INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team $\pi.r^2$

Design, study and implementation of languages for proofs and programs

Paris - Rocquencourt

Theme : Programs, Verification and Proofs

Table of contents

πr^2 is a common project with University Paris 7, within the laboratory "Preuves, Programmes et Systèmes", which is itself joint between Paris 7 and CNRS.. The team has been created on January the 1st, 2009 and became an INRIA "équipe-projet" on July the 1st, 2009.

1. Team

Research Scientist

Pierre-Louis Curien [Team leader, DR CNRS, HdR]
Hugo Herbelin [DR INRIA, HdR]

Faculty Member

Pierre Letouzey [MC Paris 7]
Yann Régis-Gianas [MC Paris 7]
Bruno Bernardo [ATER Paris 7]

Technical Staff

Vincent Gross [Ingénieur ADT Coq]

PhD Student

Stéphane Glondu [Allocation couplée, inscrit à Paris 7]
Danko Ilik [Allocation Gaspard Monge, Ecole Polytechnique]
Guillaume Munch [Elève ENS Lyon, inscrit à Paris 7]
Matthias Puech [Cotutelle avec l'Université de Bologne, financement Ministère de la Recherche italien (MIUR)]
Vincent Siles [Allocation couplée, inscrit à l'Ecole Polytechnique]
Élie Soubiran [Financement Ile de France, inscrit à l'Ecole Polytechnique]

Post-Doctoral Fellow

Jeff Sarnat [INRIA funding]
Alexis Saurin [INRIA funding]
Noam Zeilberger [Postdoc of the Foundation "Sciences Mathématiques de Paris"]

Visiting Scientist

Andreas Abel [Assistant Professor at LMU, Munich, visiting from October 1st, 2009 till March 31, 2010, funded by INRIA]

Administrative Assistant

Cécile Espiègle

2. Overall Objectives

2.1. Overall Objectives

πr^2 is a new joint INRIA team, which is devoted both to the study of foundational aspects of formal proofs and programs and to the development of the Coq proof assistant software, with a focus on the dependently typed programming language aspects of Coq. The team is part of the laboratory Proofs, Programs and Systems lab (PPS - UMR 7126, CNRS and Paris 7). Its explicit association with both University Paris 7 and CNRS is under way. The team acts as one of the strongest teams involved in the development of Coq as it hosts in particular the current coordinator of the Coq development team.

3. Scientific Foundations

3.1. Proof theory and the Curry-Howard correspondence

3.1.1. Proofs as programs

Proof theory is the branch of logic devoted to the study of the structure of proofs. An essential contributor to this field is Gentzen [40] who developed in 1935 two logical formalisms that are now central to the study of proofs. These are the so-called "natural deduction", a syntax that is particularly well-suited to simulate the intuitive notion of reasoning, and the so-called "sequent calculus", a syntax with deep geometric properties that is particularly well-suited for proof automation.

Proof theory gained a remarkable importance in computer science when it became clear, after genuine observations first by Curry in 1958 [34], then by Howard and de Bruijn at the end of the 60's [44], [63], that proofs had the very same structure as programs: for instance, natural deduction proofs can be identified as typed programs of the ideal programming language known as λ-calculus.

This proofs-as-programs correspondence has been the starting point to a large spectrum of researches and results contributing to deeply connect logic and computer science. In particular, it is from this line of work that Coquand's Calculus of Constructions [30] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [29].

3.1.2. Towards the calculus of constructions

The λ-calculus, defined by Church [28], is a remarkably succinct model of computation that is defined via only three constructions (abstraction of a program with respect to one of its parameters, reference to such a parameter, application of a program to an argument) and one reduction rule (substitution of the formal parameter of a program by its effective argument). The λ-calculus, which is Turing-complete, i.e. which has the same expressiveness as a Turing machine (there is for instance an encoding of numbers as functions in λ-calculus), comes with two possible semantics referred to as call-by-name and call-by-value evaluations. Of these two semantics, the first one, which is the simplest to characterize, has been deeply studied in the last decades [25].

For explaining the Curry-Howard correspondence, it is important to distinguish between intuitionistic and classical logic: following Brouwer at the beginning of the 20^{th} century, classical logic is a logic that accepts the use of reasoning by contradiction while intuitionistic logic proscribes it. Then, Howard's observation is that the proofs of the intuitionistic natural deduction formalism exactly coincide with programs in the (simply typed) λ-calculus.

A major achievement has been accomplished by Martin-Löf who designed in 1971 a formalism, referred to as modern type theory, that was both a logical system and a (typed) programming language [52].

In 1985, Coquand and Huet [30], [31] in the Formel team of INRIA-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [41], [61]. This formalism, called the Calculus of Constructions, served as logical foundation of the first implementation of Coq in 1984. Coq was called CoC at this time.

3.1.3. The Calculus of Inductive Constructions

The first public release of CoC dates back to 1989. The same project-team developed the programming language Caml (nowadays coordinated by the Gallium team) that provided the expressive and powerful concept of algebraic data types (a paragon of it being the type of list). In CoC, it was possible to simulate algebraic data types, but only through a not-so-natural not-so-convenient encoding.

In 1989, Coquand and Paulin [32] designed an extension of the Calculus of Constructions with a generalisation of algebraic types called inductive types, leading to the Calculus of Inductive Constructions (CIC) that started to serve as a new foundation for the Coq system. This new system, which got its current definitive name Coq, was released in 1991.

In practice, the Calculus of Inductive Constructions derives its strength from being both a logic powerful enough to formalise all common mathematics (as set theory is) and an expressive richly-typed functional programming language (like ML but with a richer type system, no effects and no non-terminating functions).

3.2. The development of Coq

Since 1984, about 40 persons have contributed to the development of Coq, out of which 7 persons have contributed to bring the system to the place it is now. First Thierry Coquand through his foundational theoretical ideas, then Gérard Huet who developed the first prototypes with Thierry Coquand and who headed the Coq group until 1998, then Christine Paulin who was the main actor of the system based on the CIC and who headed the development group from 1998 to 2006. On the programming side, important steps were made by Chet Murthy who raised Coq from the prototypical state to a reasonably scalable system, Jean-Christophe Filliâtre who turned to concrete the concept of a small trustful certification kernel on which an arbitrary large system can be set up, Bruno Barras and Hugo Herbelin who, among other extensions, reorganised Coq on a new smoother and more uniform basis able to support a new round of extensions for the next decade.

The development started from the Formel team at Rocquencourt but, after Christine Paulin got a position in Lyon, it spread to École Normale Supérieure de Lyon. Then, the task force there globally moved to the University of Orsay when Christine Paulin got a new position there. On the Rocquencourt side, the part of Formel involved in ML moved to the Cristal team (now Gallium) and Formel got renamed into Coq. Gérard Huet left the team and Christine Paulin started to head a Coq team bilocalised at Rocquencourt and Orsay. Gilles Dowek became the head of the team which was renamed into LogiCal. Following Gilles Dowek who got a position at École Polytechnique, LogiCal globally moved to Futurs with a bilocalisation on Orsay and Palaiseau. It then split again giving birth to ProVal. At the same time, the Marelle team (formerly Lemme, formerly Croap) which has been a long partner of the Formel team, invested more and more energy in both the formalisation of mathematics in Coq and in user interfaces for Coq.

After various other spreadings resulting from where the wind pushed former PhD students, the development of Coq got definitely multi-site with the development now realised by employees of INRIA, the CNAM and Paris 7.

We next briefly describe the main components of Coq.

3.2.1. The underlying logic and the verification kernel

The architecture adopts the so-called de Bruijn principle: the relatively small *kernel* of Coq ensures the correctness of the proofs validated by the system. The kernel is rather stable with modifications tied to the evolution of the underlying Calculus of Inductive Constructions formalism. The kernel includes an interpreter of the programs expressible in the CIC and this interpreter exists in two flavours: a customisable lazy evaluation machine written in Objective Caml and a call-by-value bytecode interpreter written in C dedicated to efficient computations. The kernel also provides a module system.

3.2.2. Programming and specification languages

The concrete user language of Coq, called *Gallina*, is a high-level language built on top of the CIC. It includes a type inference algorithm, definitions by complex pattern-matching, implicit arguments, mathematical notations and various other high-level language features. This high-level language serves both for the development of programs and for the formalisation of mathematical theories. Coq also provides a large set of commands and Gallina and the commands together forms the *Vernacular* language of Coq.

3.2.3. Libraries

Libraries are written in the vernacular language of Coq. There are libraries for various arithmetical structures and various implementations of numbers (Peano numbers, implementation of $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ with binary digits, implementation of $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ using machine words, axiomatisation of $\mathbb{R}$). There are libraries for lists, sorts, and for various implementations of finite maps and finite sets. There are libraries on relations, sets, orders.

3.2.4. Tactics

The tactics are the methods available to conduct proofs. This includes the basic inference rules of the CIC, various advanced higher level inference rules and all the automation tactics. Regarding automation, there are tactics for solving systems of equations, for simplifying ring or field expressions, for arbitrary proof search, for semi-decidability of first-order logic and so on. There is also a powerful and popular untyped scripting language for combining tactics into more complex tactics.

Note that all tactics of Coq produce proof certificates that are checked by the kernel of Coq. As a consequence, possible bugs in proof methods do not hinder the confidence in the correctness of the Coq checker. Note also that the CIC being a programming language, tactics can be written (and certified) in the own language of Coq if needed.

3.2.5. Extraction

Extraction is a component of Coq that maps programs (or even computational proofs) of the CIC to functional programs (in Objective Caml, Scheme or Haskell). Especially, a program certified by Coq can further be extracted to a program of a full-fledged programming language then benefiting of the efficient compilation, linking tools, profiling tools, ... of the target software.

3.3. Dependently typed programming languages

3.3.1. The emergence of dependently typed programming

Dependently typed programming (shortly DTP) is an emerging concept referring to the diffuse and broadening tendency to develop programming languages with type systems able to express program properties finer than the usual information of simply belonging to specific data-types. The type systems of dependently-typed programming languages allow to express properties *dependent* of the input and the output of the program (for instance that a sorting program returns a list of same size as its argument). Typical examples of such languages were the Cayenne language, developed in the late 90's at Chalmers University in Sweden and the DML language developed at Boston. Since then, various new tools have been proposed, either as typed programming languages whose types embed equalities (Ωmega at Portland, ATS at Boston, ...) or as hybrid logic/programming frameworks (Agda at Chalmers University, Twelf at Carnegie, Delphin at Yale, OpTT at U. Iowa, Epigram at Nottingham, ...).

DTP contributes to a general movement leading to the fusion between logic and programming. Coq, whose language is both a logic and a programming language which moreover can be extracted to pure ML code plays a role in this movement and some frameworks for DTP have been proposed on top of Coq (Concoqtion at Rice and Colorado, Ynot at Harvard, Why in the ProVal team at INRIA). It also connects to Hoare logic, providing frameworks where pre- and post-conditions of programs are tied with the programs.

3.3.2. Issues regarding dependently typed programming

DTP approached from the programming language side generally benefits of a full-fledged language (e.g. supporting effects) with efficient compilation. DTP approached from the logic side generally benefits of an expressive specification logic and of proof methods so as to certify the specifications. The weakness of the approach from logic however is generally the weak support for effects or partial functions.

3.3.2.1. Type-checking and proof automation

In between the decidable type systems of conventional data-types based programming languages and the full expressiveness of logically undecidable formulae an active field of research explores a spectrum of decidable or semi-decidable type systems for possible use in dependently programming languages. At the beginning of the spectrum, this includes for instance the system F's extension ML_F of the ML type system or the generalisation of abstract data types with type constraints (G.A.D.T.) such as found in the Haskell programming language. At the other side of the spectrum, one finds arbitrary complex type specification languages (e.g. that a sorting function returns a list of type "sorted list") for which more or less powerful proof automation tools (generally first-order ones) exist.

3.3.2.2. Libraries

Developing libraries for programming languages takes time and generally benefits of a critical mass effect. An advantage is given to languages that start from well-established existing frameworks for which a large panel of libraries exist. Coq is such a framework.

3.4. Around and beyond the Curry-Howard correspondence

For two decades, the Curry-Howard correspondence was limited to the intuitionistic case but in 1990, an important stimulus spurred on the community following the discovery by Griffin that the correspondence was extensible to classical logic. The community then started to investigate unexplored potential fields of connection between computer science and logic. One of these fields was the computational understanding of Gentzen's sequent calculus while another one was the computational content of the axiom of choice.

3.4.1. Control operators and classical logic

Indeed, a significant extension of the Curry-Howard correspondence has been obtained at the beginning of the 90's thanks to the seminal observation by Griffin [43] that some operators known as control operators were typable by the principle of double negation elimination ($\neg\neg A \Rightarrow A$), a principle which provides classical logic.

Control operators are operators used to jump from one place of a program to another place. They were first considered in the 60's by Landin [49] and Reynolds [60] and started to be studied in an abstract way in the 80's by Felleisen *et al* [37], culminating in Parigot's $\lambda\mu$-calculus [57], a reference calculus that is in fine Curry-Howard correspondence with classical natural deduction. In this respect, control operators are fundamental pieces of the full connection between proofs and programs.

3.4.2. Sequent calculus

The Curry-Howard interpretation of sequent calculus started to be investigated at the beginning of the 90's. The main technicality of sequent calculus is the presence of *left introduction* inference rules and two kinds of interpretations of these rules are applicable. The first approach interprets left introduction rules as construction rules for a language of patterns but it does not really address the problem of the interpretation of the implication connective. The second approach, started in 1994, interprets left introduction rules as evaluation context formation rule. This line of work culminated in 2000 with the design by Hugo Herbelin and Pierre-Louis Curien of a symmetric calculus exhibiting deep dualities between the notion of programs and evaluation contexts and between the standard notions of call-by-name and call-by-value evaluation semantics.

3.4.3. Abstract machines

Abstract machines came as an intermediate evaluation device, between high-level programming languages and the computer microprocessor. The typical reference for call-by-value evaluation of λ-calculus is Landin's SECD machine [48] and Krivine's abstract machine for call-by-name evaluation [47], [46]. A typical abstract machine manipulates a state that consists of a program in some environment of bindings and some evaluation context traditionally encoded into a "stack".

3.4.4. Delimited control

Delimited control extends the expressiveness of control operators with effects: the fundamental result here is a completeness result by Filinski [38]: any side-effect expressible in monadic style (and this covers references, exceptions, states, dynamic bindings, ...) can be simulated in λ-calculus equipped with delimited control.

4. Application Domains

4.1. The impact of Coq

Coq is one of the 8 most used proof assistants in the world. In Europe, its main challengers are Isabelle (developed in Munich, Germany), HOL (developed in Cambridge, UK) and Mizar (developed in Białystok, Poland).

Coq is used in various research contexts and in a few industrial contexts. It is used in the context of formal mathematics at the University of Nijmegen (constructive algebra and analysis), INRIA Sophia-Antipolis (number theory and algebra), INRIA-MSR joint lab (group theory), the University of Nice (algebra). It is used in France in the context of computer science at INRIA-Rocquencourt (certified compilation), INRIA-Saclay (certification of imperative programs), LORIA, Strasbourg (certification of geometry algorithms). Outside France, it is used in the context of computer science e.g. at U. Penn (formalisation of programming languages semantics), Yale, Ottawa and Berkeley Universities (building of a certified platform for proof-carrying code), University of Princeton (certified compilation), AIST at Tokyo (certification of cryptographic protocols), Microsoft Research Cambridge (proof of imperative programs), ... In the industry, it is used by Gemalto and Trusted Logic (JavaCard formal model and commercial applets certification).

All in all, it is difficult to evaluate how much Coq is used. Two indicators are the readership of the textbook on Coq by Yves Bertot and Pierre Castéran [26] and the number of subscribers to the Coq-club mailing list. More than 1200 copies of the book have been sold. There has been a second printing , and a Chinese translation of the book has been published. There are around 600 subscribers to the mailing list. Coq is taught or used for teaching in many universities: Paris, Bordeaux, Lyon, Nice, Strasbourg, CNAM, Nottingham, Ottawa, U. Penn, Warsaw, Krakow, Princeton, Yale, Berkeley, Rosario in Argentina, ...

5. Software

5.1. Coq

Participants: Bruno Barras [TypiCal team, Saclay], Yves Bertot [Marelle team, Sophia], Frédéric Besson [Lande team, Rennes], Frédéric Blanqui [Formes team, Beijing], Pierre Corbineau [University Joseph Fourier, Grenoble], Pierre Courtieu [CNAM], Jean-Christophe Filliâtre [ProVal team, Saclay], Julien Forest [CNAM], Stéphane Glondu, Benjamin Grégoire [Marelle team, Sophia], Vincent Gross, Hugo Herbelin [correspondant], Stéphane Lescuyer [ProVal team, Saclay], Pierre Letouzey, Assia Mahboubi [TypiCal team, Saclay], Claude Marché [ProVal team, Saclay], Julien Narboux [University of Strasbourg], Jean-Marc Notin [TypiCal team, Saclay], Russell O'Connor [University of Nijmegen], Christine Paulin [Proval team, Saclay], Loïc Pottier [Marelle team, Sophia], Matthias Puech, Vincent Siles, Élie Soubiran, Matthieu Sozeau [ProVal team and Harvard University], Arnaud Spiwack [TypiCal team, Saclay], Pierre-Yves Strub [Formes team, Beijing], Laurent Théry [Marelle team, Sophia], Benjamin Werner [TypiCal team, Saclay].

5.1.1. Version 8.2

The version 8.2 of the Coq system was released in February 2009. This version, coordinated by Hugo Herbelin, results from a collective work involving the INRIA teams πr^2, TypiCal, ProVal, and Marelle (with some help of Celtique, ex-Lande) and individuals from CNAM, University of Nijmegen, University Joseph Fourier and University of Strasbourg. It brings several significant extensions, most notably the "type classes", a powerful tool for inheriting structure and type inference borrowed from Haskell and extended to Coq by Matthieu Sozeau. The other main novelties concern arithmetics: development of a binary arithmetics (Spiwack), large numbers (Théry, Grégoire), abstract arithmetics (Besson). The proof-checker has been made autonomous (Barras).

The specific involvement of the team in this version concerns the integration work (Herbelin), the extension of the module system (Soubiran), the enrichment of the specification language and of the language of tactics (Herbelin, Letouzey, Sozeau), and the design and implementation of a more powerful type inference algorithm.

5.1.2. Towards version 8.3

Version 8.3 is planned for the beginning of 2010. The main purpose of version 8.3 is to make available to users various improvements of the features of Coq 8.2: an extended and generally more efficient module system (Soubiran), more tactics (Herbelin, Letouzey), more robust and efficient type classes (Sozeau), more efficient and comprehensive libraries (revision of the library of finite sets and of the library of abstract arithmetic by Pierre Letouzey, revision of the sorting library by Hugo Herbelin).

5.1.2.1. Graphical interface

The integrated graphical interface of Coq (CoqIDE) is under revision: Vincent Gross started to implement a new communication model based on process interaction rather than on threading (the reasons are: ability to support multiple Coq sessions, ability to interrupt Coq asynchronously, better robustness on non Unix-compliant operating systems, definition of a communication protocol reusable by other Coq interfaces).

5.1.2.2. Internal architecture of the Coq software

Pierre Letouzey's activities concerning the internal architecture of Coq includes the isolation of "plugins" parts that can now be dynamically loaded, the simplification of the current build infrastructure via Makefile, attempts to propose an alternative build infrastructure based on ocamlbuild, and code auditing via the Oug tool for finding useless or badly shaped code. Hopefully, these efforts (among which some are still in progress) contribute to having a software with a better design, more efficient, easier to maintain and extend.

5.1.3. The Technological Development Action Coq

This "Action de Développement Technologique", whose responsible is Hugo Herbelin, gathers the teams and individuals listed above.

Two national-level meetings have been organised as part of the ADT Coq. The first meeting has been organised by πr^2 in March 2009 and it gathered about 25 persons on the topic of automation in Coq. The second meeting has been organised by $TypiCal$ in June 2009 and it gathered about 15 persons on the question of tactic languages in Coq. The minutes of the meeting can be found at URL http://coq.inria.fr/adt.

The ADT Coq supported the first Coq Asian summer school that Jean-Pierre Jouannaud (Formes team) organised in August in Beijing. The ADT Coq also supported the first Coq workshop help in August in Munich. Chaired by Hugo Herbelin, the workshop, though announced lately, attracted 7 submissions of which the 9-persons program committee retained 6. About 25 persons attended. The web page of the workshop is http://coq.inria.fr/coq-workshop/2009.

5.1.4. Modules in Coq

Participant: Élie Soubiran.

During the first part of the year, Élie Soubiran has worked on an evolution of the Coq module system. In this work, he proposes a new framework where both module implementations and module types are unified into a single concept of theory. This new module system is equipped with three theory combinators that are inclusion, refinement and application. He also enriches this system with a new notion of Δ-equivalence that characterises in a decidable way the exact equalities between names. Hence by quotienting the name-space with this new equivalence, he provides to Coq a transparent name-space both for users and for the proof writing machinery (notations, proof search tactics, rewriting...). This work has been published in the proceedings of the MLPA workshop [16]. This module system is implemented in the development branch of Coq and will be available in the next public release. One can check the improvements brought by the new module system by looking at the new Structures, MSet and Numbers libraries.

Since October 2009, Élie Soubiran works on two improvements of the new module system. In the first one, he splits the primitive notion of theory into two atomic constructions of name-space and structure. This leads to a more general system where one can define not only modules but also extensible name-spaces or section[1] like name-spaces. His second improvement deals with a new merging of structure combinator that subsumes inclusion and refinement. This combinator helps, among others, to handle "diamond like" modular constructions.

5.1.5. Formalisation in Coq

Vincent Siles has done some Coq formalisation work on untyped PTS's and Sequent Calculus PTS's, which can be found at https://www.lix.polytechnique.fr/~vsiles/coq/formalisation.html.

[1] Section is a mechanism that allows to define local definitions and local parameters for a given set of proofs.

Stéphane Glondu is working with Mehdi Dogguy on the formalization in Coq of a type system for a timed π-calculus that guarantees confluence.

Hugo Herbelin implemented a few extensions of the proof language and certification language of the system. He also contributed the specification of a "mergesort" for the Coq library.

Matthias Puech integrated a record-inference mechanism in Coq for the recognition of mathematical structures.

5.2. Pangolin

Participant: Yann Régis-Gianas.

In collaboration with Johannes Kanig (PhD student, LRI/INRIA Proval/UPS), Yann Régis-Gianas has started the development of a new implementation of Pangolin, a call-by-value functional language that embeds a proof system, as described in his thesis. Indeed, a prototype implementation has already been released during Yann Régis-Gianas' PhD thesis. Yet, this prototype had some bugs, some intrinsic scalability problems and its underlying programming language was minimalistic.

To ensure correctness, the kernel of the new implementation will be extracted from a Coq development (which is a part of the metatheory proof mentioned above). A Coq-based validation process of program proofs will also be based on this Coq development.

To solve the scalability problems, the programming interface will be based on semantic patches, which should enable the reusability of proofs through evolutions of programs. The Pangolin prover should also improve the number of automatically discharged proofs.

To make the language as realistic as possible, standard programming mechanisms like exceptions, type coercion and type classes will be integrated in this new implementation.

5.3. Other software developments

Stéphane Glondu is involved in the maintenance of OCaml-related packages in Debian, which include OCaml itself, Coq, Ssreflect (an extension of Coq developed at INRIA-MSR joint center) and Ocsigen (a web framework developed at PPS). The Ubuntu distribution naturally benefits from this work. In collaboration with Stefano Zacchiroli, Mehdi Dogguy and Sylvain Le Gall, he developed a solution to enforce library linkability using inter-package relationships. This work will be presented at JFLA 2010.

In collaboration with François Pottier (INRIA Gallium), Yann Régis-Gianas maintained Menhir, an LR parser generator for Objective Caml.

6. New Results

6.1. Proof-theoretical investigations

Participants: Pierre-Louis Curien, Hugo Herbelin, Noam Zeilberger, Alexis Saurin, Guillaume Munch, Vincent Siles, Danko Ilik.

6.1.1. Sequent calculus and Computatonal duality

6.1.1.1. Axiomatisation of call-by-value

Hugo Herbelin and Stéphane Zimmermann (PPS) designed an original reduction semantics of call-by-value λ-calculus (with and without control) that is both complete with respect to the continuation-passing-style semantics of call-by-value and confluent. This has been presented to the 2009 edition of the TLCA conference [14].

6.1.1.2. Focalisation

Alexis Saurin has investigated how the focalization theorem of linear logic can be proved by interactive means in Girard's Ludics (in Terui's Computational Ludics setting [62]) , resulting in [2], which has since been improved and submitted to an international conference in early november [21]. Connections with algorithmic complexity are discussed since focalization in the framework of computational ludics can be connected with proof methods of the linear-speedup theorem [56].

In the two months he has been here, Noam Zeilberger worked on different aspects of computational duality and the Curry-Howard interpretation of polarized logic. With Paul-André Melliès and Jonas Frey from PPS, he began developing a categorical semantics of focusing proofs, which will eventually include a categorical account of abstract machines. He has also been studying polarity in intuitionistic logic, with the aim of giving a better explanation of, e.g., Goodman's Theorem, and especially of delimited continuations (an article is in preparation about the latter). Finally, he has been working on an article (with results from his dissertation) on the theory of refinement type systems (intersection and union types, subtyping, etc.) in the presence of effects.

6.1.1.3. Focalisation and classical realizability

Following the work of Curien and Herbelin [3] and Girard [42], Guillaume Munch–Maccagnoni gave a term syntax ($\mathsf{L}_{\mathrm{foc}}$) for polarised classical logic ($\mathsf{LK}_{\mathrm{pol}}$) and linear logic with strategies of reduction based on Girard's classical logic LC. This 'focalising' strategy distinguishes strict (positive) and lazy (negative) programs, and thus encompasses with a single deterministic strategy both call-by-value and call-by-name, in the spirit of Paul Blain Levy's call-by-push-value [51], however in direct style. Further syntactical investigations on this theme, establishing links with Zeilberger's work and with ludics, have been carried out in joint work between Pierre-Louis Curien and Guillaume Munch–Maccagnoni, and are to be written down for a special issue of the journal HOSC in honour of Peter Landin [12].

Guillaume Munch–Maccagnoni has extended Krivine's classical realisability [45] to $\mathsf{L}_{\mathrm{foc}}$, and therefore to CBV in particular. He showed that these tools concisely account for "imperfections" (Zeilberger) of programming languages with effect such as the value restriction required for the polymorphism in CBV to be a sound principle. This work led to a paper which was accepted to the CSL'09 conference during his Master internship [15].

Guillaume Munch–Maccagnoni then showed during his internship how classical realizability could further be seen as a tool (looking like Pitts' logical relations [59]) for the study of programming languages:

- Provided one accepts the Adequacy Lemma as an argument for type safety (as in [53]), it gives concise and modular proofs of type safety in the presence of polymorphism, sub-typing and inductive algebraic types, including in CBV.
- It can be used to derive certain parametricity results in the manner of Crary [33].
- Properties of 'internal completeness' allow one to prove additional equalities of types (such as the 'shocking equalities' of polymorphism).

Paul-André Melliès (from PPS) and Guillaume Munch–Maccagnoni started to work on the relationship between notions of category theory and features of $\mathsf{L}_{\mathrm{foc}}$, such as a possible link between dialogue categories and the cautious treatment of bilaterality in $\mathsf{L}_{\mathrm{foc}}$.

6.1.1.4. Pure type systems in sequent calculus

Hugo Herbelin and Vincent Siles investigated different formulations of pure type systems in sequent calculus building up on previous works by Hugo Herbelin [5] and Stéphane Lengrand [50], especially in connection to the problem of Expansion Postponement (see below). Vincent Siles summarised these investigations in a paper under consideration for submission.

6.1.2. On the logical contents of delimited control

6.1.2.1. Delimited control and $\Lambda\mu$-calculus

In the continuation of his work with Silvia Ghilezan [4] on showing that Saurin's variant $\Lambda\mu$ [7] of Parigot's $\lambda\mu$-calculus [58] for classical logic was a canonical call-by-name version of Danvy-Filinski's call-by-value calculus of delimited control, Hugo Herbelin studied with Alexis Saurin and Silvia Ghilezan another variant of call-by-name calculus of delimited control. This is leading to a general paper on call-by-name and call-by-value delimited control.

Saurin's calculus has the following surprising feature: its simply-typed version is equivalent to Parigot's $\lambda\mu$-calculus extended with a $\bot$ connective but its untyped version is much more expressive because it satisfies a property of completeness (Böhm separation) that $\lambda\mu$-calculus does not. A precise description of the connection between both calculi and between another call-by-name calculus of delimited control and another variant of $\lambda\mu$-calculus by de Groote has been conducted by Hugo Herbelin and Alexis Saurin resulting in a conditional acceptance to a special issue of the APAL journal on Computational Classical logic [20].

Alexis Saurin submitted two articles to international conferences since he joined the team: the first one [22] introduces a hierarchy of calculi for delimited control in call-by-name (that is a CBN correspondent to Danvy-Filinski's CPS hierarchy) and has been accepted to FOSSACS 2010 while the second one [23] establishes a standardization theorem and characterizes solvability for the $\Lambda\mu$-calculus [7] and introduces Böhm trees for $\Lambda\mu$-calculus. Those two works develop previous works by the author alone [7], [10], [8], [11].

Other current research work of Alexis Saurin, besides the on-going one mentioned above with Ghilezan and Herbelin, concerns a calculus for streams (in collaboration with Marco Gaboardi from Torino), an interactive approach to proof search using ludics [9], and the logical understanding of intersection types with Simona Ronchi della Rocca.

6.1.2.2. Control delimiters and Markov's principle

In the last months, Hugo Herbelin discovered a relation between control delimiters and Markov's principle: In an intuitionistic logic extended with classical logic for Σ^0_1-formulae (i.e. for formulae that correspond to data-types) and a control delimiter, Markov's principle (i.e. the property that $\neg\neg\exists x\, A(x) \to \exists x\, A(x)$) becomes provable while still retaining the main property of intuitionism, namely that any proof of $\exists x\, A(x)$ contains a witness t such that $A(t)$ holds.

6.1.2.3. Differential linear logic and a logical interpretation of statically bound exceptions

Guillaume Munch–Maccagnoni developed a symmetric syntax for Differential Linear Logic [36] inspired by the linear-non-linear adjunction of linear logic.

This syntax allowed him to give a logical interpretation of statically bound exceptions using primitives from differential linear logic.

In this interpretation, a co-dereliction on the catching context is used to model the fact that an exception binder can catch only one exception. The case where an exception is uncaught thus corresponds to the interaction between a dereliction and a co-weakening in differential linear logic.

As a direct consequence, Herbelin's implementation of Markov's principle on $\forall$-$\to$-free formulae using such exceptions corresponds, through this interpretation, to the validity of co-dereliction on recursively positive formulae.

6.1.3. Kripke semantics for classical logic

Hugo Herbelin and Gyesik Lee (ROPAS Center, Seoul University) presented their work on a simple proof of Kripke completeness for the negative fragment of intuitionistic logic to the 2009 edition of WOLLIC workshop [13]. This work provides an interesting case study on the representation of binders in Coq. This resulted in a paper submitted to the Special Issue on Binding, Substitution and Naming edited by C. Urban and M. Fernandez in the Journal of Automated Reasoning [19].

Hugo Herbelin, Danko Ilik, and Gyesik Lee gave a new kind of direct semantics for classical logic, similar to Kripke models, and proved constructively that it is sound and complete for first-order logic. They submitted a paper[18], which was conditionally accepted.

They also formalised the proofs in the Coq proof assistant, allowing them to do experiments on the operational behaviour of the semantics, confirming that the semantics gives rise to call-by-name proof normalisation.

Hugo Herbelin and Danko Ilik gave a constructive proof of completeness of intutionistic logic, with disjunction, with respect to a notion of model dual (call-by-value) to the one above. The proof, formalised in Coq, can be used as a normaliser for λ-calculus terms with sums. Work remains to be done in comparing the notion of model introduced to notions of models from the literature.

6.2. Metatheory of Coq and beyond

Participants: Andreas Abel, Vincent Siles, Bruno Bernardo, Yann Régis-Gianas, Hugo Herbelin.

6.2.1. Normalisation

Andreas Abel worked on the meta theory of the Calculus of Constructions (CoC). He partially encoded a normalization proof in Coq and submitted an article on normalization by evaluation for the CoC to the FLOPS 2010 conference.

Andreas Abel worked with Miguel Pagano on normalization by evaluation for Martin-Löf type theory with singletons and proof irrelevance and submitted a journal paper to the TLCA 2009 special issue. In discussions with Bruno Bernardo and Bruno Barras, he investigated the relationship between proof irrelevance and implicit quantification in the CoC. A manuscript is in preparation.

6.2.2. Unification in presence of subtyping

The core of the Calculus of Inductive Constructions (CIC, see Section 3.1.3) is a pure type system extended with a hierarchy of universes. The standard presentation of such type theories is "declarative", i.e. based on a notion of equivalence over programs, but, in practice, implementations, so as to have proof-checking decidable, need to be based on "syntax-directed" presentations. Fortunately, the CIC has a "syntax-directed" presentation that is equivalent to the declarative one (but whether the equivalence holds in general or not is a long-standing open problem called "Expansion Postponement").

Type theory with "typed equality" is the third main kind of way in which type theory can be presented. The connection between the "declarative" and "typed equality" presentations has been open for many years before being proved in 2006 by Adams [24] for a large set of type theories, called "functional". Unfortunately, the CIC is not functional and the result of Adams does not apply to Coq. Unfortunately also, the only known set-theoretical model of the CIC (this model justifies the consistency of Coq in the presence of standard mathematical axioms such as the extensional axiom of choice, i.e. it justifies that only "true" mathematical statements can then be proved in Coq) relies on the presentation of CIC with "typed equality". Extending the result of Adams to a larger class of type theories that contains the CIC is therefore crucial.

Eventually obtaining a correspondence between the "syntax-directed" presentation of the CIC and the presentation with "typed equality" is not only important for justifying the set-theoretic foundations of Coq. It is also important to support a new equality between programs that is called η-expansion and which says that any program of a functional type is indeed a function. Having η-expansion in Coq would make the system not only smoother to use from the user point of view: it would also open the way to the use of more powerful unification strategies for type inference and in particular to the use of Miller's pattern-unification.

6.2.2.1. Calculus of inductive constructions and typed equality

Hugo Herbelin and Vincent Siles showed that Adams' result extends to a category of systems that contains the CIC (the category of "full" type theory), henceforth bridging the gap between the standard presentation of the CIC and its typed presentation. This not only provides a more general solution to the long-standing problem of connecting type theory with typed equality to type theory with untyped equality in general: It also opens the way to a presentation of the CIC with η-expansion and hence the ability of formally studying unification algorithms in Coq.

Hugo Herbelin and Vincent Siles also worked on the problem of Expansion Postponement using a new approach based on "typed equality" that allows to rephrase the problem in new promising terms.

6.2.2.2. Implicit calculus of constructions

Bruno Bernardo is working on an Implicit Calculus of Constructions with dependent sums and with decidable type inference. In this calculus all the explicit static information (types and proof objects) is transparent and does not affect the computational behavior. Bruno Bernardo has already defined a formalism and studied an Implicit Calculus of Constructions [1]. Next step is to add Σ-types to the system by extending Alexandre Miquel's models based on coherence spaces [54] in order to prove the consistency and the strong normalisation property of the system.

This is joint work with Bruno Barras, researcher of the Typical team and PhD advisor of Bruno Bernardo.

6.2.3. Proofs of higher-order programs

Yann Régis-Gianas continued his collaboration with François Pottier (INRIA Gallium) about proofs of higher-order programs using Hoare Logic. They have submitted a long version of the paper "A Hoare Logic for Call-By-Value Functional Programs" [6] to a journal. An extension of this system with generalized algebraic datatypes and a machine-checked proof of its metatheory are in preparation.

In collaboration with Philippe Audebaud (Plume/ENS-Lyon) and Christine Paulin-Mohring (LRI/INRIA Proval/UPS), Yann Régis-Gianas worked on proofs of probabilistic programs. He has extended the Why [39] proof system with randomized primitives in the programming language and predicates over random distributions in the specification language.

Yann Régis-Gianas started some investigations about semantic patches, which are meta-programming operators meant to capture programming (or proving) development idioms. The purpose of this work is to use machine-checked programming language metatheory to design tools that track program (or proof) modifications and refactorize them automatically, when possible.

In collaboration with Hugo Heuzard (Master student, UPD), Yann Régis-Gianas worked on the mixing of two programming paradigms, namely functional reactive programming [55] and bidirectional programming [27], to develop user interfaces with built-in operators working on interaction history. (For instance, the standard "undo" action is such an operator.)

6.3. Coq as a functional programming language

Participants: Stéphane Glondu, Pierre Letouzey, Matthias Puech.

6.3.1. Certified libraries

In the second semester of 2009, thanks to the beginning of an INRIA "délégation" period, Pierre Letouzey has started a deep reform of some parts of the Standard Library of Coq, mainly the FSets library for finite sets and maps, and also the Numbers library of generic / efficient arithmetic. The idea is to take advantage of recent improvements of the Coq system in term of modularity (Type Classes by M. Sozeau and better Modules by E. Soubiran). This is meant to be the first steps on the way to a truly modular Standard Library for Coq, where properties and decision procedures would be shared amongst many structures, and in particular numerical datatypes. A particular attention is also taken to speed-up computations upon structures for instance with finite sets with (less / more isolated) proofs parts. This library redesign effort is still in preliminary form, but a good part of it should already appear in the forthcoming release of Coq 8.3.

Concerning decision procedures, Pierre Letouzey has supervised Lukasz Fronc (Master1 Internship), about the subject of a fully reflexive decision procedure for Presburger Arithmetic in Coq. The result of this internship was promising, and some more work needs to be done to finish and integrate it as a fully-operational tool for Coq.

6.3.2. Certified extraction

Stéphane Glondu summarized his work on formalizing the extraction in B. Barras's formalization of the Calculus of Inductive Constructions in [17].

This year, he worked mainly on the internal extraction. The goal of his work is to propose an extraction for the current Coq system, similar to the existing one, but that would also generate correctness proofs of the generated programs. The target language is ML, which is a source language considered by Z. Dargaye in [35]. The semantics of the target language is formalized in Coq, but the semantics of the source language—in which correctness proofs are stated—is left implicit.

Stéphane Glondu investigated several ways of defining the correctness of an extracted term, based on the type of its source term—atomic type constructions being inductive types and products. He also proved manually the correctness of the extraction of some basic functions involving recursion, logical preconditions, and re-use of previously defined functions (and their associated extracted term and proof). For this aim, he has designed a set of tactics to automatize boring parts of the proof (such as symbolic evaluation of ML terms). Proofs of extracted terms are not yet fully automated, and Stéphane Glondu is now making further investigations towards fully automatized generation of proofs (for some to-be-characterized terms).

6.3.3. Proof language

Matthias Puech worked on an inference mechanism in the proof assistant *Coq* for the recognition of mathematical structures. It involved the specification of a tool, integrated into the type theory, recognizing user-definable patterns in developments, and its efficient implementation as part of *Coq*'s distribution.

For this purpose, he developed an original data structure for term indexing, allowing fast retrieval of unifiers of a query term in a database of terms. When used in the type theory framework, it also allows the retrieval of instances or subterms of a lemma. It was devised, implemented and used as part of Coq's search facilities.

He is now working on the integration of this technology into the automated theorem proving tactics of *Coq*, to enhance both the efficiency and the expressiveness of these tactics: context-dependent proof search (as opposed to the current goal-directed proof search), shorter resulting proofs (avoiding useless η-expansions).

7. Other Grants and Activities

7.1. National Actions

Pierre Letouzey is member of the ANR "Decert" project. The objective of the "Decert" project is to design an architecture for cooperating decision procedures, with a particular emphasis on fragments of arithmetic, including bounded and unbounded arithmetic over the integers and the reals, and on their combination with other theories for data structures such as lists, arrays or sets. To ensure trust in the architecture, the decision procedures will either be proved correct inside a proof assistant or produce proof witnesses allowing external checkers to verify the validity of their answers.

In this prospect, Pierre Letouzey aims at integrating all results of this "Decert" project in the realm of the Coq proof assistant. Since this project is still in its early phase, there has been little such integration activity, but rather preliminary discussions. Pierre Letouzey also plans to clean up and improve the situation of some "historical" decision procedures of Coq, such as Omega for Presburger Arithmetic, since these historical tactics are currently in a quite unsatisfactory shape. Preliminary work has started.

7.2. EC projects

7.2.1. Accepted

Yann Régis-Gianas took part in the preliminary management of the EU-FP7 Certified Complexity project (CerCo). This European project will start on february 2010 as a collaboration between Bologna university (Asperti, Coen), Edinburgh university (Polack) and Paris Diderot university (Amadio, Régis-Gianas). The CerCo project aims at the construction of a formally verified complexity preserving compiler from a large subset of the C programming language to some typical micro-controller assembly language, of the kind traditionally used in embedded systems.

7.2.2. Submitted

Hugo Herbelin is the INRIA representative of the MathMate project submitted to the FP7 European programme. The project is about developing an intermediate language for representing formal mathematics on the web and combining mathematical natural language and checked proofs. The project is between Bologna university (Asperti), Stichting Katholieke Universiteit (Geuvers, McKinna, Wiedijk, Urban), Edinburgh (Fleuriot), FIZ Karlsruhe (Wegner, Teschke, Sperber) and INRIA (Formes and πr^2). Yann Régis-Gianas takes also part in this proposal.

8. Dissemination

8.1. Interaction with the scientific community

8.1.1. Collective responsibilities

Pierre-Louis Curien is deputy director of the Foundation "Sciences Mathématiques de Paris".

Hugo Herbelin coordinated the ADT Coq and the development of Coq.

8.1.2. Editorial activities

Pierre-Louis Curien is co-editor in chief of Mathematical Structures in Computer Science, and is an editor of Theoretical Computer Science and of Higher-Order and Symbolic Computation.

Pierre-Louis Curien is guest editor for a special issue of Logical Methods in Computer Science in connection with the conference TLCA 2009 (to appear in 2010).

8.1.3. Program committees and organising committees

Andreas Abel serves on the PC for FoSSaCS 2010.

Pierre-Louis Curien was the chairman of the conference Typed Lambda Calculus and Applications 2009 in Brasilia (July 2009), and is member of the PC of Logic Colloquium 2010. He is also taking part in the PC of the workshop Categorical logic, a satellite workshop of the joint conference MFCS & CSL 2010 in Brno (August 2010).

Hugo Herbelin organised the first Coq workshop held in Munich in August 2009 as a satellite event of the TPHOLs conference.

Alexis Saurin serves on the program committee for the workshop Games and Logic for Programming Languages (GaLoP V, http://perso.ens-lyon.fr/olivier.laurent/galop10), a satellite event of ETAPS 2010.

8.1.4. Jurys

In December 2009, Pierre-Louis Curien and Pierre Letouzey have been members of the "Comité de Sélection" for a position of "Maître de Conférences" at university Paris 13 Villetaneuse.

In 2009, Pierre Letouzey has been part of the jury of the entrance examination for the E.N.S. of Paris, Lyon, and Cachan ("MP" branch, "admissibilité"). He has been in charge of the written examination of Computer Science, he created the exam's content, and (with two colleagues) he graded the exam papers of the candidates.

8.1.5. Ph.D. and habilitation juries

Pierre-Louis Curien was a member of the jury of the thesis of Jesus Aranda (Ecole Polytechnique). He was a reviewer for the Habilitations of Antonio Bucciarelli (Paris 7), Krzysztof Worytkiewicz (Marseille), Daniel Hirschkoff (ENS Lyon) and Olivier Laurent (Paris 7).

8.2. Visits

8.2.1. Outbound

Andreas Abel visited C. Raffalli and P. Hyvernat (LAMA, Universite de Savoie, Chambery) to work on a termination checker for PML (2 weeks, December).

Pierre-Louis Curien has been an invited professor at Cambridge University (on a Leverhulme grant and on a fellowship of Emmanuel College) in April, May, and June. He has been Invited Professor at Tsinghua University, in September and October.

Hugo Herbelin has visited Korea (University of Seoul), China (Formes team at Tsinghua University, Beijing), and attended a workshop in Japan, in June.

Guillaume Munch visited Cambridge Computer Lab for two weeks in May.

8.2.2. Inbound

Zena Ariola (University of Oregon) visited πr^2 for a few days in September.

Josef Urban (University of Nijmegen) visited πr^2 for one week in October.

8.3. Teaching

8.3.1. Supervision of Ph.D. and internships

Pierre Letouzey is currently the PhD advisor of Stéphane Glondu. Moreover he has proposed and monitored the Master1 research internship (TRE, Travaux de Recherche Encadrée) of Lukasz Fronc, from February to June. The topic of this internship was "Full reflection for Presburger arithmetic in Coq".

Hugo Herbelin supervises the PhD of Élie Soubiran and Danko Ilik. He co-supervises the PhD of Vincent Siles with Bruno Barras (INRIA TypiCal team) and the beginning of the PhD of Matthias Puech (jointly with Andrea Asperti from Bologna University). He supervised the work of Vincent Gross, INRIA associated engineer for the ADT Coq.

Pierre-Louis Curien is the PhD advisor of Guillaume Munch (jointly with Thomas Ehrhard), and is also the supervisor of two students at PPS outside the πr^2 project (Stéphane Zimmermann, jointly with Thomas Ehrhard, and Alexis Goyet). Two of his PhD students (also outside πr^2) have defended their thesis in 2009: Christine Tasson (joint with Ehrhard) and Mauro Piccolo (joint with Silmona Ronchi, Torino).

8.3.2. Courses

Pierre-Louis Curien is teaching a 48 hours proof theory course in the Master program "Logique Mathématique et Fondements de l'Informatique" at Paris 7. He gave a 12 hour proof theory course at Tsinghua University, Beijing (September 2009).

Stéphane Glondu is a teaching assistant at University Paris Diderot-Paris 7. Vincent Siles is a teaching assistant at Ecole Polytechnique. During 2009, he taught Java programming (beginner / advanced) and introduction to networking. Since September 2009 Élie Soubiran holds a position of ATER at the university of Paris 12. He gives lessons and supervises practical works for three different teaching units: *Algorithmique avec ADA*, *Génie logiciel avec ADA* and *Projet de programmation en ADA*. Bruno Bernardo is teaching as an ATER in Paris 7 since September 2009.

8.4. Participation in conferences and seminars

8.4.1. Presentation of papers

Glondu: [17] at JFLA 2009.

Herbelin: [13] at the workshop WOLLIC'09 in Tokyo, and [18] at the informal workshop TYPES'09 in Aussois (France).

Munch: [15] at CSL 2009, Coimbra, and at the workshop Réalisabilité at Chambéry.

Soubiran: [16] at the MLPA workshop.

8.4.2. Other presentations

Herbelin presented the work done in Coq on certified SAT solvers at the second Workshop on Formal and Automated Theorem Proving and Applications (FATPA'09) held at the university of Belgrade (January).

8.4.3. Attendance to conferences, workshops, schools,...

Categorical Computer Science workshop in Grenoble, November (Munch).

Marktoberdorf Summer School on Logics and Languages for Reliability and Security (Glondu).

Journées Françaises des Langages Applicatifs, February, Grenoble (Letouzey).

Types 2009 in Aussois (Puech, Régis-Gianas, Siles).

CADE 22 (Soubiran).

MGS'09 spring school in Leicester, March (Puech).

Two meetings of the CeProMi AR, a working group about specifications and proofs of imperative higher-order programs (Régis-Gianas).

Monthly meetings of the ANR project CHOCO (Curry-Howard for concurrency) at Lyon (Munch, Curien).

Monthly meetings of a national level working group gathering researchers from PPS, IML (Marseille) and LIPN (Paris 13), on the Geometry of Interaction and its mathematical developments in the theory of operator algebras, following recent work of Girard (Curien, Saurin).

8.4.4. Talks in seminars

Abel: Talk on 15 October 2009 in the PPS seminar on normalization by evaluation for the Calculus of Constructions.

Curien: Talk at the Seminar of Cambridge Computer Lab, "What can sequent calculus do for programming languages?", June 2009.

Glondu: Gallium seminar.

Herbelin: seminar of the ROPAS group at the University of Seoul (June), seminar of the Formes team at the University Tsing Hua of Beijing (June), PPS seminar (September), on the current status of the foundations of Coq.

Ilik: Seminar of the Logic group, University of Padova,, presenting [18].

Munch: 'Réalisabilité à Chambéry' workshop (June), Groupe de travail 'Sémantique et réalisabilité' of PPS, Logic and Semantics seminar of Cambridge's Computer Laboratory.

Saurin: Prelude Workshop on september 28 (Prelude is an ANR project whose complete title is "Toward a Pragmatic Theory based upon the Ludics and the Continuations"), University of Turin (November), LaBRI, Bordeaux university, INRIA team SIGNES (November), meeting of the ANR PANDA project (December).

Siles: "Séminaire thésard" of LIAFA and PPS (organised by the PhD students).

Soubiran: "Groupe de Travail Programmation" (Paris 6 - Paris 7).

8.4.5. Groupe de travail Théorie des types et réalisabilité

This is one of the working groups of PPS, jointly organised by Hugo Herbelin and Paul-André Melliès, since September 2009. It is held weekly at the Antenne INRIA. So far, the working group has hosted talks of Noam Zeilberger, Pierre-Louis Curien, Andreas Abel, Hugo Herbelin, Alexis Saurin, Chung Kil Hur, François Pottier.

8.5. Other dissemination activities

Yann Régis-Gianas has organized the "Fête de la Science" event for the computer science department of the Paris Diderot university, with some financial support of INRIA Paris-Rocquencourt communication services. Pierre Letouzey and Guillaume Munch have also taken part in the organisation.

Yann Régis-Gianas has co-organized the “Journée Francilienne de Programmation”, a programming contest between undergraduate students of three universities of Paris (UPD, UPMC, UPS).

9. Bibliography

Major publications by the team in recent years

[1] B. BARRAS, B. BERNARDO. *The Implicit Calculus of Constructions as a Programming Language with Dependent Types*, in "FoSSaCS", 2008, p. 365-379, http://dx.doi.org/10.1007/978-3-540-78499-9_26.

[2] M. BASALDELLA, A. SAURIN, K. TERUI. *On the Meaning of Focalization*, in "(informal) Proceedings of Prelude Workshop", September 2009, http://www.pps.jussieu.fr/~saurin/Publi/BST-focalization-ludics.pdf JP .

[3] P.-L. CURIEN, H. HERBELIN. *The duality of computation*, in "Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP ’00), Montreal, Canada, September 18-21, 2000", SIGPLAN Notices 35(9), ACM, 2000, p. 233–243, http://doi.acm.org/10.1145/351240.351262.

[4] H. HERBELIN, S. GHILEZAN. *An Approach to Call-by-Name Delimited Continuations*, in "Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008", G. C. NECULA, P. WADLER (editors), ACM, January 2008, p. 383-394, http://doi.acm.org/10.1145/1328438.1328484RS.

[5] H. HERBELIN. *A Lambda-Calculus Structure Isomorphic to Gentzen-Style Sequent Calculus Structure*, in "Computer Science Logic, 8th International Workshop, CSL ’94, Kazimierz, Poland, September 25-30, 1994, Selected Papers", L. PACHOLSKI, J. TIURYN (editors), Lecture Notes in Computer Science, vol. 933, Springer, 1995, p. 61–75.

[6] Y. RÉGIS-GIANAS, F. POTTIER. *A Hoare Logic for Call-by-Value Functional Programs*, in "Proceedings of the Ninth International Conference on Mathematics of Program Construction (MPC’08)", Lecture Notes in Computer Science, vol. 5133, Springer, July 2008, p. 305–335, http://gallium.inria.fr/~fpottier/publis/regis-gianas-pottier-hoarefp.ps.gz.

[7] A. SAURIN. *Separation with Streams in the $\Lambda\mu$-calculus*, in "Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings", IEEE Computer Society, 2005, p. 356-365, http://dx.doi.org/10.1109/LICS.2005.48.

[8] A. SAURIN. *On the Relations between the Syntactic Theories of $\lambda\mu$-calculi*, in "Computer Science Logic 2008", LNCS, Springer, 2008.

[9] A. SAURIN. *Towards Ludics Programming: Interactive Proof Search*, in "ICLP 2008", LNCS, Springer, 2008, p. 253-268.

[10] A. SAURIN. *Une étude logique du Contrôle, appliquée à la programmation fonctionnelle et logique*, École Polytechnique, September 2008, Ph. D. Thesis.

[11] A. SAURIN. *Typing Streams in the $\Lambda\mu$-calculus*, in "ACM Transactions on Computational Logic", 2009, to appear.

Year Publications

Articles in International Peer-Reviewed Journal

[12] P.-L. CURIEN, G. MUNCH-MACCAGNONI. *The duality of computation under focus*, in "Higher Order and Symbolic Computation", 2010, in preparation for a special issue in honour of Peter Landin.

International Peer-Reviewed Conference/Proceedings

[13] H. HERBELIN, G. LEE. *Forcing-based cut-elimination for Gentzen-style intuitionistic sequent calculus*, in "Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-25, 2009, Proceedings", H. ONO, M. KANAZAMA, R. DE QUEIROZ (editors), Lecture Notes in Artificial Intelligence, vol. 5514, Springer, 2009, p. 209–217 KR .

[14] H. HERBELIN, S. ZIMMERMANN. *An Operational Account of Call-By-Value Minimal and Classical λ-calculus in "Natural Deduction" Form*, in "Ninth International Conference, TLCA '07, Brasilia, Brazil. July 2009, Proceedings", P.-L. CURIEN (editor), Lecture Notes in Computer Science, vol. 5608, Springer, 2009, p. 142–156.

[15] G. MUNCH-MACCAGNONI. *Focalisation and Classical Realisability*, in "Computer Science Logic '09", E. GRÄDEL, R. KAHLE (editors), Lecture Notes in Computer Science, vol. 5771, Springer-Verlag, 2009, p. 409–423.

[16] E. SOUBIRAN. *A unified framework and a transparent name-space for the Coq module system.*, in "Proceedings of the First International Workshop on Modules and Libraries for Proof Assistants (MLPA'09), Montreal, Canada", ELSEVIER ENTCS, August 2009, p. 28–42.

National Peer-Reviewed Conference/Proceedings

[17] S. GLONDU. *Extraction certifiée dans Coq-en-Coq*, in "JFLA 2009, Vingtièmes Journées Francophones des Langages Applicatifs, Saint Quentin sur Isère, France, January 31 - February 3, 2009. Proceedings", 2009.

Research Reports

[18] D. ILIK, G. LEE, H. HERBELIN. *Kripke Models for Classical Logic*, Inria, 2009, http://hal.inria.fr/inria-00371959/en/, Technical reportKR.

Other Publications

[19] H. HERBELIN, G. LEE. *Formalising Logical Meta-theory - Semantical Normalisation using Kripke Models for Predicate Logic*, 2009, submitted in November 2009 to APAL.

[20] H. HERBELIN, A. SAURIN. *$\lambda\mu$-calculus and $\Lambda\mu$-calculus, a capital difference*, 2009, submitted to APAL RS .

[21] K. T. MICHELE BASALDELLA. *From focalization of logic to the logic of focalization*, 2010, submitted in november 2009 to an international conference JP .

[22] A. SAURIN. *A Hierarchy for delimited control in call-by-name*, 2010, to appear in the proceedings of FOSSACS 2010.

[23] A. SAURIN. *Standardization and Böhm Trees for $\Lambda\mu$-calculus*, 2010, submitted in november 2009 to an international conference.

References in notes

[24] R. ADAMS. *Pure type systems with judgemental equality*, in "J. Funct. Program.", vol. 16, n^{o} 2, 2006, p. 219-246, http://dx.doi.org/10.1017/S0956796805005770.

[25] H. P. BARENDREGT. *The Lambda Calculus: Its Syntax and Semantics*, North Holland, Amsterdam, 1984.

[26] Y. BERTOT, P. CASTÉRAN. *Interactive Theorem Proving and Program Development Coq'Art: The Calculus of Inductive Constructions*, Springer, 2004.

[27] A. BOHANNON, J. N. FOSTER, B. C. PIERCE, A. PILKIEWICZ, A. SCHMITT. *Boomerang: Resourceful Lenses for String Data*, in "ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), San Francisco, California", January 2008, http://www.cis.upenn.edu/~bcpierce/papers/boomerang.pdf.

[28] A. CHURCH. *A set of Postulates for the foundation of Logic*, in "Annals of Mathematics", vol. 2, 1932, p. 33, 346-366.

[29] T. COQ DEVELOPMENT TEAM. *The Coq Reference Manual, version 8.2*, September 2008, http://coq.inria.fr/doc, Distributed electronically.

[30] T. COQUAND. *Une théorie des Constructions*, University Paris 7, January 1985, Dissertation.

[31] T. COQUAND, G. HUET. *Constructions : A Higher Order Proof System for Mechanizing Mathematics*, in "EUROCAL'85, Linz", Lecture Notes in Computer Science, vol. 203, Springer Verlag, 1985.

[32] T. COQUAND, C. PAULIN-MOHRING. *Inductively defined types*, in "Proceedings of Colog'88", P. MARTIN-LÖF, G. MINTS (editors), Lecture Notes in Computer Science, vol. 417, Springer Verlag, 1990.

[33] K. CRARY. *A Simple Proof Technique for Certain Parametricity Results*, in "ICFP", 1999, p. 82-89, http://doi.acm.org/10.1145/317636.317787.

[34] H. B. CURRY, R. FEYS, W. CRAIG. *Combinatory Logic*, vol. 1, North-Holland, 1958, §9E.

[35] Z. DARGAYE. *Vérification formelle d'un compilateur pour langages fonctionnels*, Université Paris Diderot-Paris 7, July 2009, http://gallium.inria.fr/~dargaye/these.pdf, Doctorat.

[36] T. EHRHARD, L. REGNIER. *The differential Lambda-calculus*, in "Theor. Comput. Sci.", vol. 309, n^{o} 1, 2003, p. 1–41, http://dx.doi.org/10.1016/S0304-3975(03)00392-X.

[37] M. FELLEISEN, D. P. FRIEDMAN, E. KOHLBECKER, B. F. DUBA. *Reasoning with continuations*, in "First Symposium on Logic and Computer Science", 1986, p. 131-141.

[38] A. FILINSKI. *Representing Monads*, in "Conf. Record 21st ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL'94, Portland, OR, USA, 17-21 Jan. 1994", ACM Press, 1994, p. 446-457.

[39] J.-C. FILLIÂTRE, C. MARCHÉ. *The Why/Krakatoa/Caduceus Platform for Deductive Program Verification*, in "19th International Conference on Computer Aided Verification, Berlin, Germany", W. DAMM, H. HERMANNS (editors), Lecture Notes in Computer Science, Springer-Verlag, July 2007, http://www.lri.fr/~filliatr/ftp/publis/cav07.pdf.

[40] G. GENTZEN. *Untersuchungen über das logische Schließen*, in "Mathematische Zeitschrift", vol. 39, 1935, p. 176–210,405–431.

[41] J.-Y. GIRARD. *Une extension de l'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types*, in "Second Scandinavian Logic Symposium", J. FENSTAD (editor), Studies in Logic and the Foundations of Mathematics, n^{o} 63, North Holland, 1971, p. 63-92.

[42] J.-Y. GIRARD. *A new constructive logic: Classical logic*, in "Math. Struct. Comp. Sci.", n^{o} 1, 1991.

[43] T. G. GRIFFIN. *The Formulae-as-Types Notion of Control*, in "Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL '90, San Francisco, CA, USA, 17-19 Jan 1990", ACM Press, 1990, p. 47–57.

[44] W. A. HOWARD. *The formulae-as-types notion of constructions*, in "to H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism", Academic Press, 1980, Unpublished manuscript of 1969.

[45] J.-L. KRIVINE. *Realizability in classical logic*, 2004, To appear in Panoramas et synthèses, Société Mathématique de France.

[46] J.-L. KRIVINE. *A call-by-name lambda-calculus machine*, in "Higher Order and Symbolic Computation", 2005.

[47] J.-L. KRIVINE. *Un interpréteur du lambda-calcul*, 1986, Unpublished.

[48] P. LANDIN. *The mechanical evaluation of expressions*, in "The Computer Journal", vol. 6, n^{o} 4, January 1964, p. 308–320.

[49] P. LANDIN. *A generalisation of jumps and labels*, n^{o} ECS-LFCS-88-66, UNIVAC Systems Programming Research, August 1965, Reprinted in Higher Order and Symbolic Computation, 11(2), 1998, Technical report.

[50] S. LENGRAND. *Normalisation & Equivalence in Proof Theory & Type Theory*, Université Paris 7 & University of St Andrews, 2006, Ph. D. Thesis.

[51] P. B. LEVY. *Call-by-Push-Value: A Subsuming Paradigm*, in "TLCA", 1999, p. 228-242, http://link.springer.de/link/service/series/0558/bibs/1581/15810228.htm.

[52] P. MARTIN-LÖF. *A theory of types*, n^{o} 71-3, University of Stockholm, 1971, Technical report.

[53] P.-A. MELLIÈS, J. VOUILLON. *Recursive Polymorphic Types and Parametricity in an Operational Framework*, in "20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)", IEEE Computer Society, 2005, p. 82-91, PPS//04/09//no30 (pp).

[54] A. MIQUEL. *Le Calcul des Constructions implicite: syntaxe et sémantique*, Université Paris 7, December 2001, Ph. D. Thesis.

[55] H. NILSSON, A. COURTNEY, J. PETERSON. *Functional Reactive Programming, Continued*, in "Proceedings of the 2002 ACM SIGPLAN Haskell Workshop (Haskell'02), Pittsburgh, Pennsylvania, USA", ACM Press, October 2002, p. 51–64.

[56] C. PAPADIMITRIOU. *Computational Complexity*, Addison Wesley, 1994.

[57] M. PARIGOT. *Free Deduction: An Analysis of "Computations" in Classical Logic.*, in "Logic Programming, First Russian Conference on Logic Programming, Irkutsk, Russia, September 14-18, 1990 - Second Russian Conference on Logic Programming, St. Petersburg, Russia, September 11-16, 1991, Proceedings", A. VORONKOV (editor), Lecture Notes in Computer Science, vol. 592, Springer, 1991, p. 361-380.

[58] M. PARIGOT. *Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction*, in "Logic Programming and Automated Reasoning: International Conference LPAR '92 Proceedings, St. Petersburg, Russia", Springer-Verlag, 1992, p. 190-201.

[59] A. M. PITTS. *Typed Operational Reasoning*, in "Advanced Topics in Types and Programming Languages", B. C. PIERCE (editor), chap. 7, The MIT Press, 2005, p. 245–289.

[60] J. C. REYNOLDS. *Definitional interpreters for higher-order programming languages*, in "ACM '72: Proceedings of the ACM annual conference, New York, NY, USA", ACM Press, 1972, p. 717–740.

[61] J. C. REYNOLDS. *Towards a theory of type structure*, in "Symposium on Programming", B. ROBINET (editor), Lecture Notes in Computer Science, vol. 19, Springer, 1974, p. 408-423.

[62] K. TERUI. *Computational Ludics*, in "Theoretical Computer Science", 2009, to appear.

[63] N. DE BRUIJN. *AUTOMATH, a language for mathematics*, n$^{\text{o}}$ 66-WSK-05, Technological University Eindhoven, November 1968, Technical report.