INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet Lemme

Logiciels et Mathématiques

Sophia Antipolis

THÈME 2A

Table des matières

1 Composition de l'équipe

Responsable scientifique

Loïc Pottier [Chargé de recherche INRIA]

Responsable permanent

Yves Bertot [Chargé de recherche INRIA]

Assistante de projet

Nathalie Bellesso

Personnel Inria

Francis Montagnac [Ingénieur de Recherche INRIA]

Laurence Rideau [Chargée de recherche INRIA]

Laurent Théry [Chargé de recherche INRIA]

Collaborateurs extérieurs

Frédérique Guilhot [Professeur agrégé]

André Hirschowitz [Professeur UNSA, Laboratoire J.A.Dieudonné]

Roger Marlin [Professeur UNSA, Laboratoire J.A.Dieudonné]

Monica Nesi [Maître de Conférences, Université de L'Aquila, Italie]

Olivier Pons [Post-doctorant, université du Minho, Portugal]

Ingénieurs experts

Pascal Lequang [jusqu'à mars]

Claude Pasquier [depuis juillet 99]

Ingénieur en poste d'accueil-jeune

Ahmed Amerkad [depuis septembre]

Chercheurs doctorants

Antonia Balaa [bourse INRIA]

Laurent Chicli [bourse INRIA]

Stéphane Lavirotte [bourse MESR, jusqu'en juin]

Hanane Naciri [bourse MESR]

Nicolas Magaud [bourse MESR, depuis septembre]

Venanzio Capretta [université Niemegen, septembre-octobre]

Stagiaires

Marc Chiaverini [Maîtrise MIM UNSA, avril-mai]

Michel Hirschowitz [Maîtrise MIM UNSA, avril-mai, juillet]

François-Régis Sinot [1ère année ENS Lyon, juin-juillet]

Sébastien Bardin [IIE, CNAM, juin-août]

David Pichardie [1ère année ENS Cachan-Bretagne juin-août]

2 Présentation et objectifs généraux

Le projet Lemme a été créé fin 1999, après un an de vie sous forme d'un avant-projet. L'an 2000 a été marqué par :

- la mise au point et la diffusion de la première version de Pcoq, environnement de preuves en Coq, héritier direct de Ctcoq, et implanté en Java, à partir des outils Aïoli et Figue, qui reprennent de nombreuses idées et résultats du système Centaur. Pcoq continue à être amélioré, grâce à un poste d'accueil qui lui est consacré;
- le commencement de l'action de recherche coopérative AOC (Algorithmique des ordinateurs certifiée), dans laquelle le projet joue un rôle moteur;
- l'acceptation du projet européen Vérificard, vaste projet de spécification et de vérification formelles de code Javacard, dans lequel Lemme est associé localement avec le projet Oasis;
- la mise sur pied d'une collaboration prometteuse avec la société Gem+ sur l'application des méthodes de sémantique des langages à l'aide de Coq et Pcoq au code Javacard de cartes à puces;
- la mise au point de plusieurs outils et tactiques spécialisées en Coq, qui répondent à des besoins que nous avions ressentis, et qui devraient compléter ce système;

– l'extension de l'afficheur Figue pour afficher des formules mathématiques, en s'appuyant sur la spécification MathML pour définir les constructeurs à implémenter et leurs attributs.

Les directions de recherche du projet ont légèrement évolué cette année.

La direction *environnements de preuve* a nécessité pas mal d'énergie, pour la mise au point de la première version distribuée de Pcoq.

La direction *formalisation des mathématiques* s'est un peu tassée, de par la fin de l'action CFC, et parce que la collaboration entamée avec des enseignants de l'université s'est un peu mise en veille, surtout parce ce que nos utils ne sont malheureusement pas encore prêts à être mis dans les mains des étudiants du premier cycle.

Par contre la direction *algorithmique certifiée* a continué à se développer, en particulier grâce à l'action de recherche coopérative AOC. L'arithmétique des ordinateurs est un domaine d'application motivant et riche en problèmes intéressants qui peuvent être traités par les méthodes formelles.

Enfin, le thème de la sémantique des langages a pris une importance nouvelle, avec des applications à la spécification et la vérification de code Java. Cela correspond à une demande forte de l'environnement local et international, mais reste cohérent avec les buts du projet car ces travaux d'une part s'appuient sur les résultats du projet (environnement de preuves, tactiques spécialisées, fonctions récursives bien fondées), et d'autre part fournissent des exemples significatifs pour tester certaines de nos idées.

3 Fondements scientifiques

3.1 Environnements de preuves

Mots clés : preuve, environnement, interface homme-machine.

Le but de ce thème est d'étudier les outils mécaniques de recherche et de vérification de preuves pour faciliter leur utilisation par des ingénieurs et des mathématiciens dans la production de logiciels et de théories mathématiques formelles.

Deux objectifs complémentaires apparaissent. Le premier est d'améliorer les moyens d'interaction entre l'ordinateur et l'utilisateur, afin de rendre les outils accessibles à des ingénieurs peu intéressés par la théorie de la preuve et à des mathématiciens peu intéressés par la programmation et les contraintes formelles. Le second objectif est de faciliter la maintenance de grands développements de mathématiques formelles, afin d'aider leur réutilisation et leur adaptation à des contextes variés.

Ainsi, nous voulons augmenter la pénétration des méthodes formelles dans le développement des logiciels à la fois en facilitant leur usage par des débutants et en augmentant la productivité des utilisateurs confirmés.

3.2 Formalisation de théories mathématiques

Mots clés : formalisation, mathématiques.

Le but de ce thème est d'étudier comment des théories mathématiques où interviennent de

nombreux types d'objets peuvent être représentées dans le calcul des constructions inductives (CCI en abrégé), de façon à être aussi lisibles et utilisables que possible par quelqu'un qui en a une connaissance scolaire ou académique.

Ce thème est très lié aux autres, en ce sens qu'il doit fournir des fondements sémantiques aux objets des environnements de preuves, et des bibliothèques d'objets, lemmes et théorèmes de base pour alléger le travail de preuve des algorithmes mathématiques, dont la preuve nécessite souvent de connaître et de manipuler de larges pans des mathématiques classiques.

Le CCI est assez puissant pour formaliser des mathématiques complexes : structures algébriques avec leurs dépendances et leurs opérations (comme le système de calcul formel Axiom l'a fait), mais aussi, et surtout puisqu'on veut faire des démonstrations, toutes leurs propriétés logiques, ce qui reste très marginal dans les systèmes de calcul scientifique. Le CCI permet aussi d'écrire des algorithmes, sous forme de programmes fonctionnels récursifs, avec des structures de données riches. Enfin, avec l'isomorphisme de Curry-Howard, le CCI est un langage de manipulation des preuves mathématiques. Mais ce langage est difficile d'accès, un peu comme un assembleur, bien que sa concision et sa puissance le rendent très élégant pour l'initié.

Le système Coq, les interfaces CtCoq et Pcoq offrent des outils qui permettent souvent une formalisation satisfaisante : les *"record"*, les *coercions*, le *pretty-printer* PPML, la syntaxe extensible de Coq par exemple. Mais de nombreux problèmes restent à résoudre : héritage multiple, complexité de l'empilement des coercions et du typage entre structures algébriques, manipulation difficile des types dépendants dans les preuves et les constructions d'objets, absence de sous-typage et de quotient dans le CCI.

3.3 Implémentations certifiées d'algorithmes de calcul scientifique

Mots clés : algorithme, certification.

Pour obtenir des programmes certifiés, nous proposons une approche inverse de celle généralement utilisée : plutôt que de chercher à prouver des propriétés d'un programme existant (en formalisant sa sémantique), nous proposons de produire des programmes dont la correction découle de celle de leur processus de création.

Par exemple, pour obtenir un programme de calcul de bases de Gröbner, on commence par écrire l'algorithme de Buchberger dans le CCI, puis on prouve en `Coq` qu'il calcule bien une base de Gröbner, enfin on utilise le processus d'extraction automatique de `Coq` pour produire un programme `Ocaml`, dont on sait qu'il calcule la même chose, puisque ceci est garanti par les propriétés du CCI et de son affaiblissement dans la théorie des types ML. Le programme extrait est efficace, comme on pouvait le prévoir, et comme le montre l'expérience.

Cette approche a plusieurs avantages : on travaille à un haut niveau d'abstraction, en restant indépendant du langage d'implantation final (mais plus ce langage sera fonctionnel, plus l'implantation sera efficace). On peut aussi se concentrer sur les caractéristiques propres à l'algorithme qu'on étudie, en faisant abstraction du reste (gestion de la mémoire, choix d'implantation des données). Le niveau de certification peut être ainsi variable, et paramétrable (par exemple pour le calcul de bases de Gröbner, on peut supposer qu'une implantation de l'arithmétique des polynômes est donnée).

Mais elle pose aussi quelques problèmes qui, sans être bloquants, s'avèrent difficiles à ré-

soudre. Par exemple il est difficile actuellement de faire des preuves sur des algorithmes récursifs dont la terminaison découle d'un ordre bien fondé. Il est aussi délicat de travailler dans un style impératif, ou avec des opérations qui font des effets de bord (opérations en place sur les données par exemple).

3.4 Sémantique des langages de programmation

Mots clés : sémantique, langages, programmation, Java, JavaCard, Coq.

Les algorithmes intervenant dans l'implantation des langages de programmation font également partie de notre champ d'investigation. Pour ces algorithmes, on se repose généralement sur la description sémantique d'un langage, et les propriétés que l'on cherche à établir pour un algorithme sont soit qu'il préserve la sémantique des programmes (s'il s'agit d'un algorithme de transformation ou d'optimisation) soit que les programmes qu'il produit sont exempts de certains comportements indésirables (s'il s'agit d'un compilateur ou d'un vérificateur de programmes). La complexité qui apparait dans la certification de ces algorithmes réside parfois dans la taille de la description du langage, parfois dans une complexité intrinsèque de l'algorithme, qui peut reposer sur des parcours de graphes, de l'unification, etc. Pour classifier ce type d'algorithmes et de vérification, nous parlons de preuves en sémantique des langages de programmation. Outre son intérêt intrinsèque, le travail sur ce domaine complète également notre travail sur les algorithmes, car il permet de maîtriser la correction et l'efficacité des algorithmes certifiés en réfléchissant également sur leur implantation au travers des langages de programmation.

4 Domaines d'applications

4.1 Cartes à puces

Les langages utilisés dans les cartes à puces, ainsi que les programmes écrits dans ces langages peuvent faire l'objet de traitements formels. Dans le cas particulier de l'entreprise Gem+, nous avons étudié la preuve d'un mini-vérificateur de byte-code pour un langage abstrait concernant l'initialisation, ce qui a conduit à un vérificateur de byte-code certifié qui peut être traduit en un programme `Ocaml` grâce au mécanisme d'extraction de Coq. Ce travail a été effectué avec Coq et Ctcoq, et met en oeuvre notre savoir-faire en sémantique et développement de grandes preuves en Coq dans l'environnement Ctcoq. Il va se poursuivre d'une part dans la collaboration avec Gem+, et, d'autre part, dans le cadre du projet européen VERIFICARD qui commence.

4.2 Algorithmes certifiés

Dans certains domaines, il est crucial que les algorithmes et les programmes mis en oeuvre soient absolument sûrs. C'est le cas par exemple dans les environnements permettant de faire de la preuve formelle. Dans ce domaine, à la suite de contacts avec la société Prover technology, l'implémentation certifiée de l'algorithme de Stålmarck faite dans le projet a été améliorée pour pouvoir traiter des cas significatifs. C'est aussi le cas pour l'arithmétique flottante; dans

le cadre de l'action coopérative AOC, nous avons formalisé la norme IEEE754, et, avec Paul Zimmermann, effectué la preuve d'un algorithme de calcul de racine carrée. Enfin, dans une collaboration récemment mise en place avec Alcatel, nous avons commencé à étudier des programmes de calcul d'attitude de satellites à base de quaternions.

4.3 Web, MathML, XML

Nos travaux autour de XML et MathML dans le cadre de Figue, ont deux principales retombées applicatives :

- d'une part ceci devrait nous permettre de partager des preuves (script Coq) sur le Web en générant depuis notre interface PCoq, des sources XML+MathML qui pourront être ensuite affichés (et imprimés) avec de "vraies" formules mathématiques par des navigateurs acceptant du MathML, et ceci indépendamment de Pcoq.

- d'autre part, dans la suite des travaux de Loïc Pottier sur Wims (WWW Interactive Methematics Server, developpé par XIAO Gang à l'université de Nice), nous voulons expérimenter l'interaction pour les formules mathématiques à travers le Web, afin de construire directement les preuves sur le Web.

Cette migration vers le Web devrait, à terme, élargir la visibilité de nos travaux, et nous permettre d'atteindre une plus grande communauté d'utilisateurs afin de valider nos outils.

5 Logiciels

5.1 Pcoq

Participants : Ahmed Amerkad, Yves Bertot [correspondant], Pascal Lequang, Loïc Pottier, Laurence Rideau.

La version 1.0 de Pcoq a été rendue disponible en février :
`http://www-sop.inria.fr/lemme/pcoq/pcoq-fra.html`. Pcoq fournit un environnement de travail pour le système de preuve Coq. Il a été développé en suivant une approche générale pour la construction d'interfaces utilisateurs pour les assistants de preuves. Il réunit les caractéristiques suivantes :

- Interface graphique : des polices de caractères multiples et des couleurs sont utilisées pour afficher les formules mathématiques et les commandes.

- Séparation entre l'interface et le système de preuve : l'interface graphique et Coq sont deux processus indépendants. Les utilisateurs peuvent choisir de faire tourner l'un des processus sur une autre machine accessible sur le réseau.

- Des mécanismes d'édition et de présentation structurées : l'environnement fournit des moyens pour éditer les formules en respectant leur structure. De nouvelles notations peuvent être ajoutées facilement.

- *Proof by pointing* : l'environnement utilise la structure des formules logiques pour aider l'utilisateur à effectuer les étapes du raisonnement en les désignant à la souris.
- Écrit en Java et en `Ocaml`, Pcoq bénéficie de leurs portabilités combinées.
- Langue naturelle : les preuves en cours de développement peuvent être visualisées sous forme d'un texte en français ou en anglais, sur lequel les mécanismes de *proof-by-pointing* fonctionnent.

Pcoq continue à être développé, en particulier grâce au contrat sur poste d'accueil-jeune d'Ahmed Amerkad.

5.2 Ctcoq

Participants : Yves Bertot [correspondant], Olivier Pons, Laurence Rideau.

Ctcoq (`http://www-sop.inria.fr/croap/ctcoq/ctcoq-fra.html`) continue à être distribué et à suivre les différentes versions de Coq, mais est appelé à disparaître, au profit de Pcoq.

5.3 Aïoli et Figue

Participants : Hanane Naciri, Claude Pasquier, Laurence Rideau, Laurent Théry [correspondant].

Aïoli (`http://www-sop.inria.fr/croap/aioli/doc/aioli.html`) et Figue (`http://www-sop.inria.fr/croap/figue/`) sont des briques de base de Pcoq, et à ce titre ont reçu des améliorations: sélections multiples, nouveaux combinateurs 2D (matrices, crochets, fractions, racines n-ièmes).

5.4 Stålmarck

Participants : Pierre Letouzey, Laurent Théry [correspondant].

A l'adresse (`http://www-sop.inria.fr/lemme/stalmarck/index.html`), on peut tester une implémentation certifiée de l'algorithme de Stålmarck (détection de tautologies dans les formules booléennes).

6 Résultats nouveaux

6.1 Outils pour les environnements de preuve

6.1.1 Reconnaissance de formules mathématiques

Participants : Stéphane Lavirotte, Loïc Pottier.

En juin, Stéphane Lavirotte a soutenu sa thèse sur la reconnaissance de formules. Les nouveautés de son travail par rapport à l'année passée concernent la connection à Maple

via le protocole OpenMath de son prototype de reconnaisseur de formules mathématiques typographiées et manuscrites.

6.1.2 Preuves textuelles

Participants : Yann Coscoy, Loïc Pottier.

Yann Coscoy a soutenu sa thèse en septembre, sur la traduction de preuves en langue naturelle. Une partie de son travail est intégrée au système Coq. Dans le même esprit, mais avec des méthodes différentes, Loïc Pottier a développé une traduction textuelle des preuves Coq, compatible avec les fonctionnalités de Pcoq : Pcoq permet maintenant d'afficher la preuve en cours en langue naturelle (français ou anglais). Il est possible d'agir sur cette preuve textuelle à la souris pour exécuter des tactiques (*proof-by-pointing*), ainsi que de contracter/développer des parties de la preuve. Un soin particulier a été apporté à la présentation des preuves par égalités successives. De nombreuses améliorations sont encore à apporter, mais il est tout de même possible de développer une preuve raisonnable uniquement dans ce mode textuel; le but étant d'obtenir un environnement intuitif pour Coq, utilisable dans l'enseignement.

6.1.3 XML et Édition Structurée

Mots clés : XML, XSLT, DOM, édition structurée, javabeans.

Participant : Claude Pasquier.

L'étude de l'intégration des standards de manipulation de données structurées dans les modules Figue et Aïoli a été poursuivie durant cette année dans le cadre de l'action Dyade Koala.

Intégration de DOM dans le module VTP (Virtual Tree Processor) Notre package de manipulation d'arbres a été transformé en un module abstrait. Le VTP n'est plus désormais qu'une implémentation particulière de ce module, DOM (Document Object Model) en constituant une autre. Il suffit donc de choisir l'implémentation désirée lors de l'utilisation. Ce développement a été intégré à l'outil SmartTools.

Import/Export au format XML Une représentation XML a été développée pour toutes les données manipulées dans Aïoli et dans Figue. Ainsi, l'arbre de syntaxe abstraite, les boîtes Figue, la déclaration PPML et les chemins de sélection et de modification disposent tous d'un module d'import/export en XML. Les outils ont été modifiés pour travailler directement sur les données XML. Cela a conduit à l'élaboration du module XPPML, une évolution du processeur PPML, qui permet de transformer un document XML en un document résultat en fonction de régles de transformations, exprimées elles aussi en XML.

Expérimentation de l'utilisation d'Aïoli dans un environnement distribué Le fait que toutes les données soient sérialisables dans un format standard permet d'envisager l'utilisation d'Aïoli dans un environnement réparti. Un prototype d'une telle application a été déve-

loppé. Le principe de fonctionnement retenu est que les interactions effectuées par l'utilisateur sur une représentation concrète ne sont pas directement répercutées sur le document. Elle sont transmises à un serveur qui, après avoir exécuté les actions nécessaires, informe chaque vue des modifications à effectuer. Les données transitant sur le réseau ne concernent pas des documents complets mais un encodage en XML des opérations effectuées ou à effectuer. Un article écrit sur ce sujet a été présenté au Workshop XOT2000. De plus amples informations peuvent être retrouvées sur `http://www.disi.unige.it/conferences/xot2000/xot2000.html`.

Le principe d'échange de données XML tel qu'il a été défini dans cette application a été utilisé comme base pour l'élaboration du protocole de communication inter modules dans l'outil SmartTools.

Générateur d'éditeur spécialisé de document XML Nous avons commencé le développement d'un outil qui facilite la construction d'éditeurs dédiés à un format XML donné. Notre outil permet de fournir aux utilisateurs un moyen de définir des éditeurs comme une simple transformation XSLT décrivant la manière de présenter les documents XML avec des *beans* Java. Nous étendons ensuite automatiquement cette transformation pour assurer que toute modification des objets java sera repercutée sur le document XML. Nous fournissons une bibliothèque de *beans* prêts à être utilisés, cependant, notre architecture est suffisamment ouverte pour permettre l'intégration de n'importe quel composant Java, graphique ou non. Notre application repose sur le processeur XSLT Xalan disponible en "open sources" et sur le format XML de configuration de *beans* Java défini par IBM. Une première diffusion de cet outil devrait se faire au début de l'année 2001.

6.1.4 Environnement de preuve

Participants : Ahmed Amerkad, Yves Bertot, Pascal Lequang, Loïc Pottier, Laurence Rideau.

Nous continuons à maintenir et à améliorer l'interface graphique que nous avons développée pour le système de preuve Coq. Le progrès notable réalisé en 2000 est la diffusion d'une interface construite en Java. Cette interface est utilisable aussi bien sous Unix que dans l'environnement commercial Windows. En accédant ainsi à un environnement grand public, nous espérons élargir la communauté d'utilisateurs, ce qui devrait permettre de tirer de meilleurs enseignements sur les besoins d'utilisateurs de systèmes de preuve et la validité des solutions que nous proposons.

6.1.5 Utilisation de graphes en Coq

Participants : Yves Bertot, Olivier Pons.

Dans son travail de thèse, Olivier Pons a montré qu'il était possible d'utiliser des outils d'affichage de graphes pour améliorer la compréhension de développements formels. Nous avons approfondi cette question, en nous intéressant tout particulièrement aux outils qui permettent de réduire la taille des graphes obtenus pour en rendre la compréhension possible. Le résultat est un module utilisable en Coq qui permet de produire différents types de graphes reliés à

une théorie Coq. Le format de sortie de ce module est adapté pour être utilisé avec les outils d'affichage de graphes reconnus du domaine. Nous diffusons ce module par Internet. Ce travail est décrit dans [11].

6.1.6 Interaction homme machine dans les environnements de preuves

Participants : Hanane Naciri, Laurence Rideau.

Ce travail se place dans le cadre du développement d'outils pour l'interaction homme machine dans les environnements de démonstrations mathématiques. Nous continuons à étendre et à améliorer notre outil d'affichage bidimensionnel, incrémental et interactif Figue. Cet outil permet de manipuler dynamiquement les objets structurés représentant des documents comme des programmes ou des formules mathématiques. Notre but est de présenter les objets structurés, et en particulier les formules mathématiques, de façon conviviale et d'offrir des interactions variées à la souris, comme la sélection de sous-expressions afin de pouvoir les manipuler dynamiquement (évaluation, simplification, modification, génération de code, etc...). Nous étudions le problème lié à la diversité des objets à manipuler : du texte simple, des formules mathématiques pour les systèmes de preuves ou de calcul formel, des images. Des problèmes particuliers se posent pour les formules mathématiques qui ont une structure complexe et bidimensionnelle (matrices, intégrales, indices, ...).

L'architecture extensible de Figue nous a permis d'introduire de nouveaux constructeurs graphiques mathématiques (*Racine*, *Fraction Puissance*, *Matrice*,...) qui formatent et affichent dynamiquement la formule mathématique correspondante. Pour ajouter ces constructeurs mathématiques, nous avons choisi de suivre les recommandations du standard MathML (format basé sur le langage XML pour la représentation des formules mathématiques sur le Web `http://www.w3.org/Math/`).

Pour nous conformer au standard MathML dans le cadre des environnements de preuve, nous développons une interface dans Figue permettant d'interpréter et de générer du MathML. Cette interface s'appuie sur les travaux de Claude Pasquier sur le support de XML dans les environnements de preuves. Nous assurons un lien entre les objets MathML affichés et leur structure syntaxique, ce qui offre la possibilité d'interagir avec les formules mathématiques et de les manipuler hors du contexte graphique du document. Ces derniers travaux représentent une base pour un développement futur d'un éditeur MathML. Ces travaux ont fait l'objet d'une communication à la conférence Cari'2000 [7].

Tous ces travaux ont éte intégrés à Pcoq.

D'autre part, nous avons passé du temps à comprendre en détails l'algorithme de formatage de Figue, pour corriger des problèmes désagréables à l'usage : coupures mal faites qui engendraient des lignes trop larges.

6.2 Formalisation de théories mathématiques

6.2.1 Formalisation de nouvelles structures mathématiques en Coq

Participant : Laurent Chicli.

Nous avons continué le travail entrepris l'an dernier de formalisation de différentes no-

tions de géométrie algébrique en Coq : théorie des faisceaux et algèbre ont été développés et permettent maintenant la définition des schémas affines. On trouvera un index des notions formalisées à l'adresse : `http://www-sop.inria.fr/lemme/personnel/Laurent.Chicli/FrCoq.html`. La suite de ce travail concernera tout naturellement la définition des schémas en toute généralité et de leurs propriétés.

6.2.2 Formalisation d'une méthode d'Horace en Coq

Participant : Laurent Chicli.

En s'inspirant de la thèse de Thierry Mignon "Systèmes linéaires de courbes planes", nous avons développé une axiomatisation permettant de formaliser le lemme d'Horace. Il s'agit de savoir, étant donné un entier d et une liste de multiplicités $\{m_1, \ldots, m_r\}$, quelle est la dimension de l'espace vectoriel des courbes de degré d passant par des points génériques P_i du plan avec la multiplicité m_i. Le lemme d'Horace (dont l'application demande un choix judicieux de courbes du plan) réduit le problème à un autre de même nature mais ayant un système de multiplicités plus faible, et permet donc de répondre à cette question. En effet, il est souvent possible de l'appliquer plusieurs fois pour aboutir à des systèmes résiduels bien connus de degré très faible. Notre développement a permis de tester les systèmes étudiés dans le rapport de stage d'un élève de maitrise : 40% des raisonnements étaient faux car ils appliquaient le lemme d'Horace à des configurations non valides.

6.2.3 Formalisation des complexes en Coq

Participant : Frédérique Guilhot.

En s'inspirant du cours d'algèbre d'André Hirschowitz en DEUG à l'UNSA, nous avons formalisé en Coq les bases de la théorie des nombres complexes, en allant jusqu'aux racines de l'unité. Nous pensons utiliser cela dans une preuve de la transformée de Fourier discrète. Comme l'année passée, l'accent est mis sur la lisibilité des notations mathématiques. Ce travail est effectué sous Ctcoq, mais devrait passer avant la fin de l'année sous Pcoq.

6.2.4 Quotients

Participant : Loïc Pottier.

La notion d'ensemble quotient n'est pas encore bien comprise en théorie des types. La représentation par des sétoïdes est acceptable, mais devient très vite lourde dans de grands développements. Le problème a deux causes : on perd la souplesse des réécritures avec l'égalité de Leibnitz et la définition des fonctions devient inutilement complexe. Pour résoudre ces problèmes, on propose une notion de quotients plus puissante que celle communément admise en théorie des types [17], et qui a la vertu d'être raisonnablement compatible avec la notion de fonction en théorie des types. Nous comptons reprendre le développement d'algèbre effectué dans ce cadre.

6.2.5 Extraction

Participant : Loïc Pottier.

Le travail effectué en 1998 sur l'extraction de la sorte Type en Coq a été repris et amélioré, donnant lieu à une commande Coq permettant d'extraire n'importe quel terme de Coq en un programme `Ocaml` correct et efficace. Nous avons utilisé en particulier une idée de Christine Paulin qui consiste à effondrer les termes logiques en une unique constante. Ce travail est décrit dans [16].

6.2.6 Preuves et tactiques sur les réels

Participants : Marc Chiaverini, Michel Hirschowitz, Loïc Pottier, François-Régis Sinot.

Nous avons programmé en Coq une nouvelle tactique de preuve sur les réels utilisant la méthode de Fourier-Motskin, permettant de prouver des équations et inéquations linéaires, avec éventuellement des valeurs absolues. Elle est disponible à l'adresse
`http://www-sop.inria.fr/lemme/Loic.Pottier/Coq/Fourier.tar`.

Poursuivant dans cette direction, nous nous sommes attaqués, avec le stage de François-Régis Sinot, au cas non-linéaire, en reprenant une méthode d'élimination des quantificateurs due à Kreisel et Krivine, et déjà testée par Harrison en HOL. Cela a conduit à un programme `Ocaml` permettant de résoudre des problèmes simples en degré 2. Mais la complexité de l'algorithme reste lourde et des améliorations sont encore à trouver si on veut traiter des cas non triviaux. L'avantage de la méthode est de ne se baser que sur des théorèmes simples (Rolle, valeurs intermédiaires), qui ont été prouvés en Coq par Marc Chiaverini et Michel Hirshowitz durant leur projet de maîtrise. Du coup, nous devons pouvoir reconstruire une preuve Coq (sans doute grosse...) à partir de la trace des calculs en `Ocaml`.

6.3 Certification d'algorithmes

6.3.1 Algorithme de Stålmarck en Coq

Mots clés : formules booléennes, vérification de programme, Coq.

Participant : Laurent Théry.

L'algorithme de Stålmarck est un des plus efficaces pour déterminer si une formule booléenne est une tautologie. En 1998, lors de son stage de première année de magistère, Pierre Letouzey avait prouvé la correction de l'algorithme de base et en avait dérivé une première implantation certifiée. Cette implantation étant peu efficace, un ensemble d'optimisations permettant d'appliquer la nouvelle implantation à des problèmes de taille conséquente (plus de 500 connecteurs logiques) a été validé.

Ce travail a aussi permis de développer à l'intérieur de Coq une tactique qui se sert de l'algorithme de Stålmarck pour prouver des tautologies. La solution d'interpréter directement l'algorithme dans Coq ayant des problèmes de performance, l'algorithme a été divisé en une première phase de recherche et une seconde de vérification. La première phase est effectuée en dehors de Coq et retourne en cas de succès une trace d'exécution. L'interprétation de cette

trace d'exécution nécessitant moins de calculs est effectuée dans Coq et permet d'accepter la tautologie en toute sûreté.

Ce travail a donné lieu à une publication dans les actes de la conférence TPHOL [6]. Les résultats de ce travail sont accessibles par le web à l'adresse suivante `http://www-sop.inria.fr/lemme/stalmarck/index.html`.

6.3.2 Formalisation des flottants en Coq

Mots clés : nombres flottants, norme IEEE754, certification.

Participant : Laurent Théry.

Dans le cadre de l'action de recherche coopérative AOC (Arithmétique des Ordinateurs Certifiée), Laurent Théry a commencé la formalisation des flottants machine suivant la norme IEEE 754. S'inspirant d'un premier travail de Patrick Loiseleur, il a introduit les nombres flottants et prouvé certaines de leurs propriétés élémentaires. Ce travail servira de base pour les travaux ultérieurs de l'action AOC.

6.3.3 Arithmétique en base arbitraire en Coq

Participants : Michel Hirschowitz, Laurent Théry.

Dans le cadre de son stage de maîtrise (1 mois), Michel Hirschowitz a développé une bibliothèque permettant d'effectuer des calculs (addition, soustraction, multiplication et division) dans une base arbitraire dans le démonstrateur Coq. En plus de ces opérations, une fonction de conversion permet de passer d'une base à l'autre. Ce travail se place dans le cadre de l'action AOC.

6.3.4 Expansions flottantes en Coq

Participants : Sébastien Bardin, Laurent Théry.

Dans le cadre de première année à l'IIE (2 mois et demi), Sébastien Bardin s'est attaqué à la formalisation des expansions flottantes dans Coq. Une expansion flottante est une liste de nombres flottants décroissante. Le nombre representé par une expansion est la somme de ses éléments. On peut définir sur ces expansions une arithmétique qui permet d'effectuer du calcul exact sur de grands nombres. Sébastien Bardin a introduit les notions de base des expansions s'appuyant sur la formalisation des flottants machines. Il a ensuite prouvé la correction de l'opérateur arithmétique `FastTwoSum` qui, étant donné deux nombres flottants, retourne une expansion formée de l'arrondi de leur somme et de son erreur . Il a ensuite formalisé l'addition de deux expansions, sans en prouver cependant la correction. Ce travail se place dans le cadre de l'action AOC.

6.3.5 La transformée de Fourier rapide en Coq

Participant : Venanzio Capretta.

Durant son séjour de deux mois dans le projet, Venanzio Capretta (en thèse avec H. Barendregt à Niemegen) a démontré en Coq la correction de la tranformée de Fourier rapide et de son inverse. Pour cela, il a utilisé une représentation particulière des polynômes en arbres avec les coefficients des puissances paires dans la branche gauche, et les coefficients des puissances impaires dans la branche droite, et cela récursivement.

Pour faciliter les preuves d'égalités sur ces arbres, il a utilisé une méthode de réflexion : il a défini un type des contextes d'arbres et un opérateur qui prouve l'égalité des substitutions de deux arbres égaux dans un même contexte.

La représentation en arbres n'est pas bonne pour prouver la correction de la transformée de Fourier inverse. Pour cela il a utilisé une deuxième représentation des polynômes comme fonctions d'un type fini vers le domaine des coefficients. Pour conclure ce développement, il a encore à prouver que les deux représentations sont équivalentes.

Venanzio Capretta a utilisé Pcoq puis Ctcoq pour développer les preuves. Ces outils lui ont permis de développer les preuves plus rapidement qu'avec les outils auxquels il était habitué.

6.3.6 Etude des fonctions récursives

Participants : Antonia Balaa, Yves Bertot.

Nous avons continué cette année l'étude de l'utilisation des fonctions récursives basées sur des ordres bien fondés dans Coq. Ces résultats ont mené à l'implémentation d'un jeu de commandes utilisables dans le système de preuve Coq. Ces commandes permettent la définition d'une fonction récursive basée sur un ordre bien fondé et effectuent une analyse sur cette fonction pour construire automatiquement un théorème important sur cette fonction.

Ces travaux ont donné lieu à une publication dans la conférence TPHOLs'2000 "Fix-point Equations for Well-Founded Recursion in Type Theory" [4].
L'adresse Web de cette conférence est :
`http://www.cse.ogi.edu/tpholls2000/default.htm`.

6.3.7 Changements de structure de données

Participants : Nicolas Magaud, Yves Bertot.

Nous avons commencé un travail sur la traduction de preuves d'une structure de donnée à une autre. Les expériences menées les années précédentes sur la certification d'algorithmes de calcul mathématique ont montré que les développements de preuves étaient très sensibles aux choix de structure de données. En particulier, si le développeur découvre qu'il doit changer sa structure de donnée pour améliorer l'efficacité de l'algorithme ou l'aisance avec laquelle il peut faire une preuve, il est souvent amené à refaire une grande partie du travail de preuve déjà effectué. Notre objectif est de comprendre si une grande partie de se travail peut être automatisée, en effectuant des traductions automatiques de preuves d'une structure de donnée dans une autre.

Une première solution a été ébauchée et une implémentation a été construite. Elle mène à un module qui peut être chargé dans le système et a déjà été utilisé avec succès pour traduire des démonstrations entre plusieurs façons de représenter les entiers naturels.

Ces travaux sont décrits dans un rapport [15] et ont donné lieu à un article soumis pour publication [18].

6.3.8 Certification d'algorithmes d'enveloppes convexes

Participants : David Pichardie, Yves Bertot.

Pour élargir le spectre des algorithmes certifiés en Coq, nous avons abordé la certification d'algorithmes de géométrie algorithmique. Nous nous sommes basés sur un ouvrage de D. Knuth décrivant les suppositions de base qui permettent de comprendre le fonctionnement des algorithmes d'enveloppe convexe. En partant d'une représentation du plan comme R^2, nous avons montré que les suppositions de Knuth (qu'il appelle axiomes) étaient vérifiées, et nous avons utilisé ces propositions pour établir le bon fonctionnement d'un algorithme de calcul d'enveloppe convexe. Nous avons ensuite montré que ce travail s'adaptait bien pour certifier des algorithmes différents. Le point qui reste à approfondir dans ce domaine est le traitement des positions dégénérées, habituellement ignorées par les auteurs du domaine de la géométrie algorithmique, mais incontournables en terme de certification de logiciel.

6.3.9 Formalisation des calculs sur les quaternions pour le contrôle d'attitude des satellites

Participants : Loïc Pottier, Laurence Rideau.

Suite à des discussions avec un ingénieur d'Alcatel Space Industry (Cannes), nous nous sommes intéressés à formaliser les quaternions en Coq. En effet le produit de quaternions est l'opération de base des codes permettant le contrôle d'attitude des satellites. Nous avons d'une part formalisé les quaternions sur R en Coq et prouvé qu'ils formaient un corps non commutatif. D'autre part, partant d'une procédure en ADA, nous avons formalisé un produit de vecteurs de dimension 4 (incluant la gestion de mémoire) et nous avons prouvé en Coq que le vecteur résultat de la procédure calculait bien le produit des quaternions correspondants. Cette preuve a été menée en supposant que nous travaillions sur les Réels. L'étape suivante (en cours) consiste à refaire la preuve avec des flottants (puisque c'est le cas du programme ADA de départ). Pour cela, nous voulons nous appuyer sur la formalisation de la norme IEEE754 des flottants réalisée par Laurent Théry dans le cadre d'AOC.

6.4 Sémantique des langages de programmation

6.4.1 Etude de la vérification de byte-code pour JAVA

Participant : Yves Bertot.

Dans le cadre de collaborations avec l'entreprise Gem+ et le projet Oasis, nous avons étudié la preuve d'un mini-vérificateur de byte-code pour un langage abstrait concernant l'initialisa-

tion. Le résultat est un vérificateur de byte-code certifié qui peut être traduit en un programme `Ocaml` grâce au mécanisme d'extraction de Coq.

Ce travail est décrit dans un rapport de recherche et un article soumis pour publication. Il sera prolongé en l'intégrant au vérificateur actuellement développé dans le projet Oasis qui ne traite pas des problèmes d'initialisation. Ce travail sera soutenu à partir de décembre par une subvention européenne dans le cadre d'un projet IST2000 «Verificard».

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Dyade

Résumé :

Dyade fournit les ressources pour le contrat de Claude Pasquier sur l'adaptation de Figue et Aïoli aux standards XML.

7.2 GénieII

Résumé :

Génie a fourni les ressources pour le contrat de Pascal Lequang sur le développement de `CtCoq` *et son portage en Java, jusqu'en mars.*

7.3 Dassault Aviation

Résumé :

Ce contrat finance la thèse d'Antonia Balaa, sur l'étude des fonctions récursives à terminaison non structurelle en théorie des types.

7.4 Gem+

Résumé : *En prévision pour 2001.*

7.5 Verificard

Résumé :

Le projet européen Verificard doit commencer au 1er décembre (partenaires Nimègue, Hagen (allemagne), SICS (suède), Munich, Gem+, Bull).

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 Action coopérative CFC

L'action CFC (Calcul Formel Certifié) s'est terminée en tout début d'année par une réunion de clôture à Paris, et sur un bilan positif (`http://www-sop.inria.fr/croap/CFC/`.

8.1.2 Action de Recherche Coopérative AOC

L'action de recherche coopérative AOC (Arithmétique des Ordinateurs Certifiée) commence cette année. Elle regroupe trois équipes de recherche : le projet Arénaire de l'UR Rhône-Alpes, le projet PolKa de l'UR Lorraine et le projet Lemme. Elle a pour but de certifier des algorithmes implantant des opérateurs arithmétiques ainsi que de vérifier que des algorithmes utilisant de tels opérateurs satisfont certaines propriétés. Une première réunion s'est tenue mi-février à Sophia, une seconde fin juin à Nancy et une troisième couplée avec la tenue d'un groupe de travail mi-novembre à Lyon. L'activité de cette action de recherche est présentée à l'adresse suivante :
`http://www-sop.inria.fr/lemme/AOC/`

8.2 Actions européennes

8.2.1 Réseau Types

Lemme participe activement au réseau Types dont la nouvelle formule a été acceptée par la commission européenne, et dont la réunion annuelle a eu lieu en décembre. Les membres du projet y ont fait plusieurs communications.

9 Diffusion de résultats

9.1 Manifestations scientifiques, missions

Antonia Balaa et Yves Bertot ont présenté leurs travaux à TPHOLs2000.

Antonia Balaa et Nicolas Magaud ont participé à l'école d'été APPSEM'2000 qui a eu lieu en septembre 2000 à Caminha, Minho, Portugal.

Yves Bertot a été invité pour un tutorial Coq à la conférence TPHOL (Theorem Proving in Higher Order Logics) qui a eu lieu en août 2000 à Portland, Oregon, Etats-Unis.

Yves Bertot, Loïc Pottier, Laurence Rideau et Laurent Théry se sont rendus fin juin au LORIA pour participer à la réunion de l'ARC AOC et mi-novembre à l'École Normale Superieure de Lyon pour participer au groupe de travail AOC.

Hanane Naciri a participé à l'ecole de jeunes chercheurs en programmation EJC'2000, qui a eu lieu en Mars à l'ENS de Lyon. Elle a présenté ses travaux à JED'2000, journée Ecrit et Document, spéciale jeunes chercheurs des colloques CIFED'2000 (Colloque International Francophone sur l'Ecrit et le document) et CIDE'2000 (Colloque International sur le Document Electronique), qui s'est déroulée en Juillet 2000 à Lyon. Elle a participé à l'école thématique "Nouveaux défis en Sciences de l'Information : Documents et Evolution" qui a eu lieu en septembre 2000 à Marseille, et au 5ème colloque Africain sur la Recherche en Informatique, CARI'2000 du 16 au 19 octobre 2000 à Antananarivo, Madagascar.

Claude Pasquier a participé au Workshop XOT [9].

Loïc Pottier a participé à la journée sur les prouveurs organisée par l'équipe Coq à Rocquencourt.

Laurence Rideau a participé à la 1ère conférence internationale sur MathML du 19 au 21 octobre à Urbana-Champaign, Illinois.

Laurent Théry s'est rendu à l'université de Pise (Italie) fin février début mars, fin mai à l'INRIA Rocquencourt pour donner un exposé dans le cadre du séminaire Coq, à l'université de l'Aquila (Italie) fin octobre début novembre.

9.2 Animation de la communauté scientifique

Laurent Théry a participé à l'organisation du groupe de travail AOC.

9.3 Divers

Loïc Pottier est membre nommé du CNU 27ème section et membre suppléant de commission de spécialistes 27ème section de l'UNSA.

Laurent Théry a affectué un travail de revue pour le journal AMAI.

9.4 Visites

Le professeur Deepak Kapur de l'université de New Mexico (E.U.) a rendu visite à l'équipe Lemme les 13 et 14 Novembre 2000.

9.5 Direction de thèses

Yves Bertot dirige les thèses d'Antonia Balaa et de Nicolas Magaud.

Loïc Pottier a dirigé la thèse de Stéphane Lavirotte, et codirige avec André Hirschowitz celle de Laurent Chicli.

Laurence Rideau dirige la thèse de Hanane Naciri.

9.6 Jury de thèses

Yves Bertot a participé au jury de thèse de Songsakdi Rongviryapanish (Nancy) et à celui de Yann Coscoy.

Loïc Pottier a participé au jury de thèse de Stéphane Lavirotte.

9.7 Encadrement de stagiaires

Yves Bertot : Nicolas Magaud, stage de DEA, David Pichardie, stage de première année de l'E.N.S. Cachan-Bretagne.

Loïc Pottier : François-Régis Sinot, première année de l'ENS Lyon, Marion Alliod, Guillaume Cassa, Marc Chiaverini et Michel Hirschowitz, maîtrise MIM de l'Université de Nice.

Laurent Théry : Sébastien Bardin et Michel Hirschowitz.

9.8 Enseignement

Antonia Balaa : 39h de TD sur Awk et C en DEUG 2ème année à l'UNSA.

Yves Bertot : Sémantique des langages de programmation en maîtrise (48 heures), Mécanisation des preuves en DEA (3 heures à Marseille), Mécanisation des preuves en DEA (6 heures à Nice), Présentation des systèmes de preuve (6 heures à l'ISIA avec Laurent Théry)

Laurent Chicli : 48h de cours-TD de probabilités et statistiques en Deug B et 52h de TD d'algorithmique en licence de Mathématiques à l'UNSA.

Nicolas Magaud : TP de systèmes informatiques DEUGMI2 à l'UNSA (25 heures)

Hanane Naciri : travaux pratiques du cours sur le Système Unix et le langage C, de Maîtrise d'Ingénierie Mathématique dispensé à l'Université de Nice Sophia-Antipolis (20 heures).

Travaux pratiques du stage JAVA en premiere année ESSI (20 heures) .

Travaux pratiques du cours JAVA en premiere année ESSI (22 heures).

Loïc Pottier :52h de cours d'algorithmique et logique en maîtrise Math-Info à l'UNSA.

Laurent Théry : cours dans les DEA de Nice et de Marseille ainsi qu'à l'école des Mines.

10 Bibliographie

Livres et monographies

[1] J. Harrison, M. Aagaard (éditeurs), *Lecture Notes in Computer Science, 1869*, Springer-Verlag, 2000.

Thèses et habilitations à diriger des recherches

[2] Y. Coscoy, *Explication textuelle de preuves pour le calcul des constructions inductives*, thèse de doctorat, université de Nice-Sophia Antipolis, 2000.

[3] S. Lavirotte, *Reconnaissance Structurelle de Formules Mathématiques Typographiées et Manuscrites*, thèse de doctorat, université de Nice-Sophia Antipolis, 2000.

Communications à des congrès, colloques, etc.

[4] A. Balaa, Y. Bertot, « Fix-point Equations for Well-Founded Recursion in Type Theory », *in : Theorem Proving in Higher Order Logics*, p. 1–16, 2000.

[5] L. Chicli, «Formalizing Sheaves and Schemes in Coq», *in : Workshop Types, Durham, Angleterre*, Décembre 2000.

[6] P. Letouzey, L. Théry, « Formalizing Stålmarck's algorithm in Coq », in : *Theorem Proving in Higher Order Logics* [4], p. 387–404.

[7] H. Naciri, L. Rideau, « Affichage interactif, bidimensionnel et incrémental de formules mathématiques », *in : CARI'2000, Antanarivo (Madagascar)*, 2000.

[8] H. Naciri, « Conception et réalisation d'outils pour l'interaction homme machine dans les environnements de démonstrations mathématiques. », *in : JED'2000, Lyon*, 2000.

[9] C. Pasquier, L. Théry, « A Distributed Editing Environment for XML Documents », *in : actes de XOT2000 : premier Workshop ECOOP dédié à XML et aux Technologies Objet, Cannes*, juin 2000.

[10] L. Pottier, « Extraction in Coq », *in : Workshop Types, Durham, Angleterre*, Decembre 2000.

Rapports de recherche et publications internes

[11] Y. Bertot, O. Pons, L. Pottier, « Dependency graphs for interactive theorem provers », *Rapport de recherche*, INRIA, novembre 2000.

[12] Y. bertot, « A Coq formalization of a Type Checker for Object Initialization in the Java Virtual Machine », *Rapport de recherche*, INRIA, novembre 2000.

[13] L. Chicli, « Une formalisation des faisceaux et des schémas affines en théorie des types avec Coq », *Rapport de recherche*, INRIA, Janvier 2001, soumis à TLCA 2001.

[14] C. Faure, J. Davenport, H.Naciri, « Multi-valued Results in Computer Algebra. », *Rapport de recherche n° RR-4007*, INRIA, Septembre 2000, `http://www.inria.fr/rrrt/rr-4007.html`.

[15] N. Magaud, Y. Bertot, « Changements de représentation des données dans le calcul des constructions inductives », *Rapport de recherche*, INRIA, Octobre 2000, `http://www.inria.fr/rrrt/rr-4040.html`.

[16] L. Pottier, « Extraction dans le CCI », *Rapport de recherche n° RR-4026*, INRIA, oct 2000, soumis à JFLA2001.

[17] L. Pottier, « Quotients in the CIC », *Rapport de recherche*, INRIA, nov 2000, soumis à TLCA 2001.

Divers

[18] N. Magaud, Y. Bertot, « Changements de représentation des structures de données dans Coq : le cas des entiers naturels », soumis à la conférence JFLA'01, Septembre 2000.