INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Projet* COQ

## *Spécifications et preuves de programmes*

*Rocquencourt*

THÈME 2A

# Table des matières

*Coq était jusqu'en septembre 1997 un projet commun avec le Laboratoire de l'Informatique du Parallélisme, URA 1398 du CNRS à l'ENS Lyon. Une demande de projet commun avec le LRI, UMR 8623 du CNRS à l'université Paris Sud est en cours d'élaboration. Au LRI, les membres du projet Coq appartiennent au thème "Types, preuves et programmes" de l'équipe DEMONS (Démonstration et programmation).*

# 1 Composition de l'équipe

**Responsable scientifique**

Christine Paulin [Professeur, Paris 11]

**Responsable permanent**

Gilles Dowek [CR, Inria]

**Assistante de projet (commun avec Cristal)**

Nelly Maloisel [Adjointe Administrative INRIA]

**Personnel Inria**

Gérard Huet [DR, Délégué aux Relations Internationales]

Benjamin Werner [CR]

**Collaborateurs extérieurs**

Philippe Audebaud [Maître de conférences, ENS-Lyon, en disponibilité jusqu'en septembre 1999]

Judicaël Courant [Maître de Conférences, Paris 11]

Jean Duprat [Maître de conférences, ENS-Lyon]

Jean-Pierre Jouannaud [Professeur, Paris 11]

Eduardo Giménez [Ingénieur, Dassault]

Jean Goubault-Larrecq [G.I.E. Dyade]

Hugo Herbelin [Maître de Conférences, Paris 10, détaché à l'Inria depuis le 1/09/99]

**Chercheur post-doctorant**

Jean-Christophe Filliâtre [Paris 11, contrat CALIFE puis CNET depuis le 1/09/99]

**Doctorants**

Bruno Barras [Bourse Inria jusqu'au 31/01/99]

Frédéric Blanqui [Allocataire MENRT, Paris 11]

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

Pierre Courtieu [Allocataire MENRT, Paris 11]

David Delahaye [Allocataire MENRT, Inria]

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

Jean-Christophe Filliâtre [Allocataire moniteur normalien, Paris 11 jusqu'au 31/08/99]

Benjamin Grégoire [Allocataire MENRT, Inria depuis le 1/09/99, également projet CRISTAL]

Patrick Loiseleur [Allocataire moniteur normalien, Paris 11 jusqu'au 30/06/99]

Micaela Mayero [Allocataire MENRT, Inria]

Alexandre Miquel [Allocataire MENRT, Inria]

Muriel Roger [Allocataire MENRT, G.I.E. Dyade depuis le 1/09/99]

Eva Rose [Bourse Inria, G.I.E. Dyade]

Stéphane Vaillant [Allocataire MENRT, Inria depuis le 1/09/99]

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

**Stagiaires**

Benjamin Grégoire [DEA SPP, Paris 7, avril à septembre 1999, commun avec le projet CRISTAL]

Frédéric Mazoit [Maîtrise d'informatique, Paris 11, avril à juin 1999]

Kumar Neeraj Verma [IIT Dehli, mai à juillet 1999]

Muriel Roger [DEA SPP, Paris 7, avril à septembre 1999]

Stéphane Vaillant [DEA SPP Paris 7, avril à septembre 1999]

# 2 Présentation et objectifs généraux

Le projet Coq a pour but la réalisation de *systèmes de traitement de démonstrations*, c'est-à-dire de systèmes capables de vérifier, produire et transformer des démonstrations mathématiques.

Ces systèmes peuvent en particulier être utilisés pour vérifier les démonstrations de correction de matériels et de logiciels informatiques vis-à-vis d'une spécification formelle, permettant ainsi la production de produits informatiques de qualité totale (*zéro défaut*). Ils construisent explicitement un objet représentant la preuve de correction du programme qui peut ainsi être à nouveau vérifiée dans le cadre de la certification d'un logiciel.

Dans la conception d'un système de traitement de démonstrations, le choix du formalisme dans lequel s'expriment les définitions, les axiomes, les théorèmes et les démonstrations est crucial. La théorie des ensembles n'est pas totalement adaptée aux buts orientés vers l'informatique que nous poursuivons, nous travaillons donc avec une variante issue des travaux en théorie des types appelée *Le Calcul des Constructions Inductives* et notée CCI.
Ses caractéristiques sont :

- les fonctions sont représentables par des algorithmes;
- les prédicats peuvent être définis inductivement par un ensemble de clauses;
- les preuves sont des objets du langage.

Ses avantages sont :

- la preuve d'existence d'un objet est automatiquement traduite en un programme exécutable réalisant la spécification initiale; inversement la donnée d'une spécification et d'un programme permet d'engendrer des obligations de preuves;
- l'objet preuve sert à la documentation et permet la vérification par un programme indépendant rendant ainsi la démonstration sûre quelle que soit la complexité des procédures de preuve utilisées.

Les principaux travaux portent sur:

- des concepts de haut-niveau dans le langage: modules, sous-typage, définitions par filtrage, structures infinies;
- une représentation interne des termes de preuve rendant efficaces les opérations de réduction et de démonstration;
- le développement de théories classiques des mathématiques et de programmes certifiés (en particulier le noyau du système Coq).

# 3 Fondements scientifiques

## 3.1 Environnement interactif de preuves

**Mots clés** : tactique, objet preuve.

**Glossaire :**

**Tactique** Une tactique est un programme permettant à partir d'une propriété à démontrer de construire un ensemble (éventuellement vide) de nouvelles propriétés qui sont suffisantes pour valider la propriété initiale. Les tactiques peuvent être combinées en utilisant des opérateurs appelés *tacticals*.

**Objet preuve** Un objet preuve est une représentation concrète d'une preuve qui explicite les étapes élémentaires ayant permis de justifier la correction du développement (lemmes utilisés, instanciations, récurrence).

**Calcul des Constructions Inductives** Langage issu de la théorie des types. Intégrant des types d'ordre supérieurs, polymorphes, dépendants et des définitions inductives et coinductives primitives, il permet la représentation des spécifications et des preuves et sert de base à l'assistant Coq.

**CCI** Calcul des Constructions Inductives.

**Résumé :** *Coq est un assistant à la démonstration, c'est-à-dire un outil permettant de construire des théories mathématiques en introduisant des définitions, des hypothèses, en énonçant des propriétés et enfin en construisant interactivement des preuves de théorèmes.*

Notre système de traitement de démonstrations Coq repose sur le formalisme du *Calcul des Constructions Inductives* que nous avons élaboré. Le langage de spécification associé est appelé Gallina. Il contient en particulier un calcul des prédicats d'ordre supérieur typé permettant d'exprimer des propriétés sur des objets appartenant à des structures algébriques quelconques. Il permet une représentation directe sans codage excessif des notions fondamentales d'objets structurés, de fonctions ou de relations très souvent utilisées dans les preuves informatiques.

La puissance même du système logique rend *a priori* impossible l'automatisation complète de la recherche d'une preuve d'une formule. L'approche repose sur deux autres idées. La première est qu'étant donnée une démonstration formelle assez détaillée, il est possible de vérifier mécaniquement sa correction. La seconde est que des programmes arbitraires (appelés tactiques) peuvent être utilisés pour construire par étapes successives une démonstration dont la correction est ensuite vérifiée. Ceci permet de laisser l'utilisateur construire ses propres procédures tout en garantissant la correction des preuves finales.

## 3.2 Déduction modulo

**Résumé :** *Une particularité de la théorie des types est de fournir un système logique comportant à la fois des étapes de raisonnement et des calculs. G. Dowek, Th. Hardin (Paris 6 et projet PARA) et C. Kirchner (projet PROTHEO) ont imaginé conserver ces caractéristiques, tout en restant dans le cadre plus classique de*

*la logique du premier ordre. Ils ont proposé la* déduction modulo*, une nouvelle présentation de la logique dans laquelle les étapes de calcul et les étapes de déduction sont clairement distinguées. Ce formalisme se prête particulièrement bien à la mécanisation des mathématiques qui doit pouvoir combiner un raisonnement abstrait interactif et des étapes calculatoires automatisables.*

Formellement, la déduction modulo définit une *théorie* non pas simplement comme un ensemble d'axiomes, mais comme un ensemble d'axiomes *et une congruence* définie sur les propositions du langage et typiquement présentée par un système de réécriture. Les propositions congruentes sont identifiées. Ainsi, si les propositions $2+2=4$ et $4=4$ sont congruentes toute démonstration de l'une est une démonstration de l'autre et le passage de l'une à l'autre, qui est un simple calcul, n'apparaît pas dans la démonstration. L'originalité de cette définition est que la congruence sur les propositions peut identifier des propositions atomiques avec des propositions non atomiques. Par exemple, la proposition $a \times b = 0$ peut être identifiée avec la proposition $a = 0 \vee b = 0$.

G. Dowek, Th. Hardin et C. Kirchner ont proposé une méthode de démonstration pour la déduction modulo. Cette méthode, la *résolution modulo*, est une extension de la résolution équationnelle, où une nouvelle règle permet de surréduire un littéral par une règle de réécriture transformant une proposition atomique en une proposition non atomique. Ils ont montré que cette méthode est complète pour toutes les congruences modulo pour lesquelles la déduction vérifie la propriété d'élimination des coupures.

### 3.3 Preuves et Programmes

**Mots clés** : isomorphisme de Curry-Howard, réalisabilité, logique intuitionniste, calcul des constructions inductives.

**Glossaire :**

**Logique Intuitionniste** Logique constructive dans laquelle la disjonction ou la quantification existentielle ont un sens fort : une preuve de $A \vee \neg A$ permet de décider qui de $A$ ou $\neg A$ est vérifié, une preuve de $\exists x.P(x)$ permet de construire $a$ tel que $P(a)$. Le principe du tiers-exclu $A \vee \neg A$ n'est pas a priori admis pour une formule quelconque $A$.

**Réalisabilité** Introduite par le logicien Kleene, il s'agit d'une interprétation de la logique intuitionniste dans laquelle toute formule est vue comme un ensemble de programmes qui satisfait un propriété caractérisée par la formule. La validité de la réalisabilité assure que toute preuve d'une formule peut être traduite en un programme appartenant à l'interprétation de la formule et donc satisfaisant la propriété.

**Isomorphisme de Curry-Howard** Correspondance entre les preuves en déduction naturelle et les lambda-termes typés.

**Résumé :** *Le Calcul des Constructions Inductives repose sur l'isomorphisme de Curry-Howard qui identifie les preuves et les programmes fonctionnels.* Coq *propose une chaîne de développement allant de l'écriture de la spécification jusqu'à la production d'un code exécutable certifié.*

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 sera représenté par un couple formé d'un entier et d'une preuve que cet entier est pair. La seconde 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 $Q(x, y)$ décrit une telle relation, une preuve dans le *Calcul des Constructions Inductives* de la totalité de cette relation (c'est-à-dire que $\forall x.\exists y.Q(x, y)$) est en fait une fonction qui, à tout objet $a$ associe un objet $b$ et une preuve de $Q(a, b)$. À partir de cette preuve, il est possible d'exprimer dans le calcul à la fois un programme fonctionnel $f$ et sa preuve de correction (c'est-à-dire $\forall x.Q(x, f(x))$). La fonction $f$ appliquée à l'entrée $a$ se réduira pour fournir la sortie $b$.

Mais les preuves ne sont pas, en général, des programmes efficaces, et il est nécessaire, en pratique, d'éliminer certaines parties de ces preuves 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 code machine.

Dans cette approche, la spécification du programme est vue comme une formule mathématique et le programme certifié est obtenu à partir de la preuve de cette formule.

Cette interprétation est particulièrement adaptée à la programmation fonctionnelle mais fournit également une technique de preuve de programme impératif.

# 4 Domaines d'applications

## 4.1 Panorama

**Mots clés** : télécommunication, commerce électronique, preuve de compilateur, calcul formel.

Les assistants de preuves visent les domaines de l'informatique où il est nécessaire de garantir un comportement sans faille des programmes dans des cas trop sophistiqués pour être traités de manière automatique.

Le système Coq a été ou est actuellement utilisé dans les contextes suivants :

- modélisation de protocoles cryptographiques utilisés en commerce électronique (action VIP du G.I.E. Dyade, Trusted Logic)
- modélisation de politiques de sécurité dans le cadre d'applications Java sur cartes à puce (Trusted Logic)
- étude d'algorithmes de contrôle de conformité dans les réseaux ATM en vue de leur normalisation (CNET France Télécom, contrat RNRT CALIFE)
- preuve du compilateur SCADE du langage réactif Lustre (Dassault Aviation et Aérospatiale dans le cadre de l'action de développement Génie)
- certification d'algorithmes de Calcul Formel (Action de recherche coordonnée CFC)

## 4.2 Preuves formelles de protocoles

**Mots clés** : télécommunication, commerce électronique, spécification formelle.

**Résumé :** *Les réseaux sont des systèmes complexes, ouverts et évolutifs qui offrent des fonctionnalités évoluées pilotées par des logiciels. Il est essentiel de garantir la qualité de service ainsi que la sécurité des transactions, ce qui nécessite une modélisation formelle des problèmes.*

Les protocoles sont des programmes critiques traitant les communications. Bien que leur taille soit souvent réduite, leur complexité est grande. Ils sont de plus amenés à fonctionner dans des milieux perturbés dont la modélisation précise n'est pas aisée.

De nombreux services utilisent les réseaux télématiques tels qu'Internet ou les téléphones mobiles. Il faut donc concevoir de nouveaux protocoles offrant de plus en plus de fonctionnalités et garantissant la qualité de service. Le commerce électronique ou la téléphonie mobile imposent de plus que ces services garantissent la confidentialité des informations échangées.

**Utilité des preuves formelles** Une description formelle des fonctionnalités du protocole et une justification de son bon fonctionnement sont nécessaires pour satisfaire aux exigences de la certification et obtenir ainsi l'autorisation d'utilisation ou bien la normalisation d'une technique. De plus, les protocoles évoluent très vite, leur validation ne peut donc juste reposer sur la non-découverte d'erreurs lors de tests ou d'une utilisation massive.

Prouver un protocole est une tâche difficile, cependant la preuve ne variera en général pas beaucoup entre deux protocoles d'une même famille, il pourra être plus économique de reconstruire une preuve après une modification que de rejouer à partir de zéro un jeu de tests. De plus les preuves apportent une connaissance approfondie des algorithmes et peuvent souvent être une aide précieuse pour concevoir de nouvelles méthodes.

**Mécanisation des preuves** Les preuves peuvent être faites de manière automatique lorsque les problèmes se ramènent à des modélisations finies de taille raisonnable. Cette contrainte nécessite souvent d'abstraire le problème réel. Les assistants à la démonstration tels que Coq permettent de formaliser un problème arbitrairement compliqué et de le résoudre en utilisant des étapes élémentaires correspondant au raisonnement mathématique usuel. Coq apparaît alors comme l'outil du spécialiste lui permettant de spécifier formellement un protocole et d'en justifier le bon fonctionnement. Outre l'assurance de la correction d'un protocole, Coq permettra de faire évoluer la preuve en même temps que le protocole et fournira les éléments nécessaires au processus de normalisation ou de certification. Coq est actuellement utilisé dans ce sens pour des preuves de protocoles de commerce électronique dans l'action VIP de Dyade, et l'entreprise Trusted Logic ainsi que pour des preuves d'algorithmes intervenant dans les protocoles du réseau ATM au CNET.

## 4.3 Certification de compilateurs

Les outils utilisés pour faire des preuves reposent en général sur un traitement de notations de haut-niveau qui sont ensuite traduites vers un langage exécutable. C'est le cas par exemple

de la vérification de systèmes réactifs où l'analyse est faite à partir de programmes écrits dans les langages synchrones qui sont ensuite traduits vers des automates.

Afin d'assurer la certification des programmes ainsi obtenus, il est nécessaire de justifier la correction du compilateur. Le formalisme de Coq se prête bien à ce genre de preuve du fait de la représentation possible des règles sémantiques par des définitions inductives et coinductives et de la possibilité d'obtenir un compilateur exécutable certifié à partir d'une preuve de correction de la traduction.

Dassault-Aviation en collaboration avec Aérosapatiale développe actuellement avec Coq une version certifiée du compilateur du langage synchrone Lustre composant de l'environnement Scade de Verilog.

Cette approche peut être appliquée au système Coq lui-même, le noyau de vérification de preuve est un programme critique qui s'apparente aux compilateurs et dont le développement a été formalisé à l'aide du système Coq lui-même [12].

# 5 Logiciels

**Participants** : Bruno Barras, Judicaël Courant, David Delahaye, Jean-Christophe Filliâtre, Hugo Herbelin, Patrick Loiseleur, Christine Paulin [Correspondante].

**Résumé :** *Le système Coq développé dans le projet est un assistant à la démonstration qui permet de développer interactivement des spécifications et des preuves.*

La principale originalité du logiciel 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 preuves comme des programmes certifiés mise en œuvre dans une compilation des preuves sous forme de programmes ML mais aussi des outils pour associer un programme à une spécification et engendrer automatiquement des obligations de preuve permettant de justifier sa correction;
- une notion primitive de définitions co-inductives permettant de représenter directement des structures infinies régulières et de construire des preuves sur de tels objets sans passer par la notion classique de bisimulation.

Au niveau de l'architecture de l'assistant, 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 développés à l'aide de Coq, on peut mentionner :

– modélisation et preuve du protocole d'authentification CSET utilisé en commerce électronique,

– une preuve du compilateur du langage réactif Lustre utilisé dans l'environnement industriel Scade,

– une preuve du noyau critique de l'environnement Coq,

– plusieurs modélisations des propriétés du $\pi$-calcul,

– le développement de bibliothèques d'algèbre et d'une version certifiée de l'algorithme de Buchberger utilisé en Calcul Formel.

La version V6.3 du système Coq est disponible à l'URL `http://coq.inria.fr/assis-fra.html`. Écrite en Objective Caml et Camlp4, elle 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 (au CNET Lannion, chez 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 modérée par Micaela Mayero (`mailto:coq-club@pauillac.inria.fr`) permet l'échange entre les personnes intéressées par le système.

# 6 Résultats nouveaux

## 6.1 Développement de l'assistant Coq

**Résumé :** *Nous avons distribué cette année la version Coq V6.3 en juillet et V6.3.1 en novembre.*

*Les modifications apportées à cette version répondent principalement aux besoins des utilisateurs :*

- *l'ajout d'une nouvelle procédure expérimentale de preuve automatique (`AutoRewrite`);*
- *développement d'une tactique `Quote` pour faciliter les preuves par réflection;*
- *assouplissement du langage de spécification par amélioration de l'algorithme d'inférence automatique;*
- *élargissement de la condition de garde dans les définitions récursives permettant en particulier le traitement des définitions inductives emboitées;*
- *amélioration des performances.*

*Nous préparons pour Janvier 2000 une version Coq V7 correspondant à une restructuration complète du code afin d'en améliorer la visibilité et de préparer l'intégration des fonctionnalités de preuves modulaires et modulo décrites dans 6.5.*

*D'autre part, un travail à plus long terme est la conception d'un langage de preuve déclaratif, facilitant la lisibilité des fichiers de preuve et la conception par l'utilisateur de nouvelles tactiques.*

### 6.1.1 Langage de description de preuves

**Participants** : David Delahaye, Benjamin Werner.

**Mots clés** : Langages de preuves, Langages de tactiques, Outils d'aide à la preuve.

Il existe au moins trois manières de représenter des preuves :

- Dans les systèmes à la LCF dont est issu Coq, les preuves s'effectuent en appliquant des tactiques à un ensemble de buts courants. Ceci rend les scripts de preuves peu lisibles (il faut les exécuter pour visualiser les propriétés à prouver).
- Un système comme Mizar utilise un langage de preuves déclaratif mais peu automatisé.
- En théorie des types, langage sous-jacent à Coq, toute preuve peut également se ramener à l'écriture d'un $\lambda$-terme.

En s'appuyant sur cette analyse, D. Delahaye a proposé un premier langage de preuves à base de termes et de tactiques que l'on pourrait situer entre le langage de termes pur et le langage procédural. L'utilisateur peut ainsi donner un terme dans lequel il utilise des tactiques pour les parties qu'il juge "décidables" et des métavariables (preuves incomplètes) qui seront instanciées ultérieurement. Un prototype de ce langage a été réalisé.

Parallèlement, D. Delahaye a redéfini le langage de tacticiels (opérateurs de combinaison de tactiques) en un petit noyau fonctionnel avec récurseur et filtrage qu'il a implanté. Un des objectifs de ce langage est la possibilité d'écrire un grand nombre de tactiques au niveau même de Coq plutôt que dans le langage d'implantation. D. Delahaye a expérimenté ce langage en réécrivant la tactique `Tauto` (tautologies propositionnelles intuitionnistes) Il a obtenu un gain non négligeable de place et de lisibilité (de 2000 lignes d'Ocaml à 100 lignes). Une très forte amélioration des performances fut constaté avec un pourcentage allant jusqu'à 95%. D. Delahaye élabore actuellement un système de types pour ce noyau ainsi qu'une sémantique.

### 6.1.2 Nouvelles tactiques

**Participants** : David Delahaye, Patrick Loiseleur.

**AutoRewrite** Désireux d'augmenter l'automatisation du système Coq, D. Delahaye a implanté une tactique `AutoRewrite` inspirée des tactiques de PVS. Cette tactique effectue de la réécriture automatique au moyen d'une base de règles. On peut lui donner des listes de tactiques s'appliquant soit au but principal soit aux sous-buts engendrés et qui possèdent des modifieurs permettant des comportements spécifiques.

**Quote** Dans les preuves utilisant des techniques de réflection, la procédure de décision fonctionne sur une structure concrète de terme qui est ensuite interprétée comme une proposition Coq. Pour montrer une proposition Coq particulière, il faut donc construire un terme concret qui s'interprètera dans cette proposition. P. Loiseleur a développé une tactique générique `Quote` qui automatise cette étape dans le cas où la fonction d'interprétation suit un schéma particulier.

### 6.1.3 Coq V7

**Participants** : Jean-Christophe Filliâtre, Hugo Herbelin.

J.-C. Filliâtre a entrepris de réécrire l'implantation du système Coq, dans le cadre d'un travail post-doctoral. Il s'agit en premier lieu d'isoler clairement le noyau critique de construction de l'environnement et d'attacher un soin particulier à sa réalisation, afin d'en améliorer la sécurité. Il s'agit également de clarifier le code existant pour en permettre une meilleure compréhension et son évolution. Enfin, il s'agit d'identifier les fonctions critiques au niveau des performances et de les optimiser.

H. Herbelin a réorganisé l'étape de transformation d'un pseudo-terme de Coq utilisant des notations de haut niveau (arguments implicites, coercions, filtrages multiples) vers un terme pouvant être certifié par le noyau de Coq.

Cette version devrait se substituer au système Coq actuel en janvier 2000 et servira de support à l'intégration des fonctionnalités de preuves modulaires et modulo.

### 6.1.4 Gestion des univers

**Participant** : Hugo Herbelin.

Les types du CCI sont organisés en une hiérarchie stratifiée d'univers gérée de manière implicite comme un graphe de contraintes. Le système actuel construit des graphes de taille importante ce qui réduit les performances de la vérification et du chargement de modules.

H. Herbelin a proposé une simplification de la gestion des contraintes d'univers dans le Calcul des Constructions Inductives avec univers. Il a proposé une présentation du CCI dont les univers sont eux-mêmes des termes dans une certaine algèbre, limitant alors la taille du graphe de contraintes associé à un terme au nombre d'univers apparaissant explicitement dans ce terme.

## 6.2 Modélisation dans Coq

**Résumé :** *Nous étudions la modélisation de différentes théories mathématiques et informatique pour lesquelles la vérification formelle est particulièrement importante ou innovante. Cette étude nous permet de mesurer l'adéquation du langage de spécification et des outils de preuves proposés par Coq à la résolution de problèmes spécifiques et de dégager des méthodologies de développement.*

*Cette année, nous avons plus particulièrement étudié :*

- *la modélisation d'une politique de sécurité du code mobile reposant sur le typage proposée par X. Leroy et F. Rouaix que nous détaillons dans le module consacré*

*à la sécurité 6.4;*

- *des exemples de développement de preuves sur les réels qui s'inscrivent dans un projet d'applications de preuves formelles aux problèmes d'analyse numérique;*
- *des preuves d'algorithmes réputés complexes comme un algorithme distribué de récupération de mémoire (GC) ou des algorithmes probabilistes;*
- *des preuves de procédures de décision dans le but d'intégrer dans Coq des méthodes de preuves automatiques, efficaces et sûres qui sont présentées dans 6.3;*
- *des preuves d'algorithmes utilisant des données modifiables en place en s'appuyant sur une méthode générale de preuve de programme impératifs dans Coq.*

### 6.2.1 Formalisation des nombres réels

**Participants** : Micaela Mayero, Gilles Dowek.

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

M. Mayero a poursuivi son développement concernant les nombres réels. Elle a formalisé les notions de limite et de dérivée et prouvé les propriétés élémentaires de ces notions. Une difficulté a été de choisir une présentation adaptée à Coq et au développement futur d'un dérivateur Fortran certifié.

M. Mayero a commencé la formalisation d'autres fonctions réelles telles la fonction exponentielle, les fonctions trigonométriques sinus et cosinus. La difficulté principale consiste à définir le plus judicieusement possible la notion de série infinie ainsi qu'à prouver en Coq les propriétés s'y rapportant.

### 6.2.2 Preuve d'un dérivateur Fortran

**Participants** : Micaela Mayero, Gilles Dowek.

**Mots clés** : Méthodes formelles, analyse numérique, dérivation, Odyssée, Fortran.

*Dans le cadre de ce travail, des contacts ont été pris en vue de collaborations avec des membres des projets Estime (Rocquencourt), Ondes (Rocquencourt) et Safir (Sophia).*

Suite à l'étude du système Odyssée effectuée l'an passé, Micaela Mayero a orienté son travail vers la preuve d'un dérivateur formel du langage Fortran. Pour ce faire, elle a commencé à définir une formalisation du noyau Fortran en Coq et s'intéresse actuellement au choix d'une sémantique adaptée (dénotationnelle ou opérationnelle).

Micaela Mayero a effectué la preuve d'une maquette de dérivateur. Cela lui a permis de mettre en œuvre la méthode de formalisation sur un exemple simple et de tester la commodité des choix concernant la formalisation de la dérivée dans les réels.

### 6.2.3 Algorithme de gestion de mémoire

**Participant** : Jean Duprat.

**Mots clés** : ramasse-miettes, algorithmique distribuée.

J. Duprat a collaboré avec Luc Moreau de l'Université de Southampton (UK) pour implanter en Coq une preuve de correction et de vivacité d'un algorithme de comptage de références distribué proposé par Luc Moreau [Mor97]. Ce travail a permis de clarifier la preuve et de préciser davantage les spécifications de la machine abstraite servant de support d'exécution à l'algorithme. Cette preuve est disponible dans les contributions de Coq, elle a fait l'objet d'un article soumis à Acta Informaticae [35]. J. Duprat a présenté ce travail à la conférence TYPES'99.

### 6.2.4 Algorithmes probabilistes

**Participants** : Philippe Audebaud, Christine Paulin.

En collaboration avec Richard Lassaigne (Équipe de Logique à l'université Paris 7), Ph. Audebaud et C. Paulin ont commencé à étudier la preuve d'algorithmes probabilistes en théorie des types. De tels algorithmes apparaissent difficiles à spécifier et prouver. En particulier, l'aspect probabiliste nécessite des développements mathématiques abstraits. Un schéma de preuve a été donné pour un algorithme probabiliste de recherche de coupures minimales dans des graphes, C. Paulin a présenté ce travail au workshop annuel du groupe IFIP WG 2.3. Une autre approche consiste à adapter les techniques de preuves reposant sur l'interprétation des programmes comme transformateur de distribution sur les données.

### 6.2.5 Preuve de programmes impératifs

**Participants** : Jean-Christophe Filliâtre, Christine Paulin-Mohring.

J-C. Filliâtre a soutenu sa thèse [14] le 12 juillet 1999 qui porte sur la preuve de programmes impératifs dans le cadre de la théorie des types. Les programmes considérés comprennent aussi bien des traits purement fonctionnels (fonctions comme objets de première classe) qu'impératifs (références, tableaux, exceptions). Pour cela, il a défini une traduction des programmes impératifs vers des programmes fonctionnels de même sémantique. Cette traduction s'inspire des travaux de Wadler et Moggi sur les monades, qu'il a raffinés en introduisant une notion de monades dépendant des effets du programme.

J.-C. Filliâtre a ensuite étendu sa méthode de traduction à des programmes annotés par des assertions à la manière de la logique de Floyd-Hoare [26]. Il obtient ainsi des termes de preuve partielle de la spécification du programme dont les parties manquantes sont autant d'obligations de preuve. Il a montré que la validité de ces obligations de preuve entraîne la correction totale du programme. Il a également montré un résultat de complétude relative pour cette méthode.

D'un point de vue pratique, Jean-Christophe Filliâtre a implanté cette méthode de preuve de programmes impératifs dans le système Coq et l'a utilisée pour établir la correction de plusieurs algorithmes non triviaux tels que Find, l'algorithme de Knuth-Pratt-Morris, ou encore,

[Mor97] L. Moreau, «A distributed Garbage Collector with diffusion tree reorganization and object mobility», *rapport de recherche*, Southampton University, 1997.

lors d'un travail commun avec Nicolas Magaud [25], un certain nombre d'algorithmes de tris en place (quicksort, heapsort).

## 6.3 Preuves par réflection

**Résumé :** *La réflection est une technique de preuve qui exploite l'expressivité calculatoire de la théorie des types et qui permet la construction de preuves 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 pour le cas de l'égalité dans les anneaux commutatifs, cette implantation a ensuite été améliorée par Patrick Loiseleur (tactique `Ring`).*

*Cette année, B. Werner a montré que l'on pouvait obtenir de meilleures performances en s'appuyant sur une notion intermédiaire de trace. J. Goubault et K. Neeraj Verma ont développé une bibliothèque de "Binary Decision Diagrams" afin d'automatiser les preuves nécessitant du calcul booléen.*

### 6.3.1 Reflection avec traces

**Participant** : Benjamin Werner.

La tactique `Ring` utilise une fonction Coq pour la normalisation des expressions, ce qui pose des problèmes d'efficacité (l'exécution d'un programme dans Coq est environ 100 fois plus lente que son analogue en CAML) et est spécifique à cette théorie particulière.

Pour étendre le domaine d'application de la réflection, ainsi que pour réduire le temps de calcul nécessaire, B. Werner a suggéré d'expliciter les étapes de réécriture dans le terme de preuve, sous la forme d'une "trace" engendrée par un programme indépendant. Il a utilisé cette technique pour améliorer les performances de l'implémentation Coq. Ces travaux ont été présentés au congrès TYPES'99.

Ce travail donne lieu à une collaboration avec le projet Prothéo (LORIA), dont le logiciel ELAN semble bien adapté à la construction des traces.

### 6.3.2 Preuves sur les strutures finies

**Participants** : Jean Goubault-Larrecq, Kumar Neeraj Verma.

Dans le but de développer des preuves de protocoles cryptographiques courtes et puissantes, J. Goubault-Larrecq a poursuivi son travail sur l'automatisation des preuves dans les structures finies. Il a encadré le travail de K. Neeraj Verma, qui a réalisé un code complet de BDD certifié en Coq. Cette expérience montre qu'il est possible d'intégrer des outils de model-checking avec Coq sans mettre en cause l'intégrité logique de Coq. Cette approche a été testée sur des problèmes de benchmarks matériels. Ce travail a fait l'objet d'un rapport de recherche à paraître et d'une communication au groupe d'utilisateurs Coq de FM'99, ainsi qu'au séminaire inaugural de l'équipe PPS à Paris 7.

## 6.4 Propriétés de sécurité

**Résumé :** *Du fait de notre implication dans l'action VIP (Verified Internet Protocol) du G.I.E. Dyade, nous étudions plus particulièrement la formalisation de propriétés de sécurité en particulier dans le cadre du code mobile. Cette année, nos travaux ont porté sur les points suivants :*

- *preuves formelles de propriétés de sécurité liées au typage;*
- *mise au point d'une méthode automatique d'analyse de fichiers de log en vue de détecter les tentatives d'intrusion;*
- *étude d'une méthode de vérification de bytecode Java allégée de manière à pouvoir être utilisée sur une carte à puce;*
- *méthodes d'analyse statique pour la vérification automatique de propriétés de confidentialité des protocoles.*

### 6.4.1 Propriétés de sécurité issues du typage

**Participants** : Benjamin Grégoire, Xavier Leroy [CRISTAL], Benjamin Werner.

Lors de son stage de DEA [39], co-encadré par X. Leroy et B. Werner, B. Gregoire a développé une preuve liée à la sécurité de code mobile d'après un article de F. Rouaix et X. Leroy; de plus il a réalisé la preuve en Coq d'une conjecture proposée dans l'article en question. Le travail a permis une formalisation complète d'un lambda-calcul simplement typé avec des références à la ML , d'une sémantique à réduction et d'une sémantique à grand pas. Il a présenté ce travail lors d'une réunion de l'ARC Javacard.

### 6.4.2 Sécurité de code mobile Java sur des cartes bancaires intelligentes

**Participants** : Eva Rose, Jean Goubault-Larrecq.

**Mots clés** : carte à puce Java, système embarqué, code mobile, applet, sécurité, vérification de bytecode, machine virtuelle Java.

E. Rose, sous la direction de P. Lescanne à l'ENS Lyon jusqu'en août 99 puis sous celle de J. Goubault étudie le problème de la certification de programmes en bytecode Java (applets) embarqués sur des cartes à puce. Les techniques usuelles de vérification de bytecode sont coûteuses ce qui rend difficile leur application sur les cartes. E. Rose a introduit un algorithme « Lightweight Bytecode Verification » (LBV), qui se base sur la notion d'un « Lightweight Certificate» envoyé avec l'applet en clair (sans technique cryptographique). E. Rose a montré que son algorithme était correct et sans faille de sécurité, elle en a implanté un prototype pour engendrer les certificats en temps presque linéaire en la taille des programmes et sous-linéaire en espace. Elle étudie des extensions du système de types proposé permettant d'économiser les ressources dans l'environnement qui recoit l'applet. E. Rose a présenté ses travaux à la réunion de l'ARC Javacard en juillet.

### 6.4.3 Vérification statique d'applets cryptographiques

**Participants** : Nabil EL Kadhi, Jean Goubault-Larrecq.

**Mots clés** : Protocoles cryptographiques, analyse statique, sémantique concrète, sémantique abstraite.

Nabil EL Kadhi étudie des méthodes de vérification statique des propriétés de sécurité (par exemple la préservation du secret) du bytecode d'une applet Java utilisant des primitives cryptographiques. Pour simplifier la représentation et le processus d'analyse, un langage générique permettant de représenter les différentes actions de ces applets a été défini.

Le modèle d'éxécution est composé d'un ensemble d'acteurs ou threads fixé [Bol96]. On distingue les acteurs légitimes (de confiance) d'un côté et l'intrus, représentant tout ensemble de personnes ou processus pouvant accomplir des actions frauduleuses sur le réseau. Une sémantique concrète opérationnelle et une sémantique abstraite ont été définies pour l'ensemble des actions du langage sus-mentionné. La sémantique abstraite a pour objectif de propager, au fur et à mesure de l'exécution abstraite, un ensemble de contraintes relatives, en particulier, à l'évolution de l'état de connaissances de l'intrus. La représentation des connaissances de l'intrus est basée sur une formalisation proposée par Jean Goubault-Larrecq, qui a été étendue notamment pour pouvoir traiter de la notion de fraîcheur.

Contrairement à ce qui se fait usuellement dans ce domaine, l'approche proposée se base sur une analyse statique du code de l'applet et certaines hypothèses initiales permettant d'avoir une abstraction de l'environnement d'exécution (représentant les interactions avec les autres participants potentiels du protocole et l'intrus) sans avoir une spécification complète des différents rôles du protocole. Ce travail est décrit dans [40].

### 6.4.4 Audit de logs et model-checking

**Participants** : Jean Goubault-Larrecq, Muriel Roger.

M. Roger, durant son stage de DEA, a développé une technique d'audit de logs basée sur la logique temporelle; il s'agit de repérer des atteintes possibles à la sécurité d'un système en analysant de manière automatique les fichiers de "log". Cette technique a donné lieu au dépôt d'un brevet Dyade, fera l'objet d'une publication, et est utilisée au sein de deux contrats avec Bull. L'audit de logs est par ailleurs le thème d'une collaboration avec Mireille Ducassé, de l'IRISA, projet Lande, et de l'INSA Nantes.

## 6.5 Déduction, réécriture et modularité

**Mots clés** : réécriture, modules, calcul des constructions.

**Résumé :** *La mécanisation des raisonnements et plus particulièrement des*

[Bol96] D. Bolignano, « Formal Verification of Cryptographic Protocols », *in: 3rd ACM Conference on Computer and Communication Security*, 1996, `http://www.dyade.fr/actions/VIP/approach-description.html`.

*preuves de programme repose sur trois composants :*

- *la modélisation de spécifications et de raisonnements abstraits,*
- *la résolution efficace des étapes calculatoires du raisonnement,*
- *l'organisation modulaire des développements.*

*Notre objectif est de construire un cadre général permettant d'intégrer ces trois composants et reposant sur la théorie des types pour l'aspect déductif et modulaire et la réécriture pour l'aspect calcul (comprenant l'automatisation des preuves). Chacune de ces composantes est théoriquement bien comprise, cependant leur combinaison pose de nombreuses questions tant sur le plan fondamental (préserver la cohérence) que sur celui d'une implantation efficace.*

*L'approche repose d'une part sur la déduction modulo (cf 3.2) d'autre part sur l'adaptation des techniques de réécriture (terminaison, preuve) à la théorie des types d'ordre supérieur.*

### 6.5.1 Théorie des types simples

**Participants** : Gilles Dowek, Thérèse Hardin, Claude Kirchner.

**Mots clés** : déduction modulo, théorie des types simples, calcul des substitutions explicites.

G. Dowek, Th. Hardin et C. Kirchner ont proposé une formulation de la théorie des types simples dans le cadre de la déduction modulo utilisant le calcul des substitutions explicites comme langage d'expression des fonctions. Ils ont montré que la résolution modulo appliquée à cette théorie simulait la résolution d'ordre supérieur étape par étape. Ce travail a été publié dans [23].

### 6.5.2 Élimination des coupures en déduction modulo

**Participants** : Gilles Dowek, Benjamin Werner.

G. Dowek et B. Werner ont montré que pour de nombreuses congruences, la déduction modulo avait la propriété d'élimination des coupures. En particulier pour la théorie des types, pour toutes les congruences définies par un système de réécriture sans quantificateurs et pour tous les systèmes de réécriture positifs. Ce travail a été publié dans [24].

### 6.5.3 Schéma de définition récursive

**Participants** : Frédéric Blanqui, Jean-Pierre Jouannaud.

F. Blanqui et J.-P. Jouannaud en collaboration avec Mitsuhiro Okada (Université Keio à Tokyo) ont poursuivi leur travail sur le "schéma général", un schéma de règles à l'ordre supérieur, qui est compatible avec la $\beta$-réduction et le typage. La définition de fonctions suivant ce schéma est une alternative à la définition de fonctions dans Coq sur les structures inductives par analyse par cas et point fixe gardé.

Ils ont montré que le schéma est compatible avec les types inductifs strictement positifs non-dépendants, et permet de définir les récurseurs. Ce travail a été présenté à la conférence RTA'99, et une version journal a été acceptée dans TCS [18, 22].

### 6.5.4 Ordre bien fondé pour fonctions d'ordre supérieur

**Participants** : Jean-Pierre Jouannaud, Daria Walukiewicz-Chrząszcz.

J.-P. Jouannaud, en collaboration avec Albert Rubio de l'Université Polytechnique de Catalogne à Barcelone, a poursuivi son travail sur les ordres bien fondés à l'ordre supérieur. Il a défini un nouvel ordre, "Higher Order Recursive Path Ordering" (HORPO), qui étend l'ordre "First-Order Recursive Path Ordering" de Dershowitz aux signatures polymorphes d'ordre supérieur, et est compatible avec la $\beta$-reduction. Cette nouvelle technique se révèle beaucoup plus puissante que la précédente, dont elle a récupéré certains aspects, mais n'a pour l'heure été développée que pour une discipline de typage simple, les signatures "algébriques" étant toutefois polymorphes et avec constructeurs de types. Ce travail a été présenté à la conférence LICS'99 [29].

Dans son travail de thèse, Daria Walukiewicz-Chrząszcz étudie l'extension de ces résultats au CCI. Elle a défini un système de types consistant en les règles du Calcul des Constructions étendues par un système de réécriture. Elle cherche à établir que la normalisation forte est préservée lorsque les règles de réécriture satisfont une condition décrite à l'aide d'un ordre bien-fondé HORPO.

### 6.5.5 Calcul de modules au-dessus des systèmes de types purs

**Participant** : Judicaël Courant.

**Mots clés** : PTS, module, modularité, bibliothèque de preuve, théorie mathématique.

J. Courant a continué son travail sur la mise au point d'un formalisme de modules pour les preuves, afin de permettre le développement de bibliothèques de preuves modulaires. Il étudie l'intégration de ce système à Coq ainsi que certaines extensions.

Le plus gros obstacle à la réutilisation des systèmes de modules à la SML pour la preuve 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. J. Courant a montré qu'il n'en était rien et que l'on pouvait obtenir une sémantique par réduction pour un système de modules comportant des types abstraits et construit au-dessus d'un système de types purs. L'étude des réductions s'est avérée 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.

J. Courant a publié une version anglaise de sa thèse sous forme de rapport de recherche [31] ; une version journal de ce rapport est en cours de soumission. Il travaille sur une simplification de la preuve de normalisation qui devrait permettre la définition d'extensions du calcul de

module, telles que la possibilité d'abréger des types de modules ou de définir des foncteurs surchargés (modules paramétrés se spécialisant en fonction de leur argument).

### 6.5.6 Intégrer réécriture et modularité au Calcul des Constructions

**Participants** : Jacek Chrząszcz, Jean-Pierre Jouannaud.

Jacek Chrząszcz a étudié les systèmes de modules existants vis-à-vis de leur capacité à être combinés avec des fonctions définies par des ensembles de règles de réécritures. Il est apparu que les concepts de foncteurs applicatifs ou de modules anonymes qui ont été proposés par les auteurs de systèmes de modules récents, ne sont pas adaptés à la vision habituelle de réécriture algébrique.

Jacek Chrząszcz a élaboré une version préliminaire d'un système comprenant les règles du Calcul des Constructions, un système de modules avec foncteurs génératifs et de la réécriture.

L'implantation future de cette extension dans le système Coq nécessitera encore une adaptation des méthodes de vérification de normalisation et de confluence des systèmes de réécriture à un cadre modulaire.

### 6.5.7 Structures non libres en théorie des types

**Participant** : Pierre Courtieu.

P. Courtieu, sous la direction de Laurence Puel de l'Université Paris Sud, étudie la représentation de certaines structures non libres dans le Calcul des Constructions Inductives (CCI). Une structure non libre est une structure dans laquelle des termes commençant par des constructeurs différents peuvent être égaux. Ces structures, faciles à définir en théorie des ensembles, ne sont pas aisément exprimable dans CCI. Des travaux existent en réécriture qui permettent de manipuler automatiquement des structures non libres lorsqu'il existe une structure libre des formes normales.

P. Courtieu a adapté ces techniques au Calcul des Constructions en définissant une notion de *types normalisés*, en cours d'implantation dans Coq. Il étudie à présent d'une part une automatisation de cette solution, et d'autre part une autre solution plus théorique, qui implique une extension de la théorie CCI.

## 6.6 Formalismes et automatisation des preuves

**Résumé :** *Nous étudions plus généralement des formalismes logiques adaptés à la mécanisation du raisonnement, en particulier sur le plan de l'automatisation des preuves.*

### 6.6.1 Traitement des arguments implicites

**Participants** : Alexandre Miquel, Hugo Herbelin.

**Mots clés** : argument implicite, système de types purs, synthèse de types, métathéorie.

La richesse du formalisme de Coq se paye souvent par une certaine lourdeur de la syntaxe, ce qui oblige l'utilisateur à fournir des informations de typage qui pourraient être inférées automatiquement par le système.

A. Miquel travaille sur un calcul avec arguments implicites, qui est essentiellement une variante du Calcul des Constructions dans laquelle une partie des informations de typage ne figure pas dans la représentation interne des termes, de manière à alléger la syntaxe au niveau de l'utilisateur. Il a poursuivi l'étude des propriétés théoriques de ce calcul, montrant notamment les propriétés de normalisation forte et de cohérence. Il a également effectué une étude sémantique de ce calcul qui a abouti à la construction d'un modèle basé sur les espaces cohérents, lesquels se sont révélés particulièrement adaptés à la modélisation des systèmes de types en présence de sous-typage. Ce travail a été présenté à la conférence TYPES'99.

### 6.6.2 Démonstration automatique en théorie des ensembles

**Participants** : Stéphane Vaillant, Gilles Dowek, Thérèse Hardin [Professeur Paris 6, en délégation à l'INRIA projet PARA].

La théorie des ensembles peut s'exprimer, entre autres, sous deux formes : au premier ordre avec des axiomes d'existence (e.g. 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 représentation naturelle des ensembles mais son inconvénient est que ce n'est plus une théorie du premier ordre car le symbole de compréhension est un lieur et de plus 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 et les méthodes de démonstration automatique du premier ordre ne peuvent donc pas s'appliquer.

Au cours de son stage de DEA [42], S. Vaillant a étudié une formalisation de la théorie des ensembles présentant l'avantage de l'utilisation du symbole de compréhension tout en restant au premier ordre. Cette présentation, basée sur les travaux de G. Dowek, Th. Hardin et C. Kirchner pour la théorie des types, est un codage de la théorie avec symbole de compréhension dans un langage du premier ordre par l'intermédiare d'une substitution explicite; le système de preuve associé est le calcul des séquents modulo la congruence associée à la théorie engendrée par le codage. Il en résulte que les preuves qui s'expriment dans cette présentation sont isomorphes à celles exprimées dans la théorie avec symbole de compréhension.

S. Vaillant poursuit cette étude dans son travail de thèse : cette nouvelle présentation à l'avantage d'avoir un nombre fini d'axiomes et semble plus naturelle d'utilisation que le système de von Neumann-Bernays-Gödel (premier ordre et ensemble fini d'axiomes).

### 6.6.3 Comparaison entre théorie des types et théorie des ensembles

**Participant** : Benjamin Werner.

**Mots clés** : type, ensemble.

B. Werner a poursuivi son étude des liens entre théorie des ensembles et théorie des

types [Wer97]. Avec Samuel Lacas, de l'université Paris 7, il a particulièrement étudié plusieurs variantes et traductions des constructions de Diaconescu, qui permettent de dériver le tiers-exclu de l'axiome du choix. Ils ont montré que ce phénomène pouvait également se produire en théorie des types. Ceci est bien sûr important dans des formalismes qui sont essentiellement constructifs. Ces travaux sont soumis à publication [41].

### 6.6.4 Contenu calculatoire du calcul des séquents

**Participant** : Hugo Herbelin.

**Mots clés** : appel par valeur, calcul des séquents.

Le résultat principal de ce travail est que les preuves du Calcul des Séquents peuvent s'interpréter comme des $\lambda$-termes mélangeant des aspects appel par nom et appel par valeur.

Dans sa thèse, H. Herbelin avait montré qu'un certain fragment LJT du calcul des séquents de Gentzen correspondait le $\lambda$-calcul en appel par nom. Il a étendu cette correspondance à un autre fragment, LJQ, qui lui, correspond au $\lambda$-calcul en appel par valeur, éclaircissant du même coup quelques zones d'ombre de la théorie $\lambda_V$ de G. Plotkin. Enfin, la superposition des $\lambda$-calculs par nom et par valeur correspond au calcul LJ en entier. Ces résultats s'étendent au cas de la logique classique.

### 6.6.5 Automatisation de preuves sur les constructeurs

**Participant** : Jean Goubault-Larrecq.

**Mots clés** : preuve automatique.

J. Goubault-Larrecq a poursuivi son étude d'une logique fondée sur les constructeurs et la récurrence, publiée à Tableaux'99 [28], et montré que le problème d'unification associé était indécidable, tout en proposant une procédure non terminante de résolution.

### 6.6.6 Logique modale S4 et substitutions explicites

**Participant** : Jean Goubault-Larrecq.

**Mots clés** : logique modale, substitution explicite.

Le travail de J. Goubault-Larrecq sur la logique modale S4 s'est concrétisé d'une part par l'acceptation de son article en commun avec Healfdene Goguen sur les calculs SKInT et SKIn à MSCS [27, 21], et d'autre part par de nouveaux résultats montrant les rapports étroits entre S4, les ensembles simpliciaux augmentés, et même le lambda-calcul calculatoire de Moggi : les résultats préliminaires ont été présentés au premier workshop sur les méthodes géométriques en théorie du parallélisme, au premier workshop sur les logiques modales intuitionnistes et leurs applications, et un article complet a été soumis [38].

---

[Wer97] B. Werner, « Sets in Types, Types in Sets », *in : TACS'97*, T. Ito et M. Abadi (réd.), *1281*, Springer-Verlag, 1997.

# 7 Contrats industriels (nationaux, européens et internationaux)

## 7.1 RNRT

Un projet exploratoire CALIFE (Environnement pour la Preuve formelle et le Test d'Algorithmes utilisés en Télécommunication) `http://www.lri.fr/CALIFE` 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 ALCATEL, CRIL, le CNET (France Telecom), l'INRIA, le LaBRI (Bordeaux), le LORIA, le LRI (Orsay) et le LSV (ENS Cachan).

Le projet COQ est impliqué principalement dans deux actions: la première vise à améliorer l'automatisation des preuves par l'utilisation de techniques de preuve réflexive, la seconde vise à mettre en place un prototype d'assistant à la démonstration reposant sur un calcul modulaire et modulo.

## 7.2 France-Telecom CNET

Nous avons une collaboration suivie avec P. Crégut et J.-F. Monin du CNET à Lannion. Un contrat de trois ans auquel participe le projet CROAP (U.R. Sophia-Antipolis) et portant sur le développement dans Coq d'outils de réutilisation de preuves et d'une bibliothèque permettant l'écriture de tactiques de haut niveau s'est achevé en février. Un nouveau 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 preuve a été établi avec le LRI.

# 8 Actions régionales, nationales et internationales

## 8.1 Actions nationales

### 8.1.1 Action de développement Génie

**Participants** : Jean-Christophe Filliâtre, Eduardo Giménez, Christine Paulin, Benjamin Werner.

Dans le cadre de la phase 2 du programme de recherche GENIE, Dassault-Aviation (en collaboration avec Aérospatiale, l'INRIA et VERILOG) veut montrer la faisabilité de la certification par preuve d'un générateur de code industriel utilisé pour le développement des logiciels critiques (commandes de vol Airbus, contrôle-commande de centrales nucléaires, ...). Il s'agit de certifier, à l'aide de Coq, l'outil SCADE de VERILOG, basé sur le langage Lustre développé par l'Université de Grenoble. E. Giménez a été embauché jusqu'en décembre 1999 par Dassault afin de participer à ce projet.

### 8.1.2 Action VIP du GIE Dyade

**Participants** : Gilles Dowek, Jean Goubault, Nabil EL Khadi, Christine Paulin, Muriel Roger, Eva 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). Les techniques utilisées ont été développées par Dominique Bolignano (G.I.E. Dyade) [Bol97]. Coq est utilisé comme outil pour formaliser ces preuves. Les principales difficultés concernent la recherche de représentations des problèmes dans Coq qui facilitent les preuves et la conception de techniques et d'outils de preuves permettant une automatisation partielle de la tâche.

### 8.1.3 Action de recherche concertée CFC

**Participants** : Judicaël Courant, Christine Paulin, Benjamin Werner.

Le projet Coq s'intéresse à la problématique du calcul formel certifié de par son expertise en preuves formelles et en certification de logiciel en général. Le langage du Calcul des Constructions Inductives rend compte uniformément des programmes et des preuves; il est ainsi un bon cadre pour étudier et concevoir les différents aspects de futurs systèmes qui allieraient l'expressivité et la sûreté des systèmes de preuves, et les performances du calcul formel. Ces questions sont l'objet de l'action de recherche concertée INRIA *Calcul Formel Certifié* à laquelle participent le projet CROAP (U.R. de Sophia-Antipolis) et l'équipe FOC de l'Université de Paris 6, cette action s'achève en décembre.

### 8.1.4 Action de recherche concertée Javacard

**Participants** : Jean Goubault, Benjamin Gregoire, Eva Rose.

L'action Javacard à laquelle participe les projets INRIA Cristal, Lande et Oasis ainsi que l'action Dyade VIP et l'ONERA-CERT étudie la sémantique formelle des langages Java et Javacard ainsi que des techniques d'analyse de programmes pour garantir la sécurité.

## 8.2 Actions européennes

### 8.2.1 Working Group TYPES

Le *Working Group* ESPRIT "TYPES" s'est terminé en septembre 99. Il portait sur le développement assisté par ordinateur de preuves et de programmes et regroupe principalement des équipes de Cambridge, Edinburgh, Göteborg, Helsinki, Manchester, Munich, Nijmegen, Oxford, Paris 7, Turin, Udine et Utrecht. G. Dowek fait partie du groupe chargé de la proposition du continuation de ce groupe.

[Bol97] D. Bolignano, « Towards the Formal Verification of Electronic Commerce Protocols », *in : IEEE Computer Security Foundations Workshop X*, 1997.

### 8.3 Actions internationales

#### 8.3.1 Pologne

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

#### 8.3.2 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 preuves interactives et par vérification de modèle avec une application à la certification du code embarqué dans des stimulateurs cardiaques.

### 8.4 Visites, et invitations de chercheurs

Dans le cadre de la collaboration avec l'Uruguay, Cristina Cornes a rendu visite au projet durant 3 semaines en septembre.

Kumar Neeraj Verma, étudiant de l'IIT Dehli en Inde a effectué un stage de 2 mois dans l'action VIP Dyade dans le cadre d'un accord de collaboration entre l'INRIA et l'Inde.

## 9 Diffusion de résultats

### 9.1 Animation de la Communauté scientifique

#### 9.1.1 Participation à des comités éditoriaux de revues

G. Huet fait partie des comités de lecture des revues "Journal of Symbolic Computation", "Journal of Applied Non-Classical Logics", "Journal of Logic and Computation", "International Journal on Foundations of Computer Science" et "Annals of Pure and Applied Logic".

#### 9.1.2 Participation à des comités éditoriaux de conférences

G. Dowek a été *associate chair* du comité de programme de la conférence *Logic in computer science* (LICS'99). Il a fait partie du comité de programme de la *Conference on automated deduction* (CADE'99) et *Theorem proving in higher-order logics* (TPHOLs'99).

J. Goubault-Larrecq a participé au comité éditorial de la conférence Tableaux'99 ainsi qu'à celui des *Journées francophones des langages applicatifs* (JFLA'99).

C. Paulin est membre des comités éditoriaux des conférences *Theorem proving in higher-order logics* (TPHOLs'99 et TPHOLs'00) et de la conférence *Principles and Practice of Declarative Programming* (PPDP'2000).

#### 9.1.3 Jurys

G. Dowek a été rapporteur des thèses de Olivier Pons (Conservatoire National des Arts et Métiers) et Nacira Chabane (Université Paris 6). Il a participé au jury de thèse de Matthieu Jaume (Ecole Nationale des Ponts et Chaussées).

C. Paulin a été rapporteur des thèses de Daniel Hirschkoff (Ecole Nationale des Ponts et Chaussées) et Line Jakubiec (Université de Provence). Elle a participé au jury de la thèse de Frédéric Prost (ENS Lyon). Elle a été rapporteur de l'habilitation à diriger les recherches de Yves Bertot (Université de Nice) et a participé au jury de l'habilitation de Karim Nour (Université de Savoie). Elle est rapporteur extérieur de la thèse de Conor Mc Bride (Université d'Edimbourg) et membre du "comité du manuscript" de la thèse de Milena Stefanova (Université Catholique de Nijmegen).

### 9.1.4 Distinctions

G. Huet est membre de "Academia Europaea" et membre correspondant de l'Académie des Sciences. Il est membre du "Honorary Board" du RIDS (Research Institute for Declarative Systems) à Nijmegen.

### 9.1.5 Vulgarisation Scientifique

Gilles Dowek a participé à la rédaction de deux ouvrages de vulgarisation *Quand la Science a dit ... c'est Impossible* [15] et *Paysages des Sciences* [17].

### 9.1.6 Manifestations scientifiques

G. Dowek et C. Paulin ont co-organisé avec Y. Bertot et L. Théry du projet Lemme à l'INRIA Sophia-Antipolis et A. Hirschowitz de l'université de Nice, la conférence internationale TPHOLs' 99 qui s'est tenue à Nice en septembre 1999.

C. Paulin a organisé avec Pierre Casteran de l'Université de Bordeaux une journée de "Coq users group meeting" lors de la conférence FM'99 qui s'est tenue à Toulouse en septembre 1999.

### 9.1.7 Responsabilités professionnelles

Depuis Janvier 1997, Gérard Huet est Délégué aux Relations Internationales à l'INRIA.

Gilles Dowek est *binôme scientifique* du chargé de communication de l'Unité de recherche de Rocquencourt et membre du *Réseau communication* de l'INRIA.

Gilles Dowek a participé à la rédaction d'un rapport sur *L'utilisation des nouvelles technologies pour l'accès de tous à la connaissance* pour l'*Académie des Sciences*.

## 9.2 Enseignement

### 9.2.1 DEA Sémantique, preuves et programmation

Ce DEA est cohabilité par l'Université Pierre et Marie Curie, l'Université Denis Diderot et l'Université Paris-Sud, l'École Normale Supérieure, l'École Normale Supérieure de Cachan, l' École Polytechnique et le Conservatoire National des Arts et Métiers, en convention avec l'INRIA.

G. Dowek a fait un cours de tronc commun *Preuves constructives* (20 heures) [33]. Ch. Paulin et B. Werner ont enseigné la théorie des types et l'utilisation de Coq dans un cours d'option *Logiques Constructives* (20 heures) de ce même DEA [36].

### 9.2.2 Ecole Nationale Supérieure des Techniques Avancées (ENSTA)

G. Dowek fait un cours de *Démonstration automatique* [32]. B. Werner enseigne le système Coq à raison de 20 heures dans un cours *preuves sur ordinateur*. J. Goubault-Larrecq fait un cours de 20 heures: *spécification et validation du logiciel* .

### 9.2.3 Ecole Polytechnique

Benjamin Werner est chef de travaux pratiques à l'Ecole Polytechnique, où il enseigne la programmation.

### 9.2.4 Encadrement doctoral

Gilles Dowek a soutenu son habilitation [13] en juin. Il encadre la thèse de M. Mayero.

G. Dowek et J.-P. Jouannaud encadrent la thèse de Jacek Chrząszcz effectuée au LRI à l'université Paris Sud en co-tutelle avec Paweł Urzyczyn de l'université de Varsovie.

J. Goubault encadre les thèses d'E. Rose et de M. Roger, il co-encadre la thèse de N. EL Kadhi, Université Tunis II, en collaboration avec Son Excellence Mohammed Ben Ahmed (ambassade de Tunisie au Royaume Uni), et Ahmed Mahjoub (directeur général, Tunisie Télécom).

H. Herbelin encadre la thèse de A. Miquel.

J.-P. Jouannaud encadre la thèse de Daria Walukiewicz-Chrząszcz effectuée au LRI à l'université Paris Sud en co-tutelle avec Jerzy Tiuryn de l'université de Varsovie.

C. Paulin a encadré la thèse de J.-C. Filliâtre (soutenue le 12 juillet 1999) elle encadre la thèse de P. Loiseleur et partiellement en collaboration avec Pierre Crégut celle de Cuihtlauac Alvarado au CNET Lannion.

B. Werner a encadré le travail de thèse de B. Barras (soutenue le 17 novembre 1999) il encadre le travail de thèse de D. Delahaye et co-encadre avec X. Leroy du projet CRISTAL celui de B. Grégoire.

### 9.2.5 Formation Coq

G. Dowek et C. Paulin ont donné 8 heures de cours sur le formalisme et l'utilisation de Coq lors d'une journée de formation organisée par l'ENS Cachan pour des ingénieurs EDF.

G. Dowek, H. Herbelin, C. Paulin et B. Werner ont organisé trois jours de formation à l'INRIA destinés en particulier aux utilisateurs de Coq au sein de Schlumberger et CP8.

### 9.2.6 Autres enseignements

G. Dowek, C. Paulin et B. Werner ont fait un cours à l'*École des Jeunes Chercheurs* du GDR *ALP* à Lille. J. Chrząszcz et D. Walukiewicz-Chrząszcz ont participé à cette école.

B. Werner a donné un cours de Coq (8 heures) à l'Ecole d'été "Theory and Practice of Formal Proofs", organisée à Giens par l'UR de Sophia-Antipolis dans le cadre du Working Group Esprit TYPES. P. Courtieu, J. Duprat, B. Grégoire et S. Vaillant ont participé à cette école.

Les enseignants chercheurs du projet ainsi que les doctorants dans le cadre du monitorat ou de vacations assurent des enseignements en premier et second cycle dans les universités de Paris 6, Paris 10, Paris 11 et Paris 12.

### 9.2.7 Participation à des colloques, séminaires, invitations

**Conférences internationales**

**TYPES'99** La plupart des membres du projet ont participé au workshop annuel TYPES'99 qui s'est tenu à Lökeberg (Suède) en juin 1999. G. Dowek, M. Mayero, A. Miquel et B. Werner y ont présenté leurs travaux. J.-C. Filliâtre a effectué une démonstration de Coq plus particulièrement centrée sur sa tactique `Correctness`.

**FLOCS'99** G. Dowek, H. Herbelin, J.-P. Jouannaud et C. Paulin ont participé en juillet 1999 à la *Federated logic conference* (qui regroupe *Logic in computer science*, *Conference on automated deduction*, *Rewritig techniques and applications* et *Computer aided verification*)

**TPHOLs'99** G. Dowek, J.-C. Filliâtre et C. Paulin ont participé à la conférence TPHOLs'99.

**FM'99** J. Goubault, J-C. Filliâtre et C. Paulin ont participé à la conférence *Formal Methods* (FM'99) à Toulouse en septembre 99. J-C. Filliâtre et J. Goubault ont présenté leurs travaux lors du "Coq users group meeting".

**Invitations** G. Dowek et J. Goubault ont été invités à exposer leurs travaux lors du symposium inaugural du laboratoire *Preuve, programmes et systèmes* à l'université Paris 7. G. Dowek y a présenté ses travaux sur l'élimination des coupures en déduction modulo et J. Goubault la formalisation de BDD en Coq.

Ch. Paulin a fait un exposé au groupe de travail IFIP WG 2.3 à Munich, (Allemagne) où elle était invitée comme *observer*.

B. Werner a été invité 10 jours à l'université de Bologne, où il a enseigné la théorie et la pratique du système Coq à raison de 25 heures.

## 10 Bibliographie

### Ouvrages et articles de référence de l'équipe

[1] S. Boutin, «Using reflection to build efficient and certified decision procedures», *in : TACS'97*, T. Ito, M. Abadi (éditeurs), *1281*, LNCS, Springer-Verlag, 1997.

[2] J. Courant, « A Module Calculus for Pure Type Systems », *in : TLCA'97*, *LNCS*, Springer-Verlag, p. 112 – 128, 1997, `http://www.ens-lyon.fr/~jcourant/papers/tlca97/tlca.ps.gz`.

[3] G. Dowek, T. Hardin, C. Kirchner, « Higher-order unification via explicit substitutions », *in : Logic in Computer Science*, p. 366–374, 1995.

[4] G. Dowek, « Third order matching is decidable », *Annals of Pure and Applied Logic, 69, pp. 135-155*, 1994.

[5] G. Dowek, «Le langage mathématique et les langages de programmation», *in : Voir, entendre, raisonner, calculer*, 1997.

[6] E. Giménez, *Un Calcul de Constructions Infinies et son application à la vérification de systèmes communicants*, thèse d'université, Ecole Normale Supérieure de Lyon, décembre 1996.

[7] 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.

[8] C. Muñoz, *Un calcul de substitutions pour la représentation de preuves partielles en théorie de types*, thèse de doctorat, Université Paris 7, novembre 1997.

[9] C. Parent, «Synthesizing proofs from programs in the Calculus of Inductive Constructions», *in : Third International Conference on the Mathematics of Program Construction*, *Lecture Notes in Computer Science*, 947, Springer-Verlag, juillet 1995.

[10] C. Paulin-Mohring, B. Werner, «Synthesis of ML programs in the system Coq», *Journal of Symbolic Computation 15*, 1993, p. 607–640.

## Livres et monographies

[11] Y. Bertot, G. Dowek, C. Paulin-Mohring, L. Théry (éditeurs), *International Conference on Theorem Proving in Higher Order Logics (TPHOLs'99)*, *LNCS*, *1690*. Nice, Springer-Verlag, septembre 1999.

## Thèses et habilitations à diriger des recherches

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

[13] G. Dowek, *La part du calcul*, Mémoire d'habilitation, Université Paris 7, juin 1999.

[14] J.-C. Filliâtre, *Preuve de programmes impératifs en théorie des types*, thèse de doctorat, Université Paris-Sud, juillet 1999, `http://www.lri.fr/~filliatr/ftp/publis/these.ps.gz`.

## Articles et chapitres de livre

[15] J.-M. Alimi, G. Dowek, L. Rolland, N. Farouki, P. Avocat, *Quand la Science a dit ... c'est Impossible*, Le Pommier, 1999.

[16] P. Audebaud, E. Zucca, «Deriving Proof Rules from Continuation Semantics», *Formal Aspects of Computing 3*, 1999, p. 1–21.

[17] C. Auffray, G. Dowek, J.-G. Ganascia, C. Houzel, A. Jacquard, E. Klein, P. Laszlo, P. Léna, J.-P. Poirier, *Paysages des Sciences*, Le Pommier, 1999, Sous la direction de Michel Serres et Nayla Farouki.

[18] F. Blanqui, J.-P. Jouannaud, M. Okada, « Inductive Data Type Systems », *Theoretical Computer Science*, 1999, à paraître.

[19] G. Dowek, «Collections, sets and types», *Mathematical structures in computer science 9*, 1999, p. 1–15.

[20] G. Dowek, *Handbook of Automated Reasoning*, Elsevier, 1999, ch. Higher-order unification and matching, à paraître.

[21] H. Goguen, J. Goubault-Larrecq, «Sequent Combinators: A Hilbert System for the Lambda Calculus», *Mathematical Structures in Computer Science 9*, 6, décembre 1999.

## Communications à des congrès, colloques, etc.

[22] F. Blanqui, J.-P. Jouannaud, M. Okada, «The Calculus of Algebraic Constructions», *in : RTA'99*, 1999.

[23] G. Dowek, T. Hardin, C. Kirchner, «HOL-$\lambda\sigma$: an intentional first-order expression of higher-order logic», *in : Rewriting Techniques and Applications (RTA)*, P. Narendran, M. Rusinowitch (éditeurs), *LNCS*, 1630, Springer-Verlag, p. 317–331, 1999.

[24] G. Dowek, B. Werner, «Proof normalization modulo», *in : Types for proofs and programs 1998*, T. Altenkirch, W. Naraschewski, B. Reus (éditeurs), *LNCS*, *1657*, Springer-Verlag, 1999.

[25] J.-C. Filliâtre, N. Magaud, « Certification of Sorting Algorithms in the System Coq », *in : Theorem Proving in Higher Order Logics: Emerging Trends*, 1999, `http://www.lri.fr/~filliatr/ftp/publis/Filliatre-Magaud.ps.gz`.

[26] J.-C. Filliâtre, «Proof of Imperative Programs in Type Theory», *in : Types for proofs and programs 1998*, T. Altenkirch, W. Naraschewski, B. Reus (éditeurs), *1657*, 1999.

[27] J. Goubault-Larrecq, «Conjunctive Types and SKInT», *in : Types for proofs and programs 1998*, T. Altenkirch, W. Naraschewski, B. Reus (éditeurs), *LNCS*, *1657*, Springer Verlag, Kloster Irsee, Allemagne, 1999. Version complète : rapport RR-3475, août 1998, INRIA, `http://www.inria.fr/RRRT/RR-3475.html`.

[28] J. Goubault-Larrecq, «A Simple Sequent System for First-Order Logic with Free Constructors», *in : Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-99)*, N. V. Murray (éditeur), *LNAI*, *1617*, Springer-Verlag, p. 202–216, Saratoga Springs, NY, USA, juin 1999. Version complète et étendue : rapport de recherche RR-3653, INRIA, 1999, `http://www.inria.fr/RRRT/RR-3653.html`.

[29] J.-P. Jouannaud, A. Rubio, « The Higher-Order Recursive Path Ordering », *in : Logic in Computer Science*, 1999, `ftp://ftp.lri.fr/LRI/articles/jouannaud/horpo.ps.gz`.

## Rapports de recherche et publications internes

[30] B. Barras, S. Boutin, C. Cornes, J. Courant, Y. Coscoy, D. Delahaye, D. de Rauglaudre, J. Filliâtre, E. Giménez, H. Herbelin, G. Huet, H. Laulhère, P. Loiseleur, C. Muñoz, C. Murthy, C. Parent, C. Paulin, A. Saïbi, B. Werner, *The Coq Proof Assistant Reference Manual – Version V6.3*, juillet 1999, `http://coq.inria.fr/doc-fra.html`.

[31] J. Courant, «$\mathcal{MC}$: A module calculus for Pure Type Systems», *Research Report n° 1217*, LRI, juin 1999, `http://www.lri.fr/~jcourant/papers/thesis/main.ps`.

[32] G. Dowek, *Démonstration automatique*, Ecole Nationale Supérieure des Techniques Avancées, 1999, Notes de cours.

[33] G. Dowek, *Théories des types*, DEA Programmation, 1999, Notes de cours.

[34] G. HUET, G. KAHN, C. PAULIN-MOHRING, *The* Coq *Proof Assistant - A tutorial - Version 6.3*, juillet 1999, `http://coq.inria.fr/doc-fra.html`.

[35] L. MOREAU, J. DUPRAT, « A Construction of Distributed Reference Counting », *rapport de recherche n° RR1999-18*, LIP, 1999, `ftp://ftp.ens-lyon.fr/pub/LIP/Rapports/RR/RR1999/RR1999-18.ps.Z`.

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

[37] K. N. VERMA, J. GOUBAULT-LARRECQ, « Reflecting BDDs in Coq », *rapport de recherche*, INRIA, décembre 1999.

## Divers

[38] J. GOUBAULT-LARRECQ, É. GOUBAULT, «On Intuitionistic S4 and Augmented Simplicial Sets», Élaboration de "Order-Theoretic, Geometric and Combinatorial Models of Intuitionistic S4 Proofs", présenté au "1st Workshop on Intuitionistic Modal Logics and Applications", Trente, Italie, ainsi qu'au "1st Workshop on Geometric Methods in Concurrency Theory", Aalborg, Danemark, 1999.

[39] B. GREGOIRE, *Certification en Coq de propriétés de sécurité issues du typage*, Rapport de DEA, Université Paris 7, 1999.

[40] N. E. KADHI, « Vers une Vérification Statique des Applets Cryptographiques », Soumis aux Journées Francophones des Langages Applicatifs 2000, septembre 1999.

[41] S. LACAS, B. WERNER, «Which choices lead to excluded middle?», Soumis à publication, 1999.

[42] S. VAILLANT, *Preuves en théorie des ensembles, une présentation au premier ordre*, Rapport de DEA, Université Paris 7, 1999.

[43] B. WERNER, «Reflecting traces», Soumis à publication, 1999.