sous, entre autres, deux formes : au premier ordre avec des axiomes d'existence (par exemple, la théorie de Zermelo), ou avec un symbole (dit de compréhension) représentant l'ensemble des éléments d'un ensemble A qui vérifient une proposition P.

Cette seconde présentation a pour avantage une notation naturelle pour les ensembles définis par compréhension mais son inconvénient est que ce n'est plus une théorie du premier ordre car le symbole de compréhension est un lieur et il prend en argument une proposition; ce n'est donc pas un symbole de fonction du premier ordre.

De plus ces deux théories ont un ensemble infini d'axiomes; les méthodes de démonstration automatique du premier ordre ne peuvent donc pas s'appliquer directement.

Stéphane Vaillant étudie une présentation de la théorie des ensembles au premier ordre avec une syntaxe proche de celle utilisant un lieur et avec un nombre fini d'axiomes. Cette présentation a été obtenue à partir des travaux de Gilles Dowek Thérèse Hardin et Claude Kirchner pour la théorie des types : c'est un codage de la théorie avec symbole de compréhension dans un langage du premier ordre par l'intermédiaire d'une substitution explicite. La théorie obtenue est une extension conservative de la théorie de Zermelo.

Une démonstration dans ce système peut s'exprimer en déduction modulo : une partie des axiomes (ceux concernant le codage) s'exprime sous forme d'une congruence (décidable) sur les propositions, les étapes de la démonstration liées au codage sont alors supprimées et la démonstration obtenue est proche d'une démonstration obtenue dans le système avec lieur.

Le travail de Stéphane Vaillant s'oriente vers une implémentation d'un système de démonstration en déduction modulo pour la présentation étudiée. Il sera alors intéressant d'une part d'établir une comparaison entre les résultats obtenus avec cette implémentation et ceux déja obtenus en résolution avec la théorie des ensembles de Von Neumann, Bernays et Gödel (NBG), et d'autre part de comparer les deux présentations.

6.3.8 Paradoxes en théorie des types

Participants : Alexandre Miquel, Hugo Herbelin.

Mots clés : Paradoxe de Russell, théorie des ensembles, ensembles antifondés.

Alexandre Miquel a étudié les liens entre les paradoxes de la théorie des ensembles et ceux de la théorie des types. Tout comme la première théorie des ensembles — due à G. Cantor — était contradictoire (paradoxes de Burali-Forti et de Russell), plusieurs théories des types introduites au cours des années 1960-1970 se sont également révélées contradictoires (Type:Type, système U). Toutefois, contrairement à la théorie des ensembles, l'étude de ces formalismes incohérents présente un certain intérêt en raison des traits de programmation que ces formalismes proposent, en dépit de leur faiblesse logique inhérente.

Alexandre Miquel a ainsi montré qu'on pouvait encoder la théorie des ensembles de Cantor dans les systèmes U et U^- de manière à pouvoir dériver un équivalent du paradoxe de Russell en théorie des types. Ce travail se base sur un codage des ensembles antifondés en théorie des types à l'aide de graphes pointés. Alexandre Miquel a également montré que ce codage pouvait être étendu à d'autres théories des types (non paradoxales), permettant ainsi d'exprimer la puissance logique de ces systèmes de types en termes de théorie des ensembles.

Ce travail a fait l'objet d'un exposé au congrès TYPES au mois de décembre 2000 à Durham.

6.3.9 Représentation des structures non libres en théorie des types

Participant : Pierre Courtieu.

Mots clés : structure non libre, quotients, théorie des types, Calcul des constructions inductives, types inductifs, constructeurs.

Le problème posé par les structures non libres et en particulier par les quotients a été étudié dans le cadre des théories intensionnelles 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 démonstration 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 démonstrations 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.

6.4 Modularité

6.4.1 Calcul de modules au-dessus des systèmes de types purs

Participant : Judicaël Courant.

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

Le plus gros obstacle à la réutilisation des systèmes de modules à la SML pour la démonstration était jusqu'ici la difficulté de leur donner une sémantique par réduction. Il semblait en effet y avoir incompatibilité entre la possibilité de définir des types abstraits et la préservation du typage par réduction. En revenant sur les préjugés qui conduisaient à cette incompatibilité, Judicaël Courant a montré qu'on pouvait obtenir une sémantique par réduction pour un système de modules au-dessus des systèmes de types purs dans lequel on pouvait définir des types abstraits. L'étude des réductions s'est révélée non triviale, la réduction n'étant en effet pas confluente sur les termes non typés. La propriété de confluence est cependant vraie sur les termes typés. Les techniques dues à Geuvers étant inapplicables, la démonstration de cette propriéte se fait simultanément avec celle de la normalisation forte, en adaptant des techniques dues d'une part à Gallier et Coquand et d'autre part à Goguen.

L'implantation de ce calcul de modules posant encore quelques problèmes, Judicaël Courant a commencé l'implantation d'un prototype de vérificateur de types de ce système.

Judicaël Courant a publié une version anglaise de sa thèse sous forme de rapport de recherche [Cou99] ; une version journal de ce rapport est en cours de préparation. Ce travail de remise en forme de ses résultats devrait permettre, à terme, de simplifier la démonstration de normalisation ce qui devrait ouvrir la porte à la définition d'extensions de son calcul de module, 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).

6.4.2 Réécriture anonyme modulaire

Participants : Jacek Chrzaszcz, Jean-Pierre Jouannaud.

Mots clés : réécriture, modules, calcul des constructions.

Jacek Chrząszcz a étudié l'extension du système de modules anonymes de Judicaël Courant [Cou98] par des définitions par réécriture et des définitions inductives. Le concept qui est naturellement apparu, la réécriture anonyme, s'inscrit dans le cadre d'études d'objets anonymes abstraits, notamment des fonctions (le λ-calcul), des définitions inductives (le Calcul des Constructions Inductives) et récemment des modules.

Jacek Chrząszcz a élaboré un formalisme qui regroupe ces quatre composantes en utilisant la notion de spécification pour donner des types à tous les objets du calcul y compris les définitions par réécriture. L'utilisation de spécifications a également permis de résoudre la

[Cou99] J. Courant, «$\mathcal{MC}$: A module calculus for Pure Type Systems», Manuscript, juin 1999.

[Cou98] J. Courant, *Un calcul de modules pour les systèmes de types purs*, Thèse de doctorat, Ecole Normale Supérieure de Lyon, 1998.

question de la généricité des noms en donnant à l'utilisateur le droit de décider si sa définition doit être générique ou pas. Ce travail a été présenté à la conférence TYPES 2000.

6.4.3 Sémantique du sous-typage en théorie des types

Participants : Alexandre Miquel, Hugo Herbelin.

Mots clés : Calcul des Constructions, arguments implicites, sous-typage, types intersection, espaces cohérents.

Au cours de sa seconde année de thèse (encadrée par Hugo Herbelin), Alexandre Miquel s'est intéressé à la sémantique du sous-typage dans le cadre de la théorie des types. Le sous-typage est un trait des langages de programmation très utilisé dans l'approche orientée objet ainsi que dans le cadre des langages fonctionnels dans le style de ML, mais qui est encore peu étudié dans le cadre d'un formalisme tel que celui du système Coq dans lequel les preuves mathématiques sont des objets de première classe. Ce travail a abouti à la construction d'une famille de modèles de la théorie des types avec sous-typage et types intersection dans les espaces cohérents, lequel a fait l'objet d'une publication dans [34].

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 labelisé 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.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 Action VIP du GIE Dyade

Participants : Gilles Dowek, Jean Goubault, Nabil EL Khadi, Christine Paulin, Muriel Roger, Éva Rose.

L'action VIP du G.I.E. Dyade, s'intéresse à la modélisation et vérification de composants intervenants dans le commerce électronique (protocoles cryptographiques, code java certifié). Coq est utilisé comme outil pour formaliser ces démonstrations. Les principales difficultés concernent la recherche de représentations des problèmes dans Coq qui facilitent les démonstrations et la conception de techniques et d'outils de démonstrations permettant une automatisation partielle de la tâche.

8.1.2 Action de recherche concertée SJava

Participants : Jean Goubault, Benjamin Grégoire, Éva Rose.

L'action SJava à laquelle participe 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 Working Group GEMPLUS

Le Projet Verificard étudie des questions de sécurité et de sureté 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 Bull.

8.2.3 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).

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

Gilles Dowek a été membre du comité de programme de LPAR 2000, TPHOLs 2000, RTA 2000, WESTAPP 2000. Jean Goubault-Larrecq a été membre du comité de programme de Tableaux 2000. Christine Paulin a été membre du comité de programme de TPHOLs 2000 et de PPDP 2000. Benjamin Werner a été membre du comité de programme de JFLA 2000 et du workshop Logical Frameworks and Meta-Languages 2000.

9.1.2 Jurys

Gilles Dowek a été rapporteur de l'habilitation de Catherine Dubois et de la thèse de Yann Coscoy. Jean Duprat a été rapporteur de la thèse de Lionel Vintenat. Jean Goubault-Larrecq a été rapporteur de thèse de Dominique Larchey-Wendling. Christine Paulin a été rapporteur de la thèse de Cécile Dumas Canovas et de celle de Michael Franssen. Benjamin Werner a été rapporteur de la thèse de Sylvain Boulmé.

Gilles Dowek a été membre du jury de la thèse de Dominique Larchey-Wendling. Jean Goubault-Larrecq été membre du jury d'habilitation de Serenella Cerrito et du jury de thèse de Romain Guider. Christine Paulin a été membre du jury de la thèse de Sylvain Boulmé.

9.1.3 Vulgarisation scientifique

Gilles Dowek a reçu le Grand Prix d'Alembert des Lycéens. Il a publié un article dans la revue *Pour la Science* [42].

9.1.4 Visites

De janvier à août 2000, Jean-Christophe Filliâtre a effectué une formation post-doctorale (bourse INRIA) au Stanford Research Institute (Menlo Park, Californie) au sein de l'équipe PVS. En collaboration avec N. Shankar et H. Rueß, il y a conçu et développé une nouvelle procédure de décision pour le système de traitement de démonstrations mathématiques PVS. Il s'agit d'une procédure de décision complète pour les propositions sans quantificateur sur des termes du premier ordre avec égalités, inégalités linéaires et symboles de fonctions interprétés (dans des théories satisfaisant les hypothèses de l'algorithme de Shostak) ou non-interprétés.

Cette procédure de décision sera disponible dans la prochaine version du système PVS mais également distribuée comme une bibliothèque indépendante.

Benjamin Werner et Hugo Herbelin ont été invités deux semaines à Kyoto et Kobe par le professeur Susumu Hayashi dans le cadre de son projet "Proof animation". Cette visite a donné lieu à un colloque où ils ont présenté leurs travaux. Benjamin Werner a également effectué un cours de Coq à l'université de Kobe.

Micaela Mayero a séjourné une semaine chez INTEL (Hillsboro, Etats-Unis) du 10 au 17 Septembre 2000. Elle y a présenté son travail sur les nombres réels.

Gilles Dowek a été invité dix jours au Laboratoire ICASE de la NASA à Langley (États-unis).

9.1.5 Colloques.

Philippe Audebaud, Jacek Chrząszcz, Judicaël Courant, Pierre Courtieu, David Delahaye, Gilles Dowek, Jean Duprat, Jean-Christophe Filliâtre, Hugo Herbelin, Micaela Mayero, Alexandre Miquel, Christine Paulin, Daria Walukiewicz-Chrząszcz et Benjamin Werner ont participé à TYPES2000. Jean Duprat, Jacek Chrząszcz, Stéphane Vaillant, Alexandre Miquel, Gilles Dowek, David Delahaye et Micaela Mayero y ont présenté une communication. Hugo Herbelin, Jean-Christophe Filliâtre et David Delahaye ont présenté une démonstration du système Coq V7. Micaela Mayero y a organisé une réunion sur les différentes approches concernant les développements des nombres réels en Coq.

Philippe Audebaud a participé au workshop ITRS 2000 organisé dans le cadre de ICALP 2000, à Genève, en juillet 2000.

Micaela Mayero a participé au "Workshop on certified arithmetic" du 15-16 Novembre à Lyon.

David Delahaye a participé à LPAR 2000 où il a présenté une communication.

Jean Goubault-Larrecq a participé à la journée Dyade de présentation des activités de Dyade à Bull où il a présenté une communication.

Jean Goubault-Larrecq a participé au workshop international FMPPTA 2000 où il a présenté une communication invitée.

Jean Goubault-Larrecq a participé aux journées ASPROM sur la sécurité du logiciel où il a présenté une communication invitée.

Nabil El Kadhi a participé au workshop Security Language and Middleware, où il a présenté une communication,

Nabil El Kadhi a participé au workshop Mathematical Foundations of Programming Semantics (MFPS XVI), où il a présenté une communication,

Nabil El Kadhi a participé aux JDIR 2000, où il a présenté une communication,

Éva Rose a participé au workshop Formal Techniques for Java Programs de la conférence ECOOP 2000.

Hugo Herbelin a participé à la conférence ICFP 2000 à Montréal où il a présenté une communication en commun avec Pierre-Louis Curien.

Hugo Herbelin a participé à la conférence RTA 2000 et à WESTAPP 2000 où il a présenté une communication.

Stéphane Vaillant a participé à WESTAPP 2000 où il a présenté une communication.

Frédéric Blanqui a participé à RTA 2000 où il a présente une communication et à WESTAPP 2000.

Gilles Dowek a participé à RTA 2000, à WESTAPP 2000 et à LC 2000.

Gilles Dowek a participé à FroCoS 2000, où il a présenté une communication invitée.

Gilles Dowek a participé au workshop "Fondations and computations" à Edinburgh où il a présenté une communication.

Daria Walukiewicz-Chrząszcz et Jacek Chrząszcz ont participé à la EEF Foundations school in Deduction and Theorem Proving.

Daria Walukiewicz-Chrząszcz a participé au Workshop on Logical Frameworks and Metalanguages et à International Conference on Logic in Computer Science (LICS 2000).

9.1.6 Responsabilités diverses

Christine Paulin est membre du steering comittee de l'European Association for Computer Science Logic (EACSL) et du Educational European Forum. Christine Paulin a participé à un comité de qualification à des postes de professeurs à la Technishe University of Danemark. Gilles Dowek est membre du Steering Committee du working group *Types*. Gilles Dowek est président du comité des thèses de l'Association Française d'Informatique Théorique. Gilles Dowek est membre de la commission de spécialistes de l'École Normale Supérieure. Jean Duprat est membre de la commission de spécialistes de l'École Normale Supérieure de Lyon et du Conseil Scientifique de l'École Normale Supérieure de Lyon. Hugo Herbelin a géré les séminaires internes et publics du projet. Il a notamment organisé 3 journées thématiques rassemblant plus de 30 participants. Bruno Barras est conseiller scientifique chez Trusted Logic. Jean-Christophe Filliâtre est conseiller scientifique chez France Télécom.

9.2 Enseignement

Christine Paulin, Judicaël Courant et Jean-Pierre Jouannaud sont enseignants à l'Université de Paris XI. Philippe Audebaud et Jean Duprat sont enseignants à l'École Normale de Lyon.

Benjamin Werner a enseigné un cours de tronc commun *Preuves Constructives* dans le DEA Programmation. Christine Paulin et Benjamin Werner ont enseigné un cours d'option *Logiques Constructives* (20 heures) dans le DEA Programmation [PW99]. Jean Goubault-Larrecq a enseigné un cours *Démonstration automatique*, dans le DEA Programmation.

Jean Goubault-Larrecq a enseigné un cours *Analyse statique de programmes* dans le DESS Développement de Logiciels Sûrs.

Jean Goubault-Larrecq a enseigné un cours *lambda-calcul et programmation* au Magistère de Mathématiques Fondamentales et Appliquées à l'Informatique.

Gilles Dowek est professeur chargé de cours à l'École Polytechnique. Benjamin Werner est chef de travaux pratiques à l'Ecole Polytechnique.

[PW99] C. Paulin, B. Werner, *Introduction to Coq*, TYPES'99 Summer School: Theory and Practice of Formal Proofs, 1999, Notes de cours.

Benjamin Werner a donné un cours *Preuves sur ordinateur* à l'Ecole Nationale Supérieure de Techniques Avancées (ENSTA). Gilles Dowek a donné un cours *Démonstration Automatique* à l'Ecole Nationale Supérieure de Techniques Avancées (ENSTA).

Jean-Christophe Filliâtre est ATER à l'Université Paris-Sud.

Benjamin Grégoire assure des Travaux Pratiques à l'École Polytechnique.

Micaela Mayero est monitrice à l'Université Paris XIII. Micaela Mayero est demi-ATER à l'Université Paris VI.

David Delahaye est moniteur à l'université Paris VI. David Delahaye est demi ATER à l'université Pierre et Marie Curie

Frédéric Blanqui est moniteur à l'IUT d'Orsay.

Nabil El Kadhi a enseigné divers cours d'informatique à l'EPITA.

Gilles Dowek, Jean Duprat, Christine Paulin et Benjamin Werner sont intervenus à l'Ecole Jeunes Chercheurs qui s'est tenue à Lyon au printemps 2000.

Gilles Dowek encadre les travaux de thèse de Micaela Mayero et de Stéphane Vaillant. Jean Goubault-Larrecq encadre les travaux de thèse d'Éva Rose, Muriel Roger et Kumar Neeraj Verma. Jean Goubault-Larrecq et Mohammed Ben Ahmed encadrent les travaux de thèse de Nabil El Kadhi. Hugo Herbelin encadre les travaux de thèse de Alexandre Miquel. Jean-Pierre Jouannaud encadre les travaux de thèse de Frédéric Blanqui et Daria Walukiewicz-Chrząszcz. Jean-Pierre Jouannaud et Gilles Dowek encadrent les travaux de thèse de Jacek Chrząszcz. Christine Paulin encadre les travaux de thèse de Pierre Letouzey. Christine Paulin et Pierre Crégut encadrent les travaux de thèse de Cuihtlauac Alvarado. Benjamin Werner encadre les travaux de thèse de David Delahaye. Benjamin Werner et Xavier Leroy encadrent les travaux de thèse de Benjamin Grégoire.

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 Automaqique, 1998.

[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 a nd Properties», *in : Proceedings of the conference Typed Lambda Calculi a nd 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.

Articles et chapitres de livre

[11] G. Dowek, «About folding-unfolding cuts and cuts modulo», *Journal of Logic and Computation*, À paraître, Rapport de Recherche 4004, Institut National de Recherche en Informatique et en Automatique, 2000.

[12] G. Dowek, T. Hardin, C. Kirchner, «Higher-order unification via explicit substitutions», *Information and Computation 157*, 2000, p. 183–235.

[13] 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.

[14] J.-C. Filliâtre, « Verification of Non-Functional Programs using Interpretations in Type Theory », *Journal of Functional Programming*, À paraître, `http://www.lri.fr/~~filliatr/ftp/publis/jphd.ps.gz`.

[15] L. Moreau, J. Duprat, «A Construction of Distributed Reference Counting», *Acta Informatica*, Á paraître, Rapport de Recherche LIP 1999-18.

[16] L. Moreau, J. Duprat, «A Construction of Distributed Reference Counting», *Acta Informatica*, À paraître, RR LIP 1999-18.

Communications à des congrès, colloques, etc.

[17] F. Blanqui, « Termination and confluence of higher-order rewrite systems », *in : Rewriting Techniques and Applications*, L. Bachmair (éditeur), *Lecture Notes in Computer Science, 1833*, p. 47–61, 2000.

[18] P.-L. Curien, H. Herbelin, « The Duality of Computation », *in : 5th ACM SIGPLAN International Conference on Functional Programming (ICFP 2000)*, p. 233–243, Montréal, September 2000.

[19] D. Delahaye, « Information retrieval in a Coq proof library using type isomorphisms », *in : TYPES'99*, T. Coquand, P. Dybjer, B. Nordström, J. Smith (éditeurs), *Lecture Notes in Computer Science*, Springer-Verlag, À paraître.

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

[21] D. Delahaye, «A Tactic Language for the System Coq», *in : Logic for Programming and Automated Reasoning*, Springer-Verlag, 2000, `ftp://ftp.inria.fr/INRIA/Projects/coq/David.Delahaye/ltac/LPAR2000-tac-def.ps`.

[22] G. Dowek, «Automated theorem proving in first-order logic modulo: on the difference between type theory and set theory», *in : Automated Deduction in Classical and Non-Classical Logics*, *Lecture Notes in Artificial Intelligence*, *1761*, Springer-Verlag, p. 1–22, 2000.

[23] G. Dowek, « Axioms vs. rewrite rules: from completeness to cut elimination», *in : Frontiers of Combining Systems*, *Lecture Notes in Artificial Intelligence*, *1794*, Springer-Verlag, p. 62–72, 2000.

[24] G. Dowek, « The Stratified Foundations as a theory modulo», *in : Typed Lambda Calculus and Applications*, 2001. À paraître.

[25] N. El Kadhi, « Static Analysis for Cryptographic Properties Verification », *in : MFPS XVI Workshop*, Hoboken, NJ, USA, avril 2000.

[26] N. El Kadhi, «Static Secret Properties Verification», *in : Security Languages and Middleware*, Stockholm, Suède, avril 2000.

[27] N. El Kadhi, «Vers une vérification statique des applets cryptographiques», *in : Communication aux JDIR 2000*, novembre 2000.

[28] J. Goubault-Larrecq, É. Goubault, «Order-Theoretic, Geometric and Combinatorial Models of Intuitionistic S4 Proofs», *in : 1st Workshop on Intuitionistic Modal Logics and Applications , Trento, Italie, et 1st Workshop on Geometric Methods in Concurrency Theory, Aalborg, Danemark*.

[29] J. Goubault-Larrecq, « A Method for Automatic Cryptographic Protocol Verification (Extended Abstract) », *in : Workshop on Formal Methods in Parallel Programming, Theory and Applications (FMPPTA 2000)*, *Lecture Notes in Computer Science*, Springer Verlag, p. 977–984, 2000.

[30] H. Herbelin, « Explicit Substitutions and Reducibility », *in : 3rd international Workshop on Explicit Substitutions, Theory and Applications to Programs and Proofs (WESTAPP 2000)*, p. 53–69, Norwich (UK), July 2000.

[31] P. Letouzey, L. Théry, «Formalizing Stålmarck's algorithm in Coq», *in : Theorem Proving in Higher Order Logics: 13th International Conference, TPHOLs 2000*, J. Harrison, M. Aagaard (éditeurs), *Lecture Notes in Computer Science*, *1869*, Springer-Verlag, p. 387–404, 2000, `http://www.eleves.ens.fr/home/letouzey/download/stalmarck.ps.gz`.

[32] M. Mayero, «The Three Gap Theorem (Steinhaus Conjecture)», *in : TYPES'99*, T. Coquand, P. Dybjer, B. Nordström, J. Smith (éditeurs), À paraître.

[33] A. Miquel, «Affichage générique d'arbres à l'aide de la géométrie hyperbolique», *in : Actes des Journées francophones des langages applicatifs (JFLA 2000)*, p. 49–62, 2000.

[34] A. Miquel, « A Model for Impredicative Type Systems with Universes, Intersection Types and Subtyping», *in : Proceedings of the 15 th Annual IEEE Symposium on Logic in Computer Science (LICS'00)*, 2000.

[35] E. Rose, K. H. Rose, « Java Access Protection Through Typing », *in : ECOOP 2000 Workshop on Formal Techniques for Java Programs*, 2000. À paraître in Concurrency: Practice and Experience.

[36] K. N. Verma, J. Goubault-Larrecq, S. Prasad, S. Arun-Kumar, « Reflecting BDDs in Coq », *in : 6th Asian Computing Science Conference (ASIAN 2000)*, Springer Verlag, Penang, Malaysia, novembre 2000.

[37] D. Walukiewicz-Chrzaszcz, « Termination of rewriting in the Calculus of Constructions », *in : Workshop on Logical Frameworks and Meta-languages, Santa Barbara, California*, 2000. Part of the LICS 2000.

Rapports de recherche et publications internes

[38] B. Bérard, P. Castéran, E. Fleury, L. Fribourg, J. F. Monin, C. Paulin, A. Petit, D. Rouillard, « Automates temporisés CALIFE », *Fourniture F1.1*, Calife, 2000.

[39] P. Castéran, E. Freund, C. Paulin, D. Rouillard, « Bibliothèques Coq et Isabelle-HOL pour les systèmes de transitions et les p-automates », *Fourniture F5.4*, Calife, 2000.

[40] E. Freund, C. Paulin, « Modélisation d'automates temporisés dans Coq », *chapitre de [39]*, 2000.

Divers

[41] G. Dowek, B. Werner, « An inconsistent theory modulo defined by a confluent and terminating rewrite system », Manuscript, 2000, `http://logical.inria.fr/~~dowek/Publi/counterexample.ps.gz`.

[42] G. Dowek, « L'infini et l'univers des algorithmes », *in : Les Infinis, Pour la Science*, Décembre 2000.

[43] J. Goubault-Larrecq, É. Goubault, « On Intuitionistic S4 and Augmented Simplicial Sets », Soumis à publication, `http://www.dyade.fr/fr/actions/vip/jgl/top.ps.gz`.

[44] P. Letouzey, *Exécution de termes de preuves: une nouvelle méthode d'extraction pour le Calcul des Constructions Inductives*, Rapport de dea, Université Paris VI, 2000, `http://www.eleves.ens.fr/home/letouzey/download/rapport_dea.ps.gz`.

[45] P. Letouzey, « Extraction dans le Calcul des Constructions Inductives », Manuscript, 2000.

[46] X. Rival, *Implémentation d'une librairie d'automates d'arbres en Coq*, Stage de maîtrise, École Normale Supérieure, septembre 2000.