IN PARTNERSHIP WITH:
CNRS

Université Denis Diderot (Paris 7)

Activity Report 2012

Project-Team PI.R2

Design, study and implementation of languages for proofs and programs

IN COLLABORATION WITH: Laboratoire Preuves, Programmes et Systèmes

RESEARCH CENTER
Paris - Rocquencourt

THEME
Programs, Verification and Proofs

Table of contents

Project-Team PI.R2

Keywords: Programming Languages, Interactive Theorem Proving, Type Systems, Proofs Of Programs, Proof Theory

In April 2012, πr^2 has become officially an Equipe-Projet Commune, between Inria, Université Paris Diderot Paris 7, and CNRS.

From December 11, 2012, the team has moved to the Sophie Germain building of the university, which hosts the Mathematics and Computer Science departments of the university.

Creation of the Project-Team: January 01, 2009 *, Updated into Project-Team:* January 01, 2011 .

1. Members

Research Scientists
Pierre-Louis Curien [Team leader, DR CNRS, HdR]
Yves Guiraud [CR Inria]
Hugo Herbelin [DR Inria, HdR]
Alexis Saurin [CR CNRS]
Matthieu Sozeau [CR Inria]

Faculty Members
Pierre Letouzey [MC Paris 7]
Yann Régis-Gianas [MC Paris 7]
Philippe Malbos [MC Lyon 1, en délégation Inria]

Engineer
Xavier Clerc [Ingénieur Coq]

PhD Students
Pierre Boutillier [Allocation couplée, Paris 7]
Guillaume Claret [Allocation couplée, Paris 7, since October 2012]
Lourdes del Carmen Gonzalez Huesca [Funding by the ANR PARAL-ITP, since December 2011]
Guillaume Munch-Maccagnoni [Allocation couplée, Paris 7]
Pierre-Marie Pédrot [Allocation couplée, Paris 7]
Matthias Puech [Cotutelle avec l'Université de Bologne, ATER Paris 7]

Post-Doctoral Fellows
Federico Aschieri [Inria funding, until September 2012]
Nicolas Ayache [Postdoc of the european project "CerCo", until June 2012]
Jaime Gaspar [Postdoc of the Foundation SMP, since September 2012]
Arnaud Spiwack [Inria funding, until September 2012]
François Bobot [Funding by European project CERCO, October-December 2012]

Visiting Scientist
Zena Ariola [in sabbatical, from September 2012]

Administrative Assistant
Lindsay Polienor

Other
François Ripault [stagiaire ADT Coq, from September 2012]

2. Overall Objectives

2.1. Overall Objectives

The research conducted in πr^2 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 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.

Since 2012, the team has also extended its scope to the study of the homotopy of rewriting systems, which shares foundational tools with recent advanced works on the semantics of type theories.

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 [49] 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 [43], then by Howard and de Bruijn at the end of the 60's [56], [73], 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 [40] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [39].

3.1.2. Towards the calculus of constructions

The λ-calculus, defined by Church [38], 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 characterise, has been deeply studied in the last decades [34].

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 [62].

In 1985, Coquand and Huet [40], [41] in the Formel team of Inria-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [50], [68]. 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 [42] 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 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 well-delimited *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 OCaml 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. 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, list of a specified length, 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 OCaml, 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

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.

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.

3.3.1. 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. 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 [51] 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 [61] and Reynolds [67] and started to be studied in an abstract way in the 80's by Felleisen *et al* [45], culminating in Parigot's $\lambda\mu$-calculus [64], 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 [60] and Krivine's abstract machine for call-by-name evaluation [59], [57]. 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 [46]: 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, Harvard (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 [35] 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, Harvard, MIT, Princeton, Yale, Berkeley, Warsaw, Krakow, Rosario in Argentina, ...

Users of the assistant are also disseminating the use of the tool: A collaborative effort led by B. Pierce's team at U. Penn gave rise to a set of courses named Software Foundations (http://www.seas.upenn.edu/~cis500/current/sf/index.html) [66] on basic logic and computer science in Coq, that is used by many universities and individuals throughout the world. A. Chlipala wrote an advanced textbook on "Certified Programming with Dependent Types" in Coq, freely available on the web and soon to be published by MIT Press [37].

5. Software

5.1. http://coq.inria.frCOQ

Participants: Bruno Barras [TypiCal team, Saclay], Yves Bertot [Marelle team, Sophia], Pierre Boutillier, Xavier Clerc [SED team], Pierre Courtieu [CNAM], Maxime Dénès [Marelle team, Sophia], Julien Forest [CNAM], Stéphane Glondu [CARAMEL team, Nancy Grand Est], Benjamin Grégoire [Marelle team, Sophia],

Vincent Gross [Consultant at NBS Systems], Hugo Herbelin [correspondant], Pierre Letouzey, Assia Mahboubi [TypiCal team, Saclay], Julien Narboux [University of Strasbourg], Jean-Marc Notin [TypiCal team, Saclay], Christine Paulin [Proval team, Saclay], Pierre-Marie Pédrot, Loïc Pottier [Marelle team, Sophia], Matthias Puech, Yann Régis-Gianas, François Ripault, Matthieu Sozeau, Arnaud Spiwack, Pierre-Yves Strub [Formes team, Beijing], Enrico Tassi [TypiCal team, Saclay], Benjamin Werner [TypiCal team, Saclay].

5.1.1. Version 8.4

Version 8.4 was released in August 2012. It introduced a new proof engine designed and implemented by Arnaud Spiwack and a new extensive modular library of arithmetic contributed by Pierre Letouzey. It also included an extension of the underlying logic with "η-conversion" by Hugo Herbelin and "commutative-cuts compliant guard condition" by Pierre Boutillier, an extension of the pattern-matching compilation algorithm by Hugo Herbelin, an extension of the procedure of simplification of polynomial expressions by Loïc Pottier, a refinement of the type classes mechanism by Matthieu Sozeau, a new communication model by Vincent Gross for the graphical user interface CoqIDE, that Pierre Letouzey, Pierre Boutillier and Pierre-Marie Pédrot further extended.

Several users gracefully contributed improvements of various features (Tom Prince, Enrico Tassi, Daniel Grayson, Hendrik Tews, ...).

5.1.2. Graphical user interface

Pierre Letouzey has finalised and extended the work initiated by Vincent Gross (former ADT engineer) concerning the CoqIDE user interface: CoqIDE and Coq are now separate Unix processes, enhancing the reliability and improving the user experience.

In Fall 2012, Pierre Letouzey also revised the event infrastructure of CoqIDE, from a thread-based model to pure GTK event-loop. This way, CoqIDE is more reactive, less subject to deadlocks (especially under Windows), and the source code is more idiomatic and easier to understand. Interestingly, this work takes advantage of deeper notions such as C.P.S. (continuation passing style).

Pierre Boutillier and Pierre-Marie Pédrot built an abstract communication interface between Coq and CoqIDE based on XML syntax. They also refined the ability to customise CoqIDE. Pierre Boutillier made CoqIDE rely on Gtksourceview.

5.1.3. Proof engine

Arnaud Spiwack has proposed an extension of the expressiveness of tactics based on his previous work for a new proof engine. It allows for more atomic tactics, has a primitive support for backtracking, and allows for tactics which manipulate several goals.

5.1.4. Evaluation algorithms

Pierre Boutillier has proposed a new unfolding algorithm for global constants so that the definition of these ones are unfolded only if it triggers extra reductions. This helps users to keep goals concise during interactive proofs.

5.1.5. Type classes, internal representation

Matthieu Sozeau is adapting the type-classes mechanism to benefit from the new tactic engine and avoid reimplementing a whole proof-search algorithm with backtracking on top of the tactic language. This will bring high benefits in terms of efficiency and ease of use to the users. Forward proof-search for type classes was stabilised and is now used in libraries for better control on the search space, notably in the MathClasses library developed in Nijmegen [69].

An important shortcoming of type classes is the verbosity of the representation of projections from a class, as was illustrated in François Garillot's PhD thesis [48]. Matthieu Sozeau has developed a branch of Coq supporting an efficient representation of these projections based on the idea of bidirectional type checking which is now under stabilisation. This support will also enhance the performance of the assistant on developments using regular parameterised records and dependent sums like the HoTT library on homotopy type theory and the Forcing plugin developed by Sozeau *et al* [32].

5.1.6. Universes

While visiting the Institute for Advanced Study, Matthieu Sozeau implemented a new system of universe polymorphism that makes it possible to develop highly generic theories in the Coq system. Based on ideas from Harper and Pollack's [53] design of polymorphism as an elaboration in the Lego theorem prover, he developed an original algorithm for type inference of universes and implemented it in Coq. Its first application is inside the Homotopy Type Theory (HoTT) research program, as the formalisation of HoTT requires a high level of polymorphism that was not available before. Many other theories will benefit from this, including Sozeau's work on Forcing, B. Barras' (Typical) work on models of type theory or in the Math Classes library mentioned before. It also opens up possibilities to formalise category-theoretic notions without being limited by the universe system, a long-standing barrier in the Coq proof assistant.

5.1.7. The Equations plugin

Matthieu Sozeau continued the maintenance of the Equations plugin and developed a new Forcing plugin for Coq (see below).

5.1.8. Tools

Pierre-Marie Pédrot has written a program using the internal representation of libraries to compact Coq object files. It is based on well-known automata algorithms, representing memory as transition systems. The idea underlying this program is generalisable to any OCaml data structure, provided some conditions on its use are satisfied, and was formalised in a paper that was accepted at JFLA 2013.

5.1.9. Internal architecture of the Coq software

Pierre Letouzey continued a large reorganisation of the internal components of Coq, since these components are currently too much interdependent. This work brought better isolation between some of the Coq components and explicit interfaces between them. This allowed to simplify the compilation of Coq, since it is now easier to build the OCaml syntax extension used when compiling many advanced parts of Coq. Moreover, this clearer architecture should also help new contributors when they discover and interact with this large and complex code-base.

Pierre-Marie Pédrot also made some reorganisations of the code. This includes a clean generic library superseding the one of OCaml, pushing the CAMLP4/5 dependent parts out of the lower strata, as well as benefiting from the OCaml module system to get more uniformity in the naming of usual data structures.

Pierre Letouzey proposed a nicer backtracking infrastructure to Coq, used when the user wants to cancel some recent commands and go back before them. This new infrastructure unifies and improves what was used earlier by ProofGeneral and CoqIDE, the two main user interfaces for Coq.

Pierre Letouzey also dedicated many efforts to improve the support of the Windows platform by Coq.

5.1.10. Efficiency

Pierre-Marie Pédrot has been trying to optimise various parts of the Coq system, including the new tactical system designed by Arnaud Spiwack. Some neat tricks on garbage collection permitted to reach a substantial time improvement in compilation of object files. Various architectural modifications were also made in the process, like trying to get rid of the generic comparison in the code base.

Pierre Letouzey investigated an alternative implementation of the code dealing with Coq universes. These universes are a critical part of Coq: they have direct consequences on Coq safety, and handling them is time-consuming (between 10% to 20% depending on the Coq usage). This alternative implementation looks promising, but still requires some more work and stress-tests before being integrated in mainstream Coq.

5.1.11. Documentation generation

François Ripault and Yann Régis-Gianas developed a new version of coqdoc, the documentation generator of Coq. This new implementation is based on the interaction protocol with the Coq system and should be more robust with respect to the evolution of Coq.

5.1.12. General maintenance

Pierre Letouzey has been the main maintainer of Coq with extra contributions from Hugo Herbelin, Pierre Boutillier, Matthieu Sozeau, Pierre-Marie Pédrot, ...

5.1.13. Development Action

An "Action de Développement Technologique" about Coq started September 2011 and continued this year. It gathers the πr^2 team, the Marelle team and the CPR team from CNAM, Hugo Herbelin acting as the coordinator. It supports visits and meetings between developers and aims at strengthening the community of Coq users and contributors.

Yann Régis-Gianas set up an "osqa" server for Frequently Asked Questions.

The ADT Coq supported the internship of François Ripault.

Hugo Herbelin formalised a type-theoretic construction of semi-simplicial sets answering a problem raised early this year by Steve Awodey, Peter LeFanu Lumsdaine and others, in relation with the homotopy models of type theory.

5.2. Pangolin

Participant: Yann Régis-Gianas.

Yann Régis-Gianas maintained a prototype version of Pangolin. He used it to prove concrete complexity bounds for a set of functional programs using the method described in his FOPARA 2011 paper [19].

5.3. Other software developments

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

6. New Results

6.1. Proof-theoretical and effectful investigations

Participants: Federico Aschieri, Pierre Boutillier, Pierre-Louis Curien, Hugo Herbelin, Guillaume Munch-Maccagnoni, Pierre-Marie Pédrot, Alexis Saurin, Arnaud Spiwack.

6.1.1. Sequent calculus and computational duality

System L syntax. Pierre-Louis Curien studied in some detail the differences (and translations) between variants of "system L" syntax for polarised classical logic (developed by Guillaume Munch-Maccagnoni and himself):

- weakly focalised systems (where negatives can be worked on at any moment in a proof) versus focalised systems (where negative and positive phases alternate strictly), versus strongly focalised systems (where furthermore negative phases have to decompose negatives completely);
- systems where changes of polarity are implicit (like in Girard's LC) versus systems where they are explicitly marked using shift operators. These shift operators are formally adjoint, and as a matter of fact a suitable intuitionistic fragment of system L corresponds exactly to Levi's CBPV;
- systems with stoup (which retain only proofs that follow the focalisation discipline) versus (still focalised) systems without stoup (where the focalisation is forced by the dynamics of reduction);
- one-sided systems (with an implicit negation given by De Morgan duality) versus two-sided systems (allowing for explicit negation, and for distinguishing the left/right and the positive/negative dualities).

Pierre-Louis Curien is also currently studying a polarised version of a notion of general connective suggested earlier by Hugo Herbelin (unpublished work), and the composition structure of these connectives (in the spirit of operads).

Categorical semantics. Guillaume Munch-Maccagnoni investigated a notion of "direct style" for adjunction models, inspired by his work on polarisation in the "L" system, in the lineage of Führmann's [47] direct-style characterisation of monadic models. (It is part of joint work with Marcelo Fiore and Pierre-Louis Curien.)

Polarised Peano arithmetic. Guillaume Munch-Maccagnoni investigates the computational contents of polarised classical logic in arithmetic and in natural deduction. This allows him to compare the constructivisation of the principle $\neg\forall \Rightarrow \exists\neg$ based on classical realisability (Krivine) and the one based on delimited control (via "double negation shift"); both of which seem to be simplified by a better understanding of the "formulae-as-types" paradigm for a negation which is involutive in a strong sense.

Guillaume Munch-Maccagnoni investigates how a notion of classical realisability structure (inspired by Krivine's) can be used to prove properties of type systems which are usually regarded as syntactic.

Classical call-by-need and the duality of computation. In 2011, Zena Ariola, Hugo Herbelin and Alexis Saurin characterised the semantics of call-by-need calculus with control in the framework of *the duality of computation*. The same set of authors extended with Paul Downen and Keiko Nakata worked on abstract machines and continuation-passing-style semantics for call-by-need with control, resulting into a paper presented at FLOPS 2012 [20].

Further work has been done by Zena Ariola, Hugo Herbelin, Luís Pinto, Keiko Nakata and José Espírito Santo on typing the continuation-passing-style of call-by-need calculus, opening the way to a proof of normalisation of simply-typed call-by-need with control, and from there to a proof of consistency of classical arithmetic with dependent choice.

Zena Ariola also investigated how to formulate a parametric theory which encompasses call-by-value, call-by-name and call-by-need. Each theory is obtained by giving the appropriate definition of what is a value and a co-value. The theory also includes so called lifting axioms which allow one to relax the syntactic restrictions previously imposed on the call-by-value, call-by-name and call-by-need calculi. The theory also allows to include the η-rules which before were causing confluence to fail. The approach can be applied to natural deduction and this allows to express different embeddings of natural deduction into sequent calculus directly in the theory. The advantage of the new formalisation is that analogously to natural deduction, one can experiment with different strategies starting from the same term. Moreover, the theory is well-suited for continuation passing style transformation and, in particular, it leads to a different and simpler formalisation of classical call-by-need, its abstract machine and continuation passing style.

6.1.2. Dependent monads

Pierre-Marie Pédrot generalised the notion of monad in order to be able to use it in a dependent framework. This new structure allows to write effects in a pure functional language, such as Coq, through a monadic encoding.

This way, the whole monadic apparatus can be lifted to dependent programs, as well as proofs.

6.1.3. Linear dependent types

Arnaud Spiwack continued his investigations on dependently typed linear sequent calculus (based on Curien & Herbelin's $\mu\tilde{\mu}$). The current version of his system resembles Andreoli's focalised linear logic (yet to be published).

Pierre-Marie Pédrot has been working on a delimited CPS translation of the Calculus of Inductive Constructions, seen through the prism of polarised linear logic. Restricting dependencies to positives naturally fits into the scenery of delimited control, while extending negatives to infinitary objects permits to recover some properties of the involutivity of linear double-negation.

6.1.4. Proving with side-effects

Axiom of dependent choice. Hugo Herbelin showed that classical arithmetic in finite types extended with strong elimination of existential quantification proves the axiom of dependent choice. To get classical logic and choice together without being inconsistent is made possible first by constraining strong elimination of existential quantification to proofs that are essentially intuitionistic and secondly by turning countable universal quantification into an infinite conjunction of classical proofs evaluated along a call-by-need evaluation strategy so as to extract from them intuitionistic contents that complies to the intuitionistic constraint put on strong elimination of existential quantification. This work has been presented at LICS 2012 [22].

Memory assignment, forcing and delimited control. Hugo Herbelin investigated how to extend his work on intuitionistically proving Markov's principle [54] and the work of Danko Ilik on intuitionistically proving the double negation shift (i.e. $\forall x\,\neg\neg A \to \neg\neg\forall x\,A$) [15] to other kind of effects. In particular, memory assignment is related to Cohen's forcing as emphasised by Krivine [58] and by the observation that Cohen's translation of formula P into $\forall y \leq x \exists z \leq y\, P(z)$ is similar to a state-passing-style transformation of type P into $S \to S \times P$.

Hugo Herbelin then designed a logical formalism with memory assignment that allows to *prove* in direct-style any statement provable using the forcing method, the same way as logic extended with control operators allows to support direct-style classical reasoning. Thanks to the use of delimiters over "small" formulas similar to the notation of Σ^0_1-formulas in arithmetic, the whole framework remains intuitionistic, in the sense that it satisfies the disjunction and existence property.

Two typical applications of proving with side-effects are global-memory proofs of the axiom of countable choice and an enumeration-free proof of Gödel's completeness theorem.

The main ideas of this research program have been communicated during the Logic and Interaction weeks in Marseille in February 2012.

In the continuation of his work with Silvia Ghilezan [4] on showing that Saurin's variant $\Lambda\mu$ [8] of Parigot's $\lambda\mu$-calculus [65] 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.

Classical logic, stack calculus and stream calculus. Alexis Saurin studied the connection between the stack calculus recently proposed by Ehrhard et al and $\lambda\mu$-calculus and how the former can be precisely compared to the target of the CPS of the latter. He also investigated separation issues related to the stack calculus. During a visit to UPenn in the spring, Alexis Saurin and Marco Gaboardi investigated type systems for a stream calculus which contains $\Lambda\mu$.

Moreover, Alexis Saurin's paper *Böhm theorem and Böhm trees for the $\Lambda\mu$-calculus* [16] was published in TCS early 2012.

6.1.5. PTS and delimited control

From the study of one-pass CPS on the one side and of previous presentations of pure type systems with control operators on the other side, Pierre Boutillier and Hugo Herbelin have investigated how splitting terms into categories opens a new way to merge dependent types and monads. A preliminary set of rules has been presented during the third week of Logic and Interaction in Marseille.

It was refined since then but has not reached yet the maturity required to be accepted for publication in an international conference.

6.1.6. Interactive realisability

Thanks to the Curry-Howard correspondence for classical logic, it is possible to extract programs from classical proofs. These programs use control operators as a way to implement backtracking and processes of intelligent learning by trial and error. Unfortunately, such programs tend to be poorly efficient. The reason is that, in a sense, they are designed in order to keep their correctness and termination proofs simple. Each small modification of these programs seems, at best, to require major and difficult adaptations of their correctness proofs. This is due to a lack of understanding and control of the backtracking mechanism that interprets classical proofs. In order to write down more efficient programs, it is necessary to describe exactly: a) what the programs learn, b) how the knowledge of programs varies during the execution.

A first step towards this goal is the theory of Interactive Realisability, a semantics for intuitionistic arithmetic with excluded middle over semi-decidable predicates. It is based on a notion of state, which describes the knowledge of programs coming from a classical proof, and explains how the knowledge evolves during computation.

Federico Aschieri has extended the theory of interactive realisability to a full classical system, namely first-order Peano arithmetic with Skolem axioms. This is a very expressive system, with non-trivial axioms of choice and comprehension. The resulting programs are interpreted as stratified-learning algorithms, which build in a very organised way the approximations of the Skolem functions used in the proofs. The work has appeared in the proceedings of the conference Computer Science Logic 2012. A careful implementation of this extended theory –yet to be developed – will lead to a dramatic efficiency improvement over the already existing computational interpretations.

Federico Aschieri has also showed how to use interactive realisability to provide purely proof-theoretic results. He proved with a new method the conservativity of Peano arithmetic with Skolem axioms over Peano arithmetic alone for arithmetical formulas. In particular, the method can be seen as a constructivisation and substantial refinement of Avigad's forcing. The work has appeared in the proceedings of the workshop Classical Logic and Computation 2012.

6.1.7. Reverse mathematics

Hugo Herbelin explored with Gyesik Lee and Keiko Nakata the constructive content of the big five subsystems of second-order arithmetic considered in the context of (classical) reverse mathematics. They obtained a uniform characterisation of these systems in terms of variants of the comprehension axiom called separation, co-separation and interpolation.

This is the first step in a larger project attempting first to connect to predicative type theory the subsystems of System F underlying the proof-as-program structure of the big five subsystems of second-order arithmetic, and secondly to reformulate these subsystems in terms of pure systems of inductive definitions.

Jaime Gaspar has several projects running simultaneously. For example, in one of his projects he created a small unoptimised automated theorem prover, and he hopes to optimise it and use it to obtain a certain completely formalised proof to which he can apply a proof interpretation in order to extract computational content. As another example, in another project he his trying to show that several classical proof interpretations

are instances of a unified proof interpretation, in a parallel way to what is known for intuitionistic proof interpretations.

6.2. Type theory and the foundations of Coq

Participants: Pierre Boutillier, Pierre-Louis Curien, Hugo Herbelin, Pierre-Marie Pédrot, Yann Régis-Gianas, Alexis Saurin, Matthieu Sozeau.

6.2.1. Calculus of inductive constructions and typed equality

The work of Hugo Herbelin and Vincent Siles on the equivalence of Pure Type Systems with typed or untyped equality has been published [17].

6.2.2. Substitutions and isomorphisms

Pierre-Louis Curien completed his joint work with Martin Hofmann (Univ. of Munich) and Richard Garner (MacQuarie University, Sydney) on comparing two categorical interpretations of (extensional) type theories. More precisely, we wanted to compare two ways of giving a categorical interpretation of Martin-Löf type theory, both overcoming the following mismatch: syntax has exact substitutions, while their categorical interpretation, in terms of pullbacks or fibrations, "implements" substitutions only up to isomorphism. One can then either change the model (strictification) [55], or modify the syntax (by introducing explicit substitutions and more importantly explicit coercions between types that are now only isomorphic) [2]. In the latter case, one has to prove a coherence theorem to show that the interpretation is in the end independent from these coercion decorations. Such a proof was given in [2], using rewriting methods. These approaches turn out to be related through a general machinery that relates three kinds of categories, with strict or non strict objects and morphisms. As a bonus we get a new, more conceptual proof of coherence. These results are now being written up for a special issue in honour of Glynn Winskel. In further work, we wish to address intensional, and homotopy type-theoretic versions of these coherence problems.

6.2.3. Homotopy type theory

The univalence axiom proposed by Voevodsky states that for any two types to be equal exactly means being of same cardinality. This new axiom for type theory turns to have very interesting consequences for the practical foundations of formal mathematical reasoning: it smoothly implies other axioms such as functional extensionality or propositional existential but before all it says that any property proved about some mathematical structure immediately applies to any other other type ("sets" informally) which it is isomorphic to.

This axiom however contradicts the current logical foundations of Coq (in the presence of Streicher's axiom K). Investigations have then been started to understand how to weaken the Calculus of Inductive Constructions implemented in Coq so as to make it compatible with univalence. In a first step, this resulted in the design of a new rule for singleton elimination that has been implemented by Hugo Herbelin as an optional feature of Coq (singleton elimination is the ability to build objects in datatypes from canonically-proved propositional properties such as equality).

6.2.4. Models of type theory

The existing models of homotopy type theory are based on simplicial sets or on their extensions as Kan complexes. Hugo Herbelin developed a concrete type-theoretic formalisation of semi-simplicial sets following ideas from Steve Awodey, Peter LeFanu Lumsdaine and other researchers both at Carnegie-Mellon University and at the Institute of Advanced Study. The technique he used seems to straightforwardly generalise to provide type-theoretic constructions for arbitrary presheaves on inductively generated categories.

6.2.5. Forcing in type theory

Together with Nicolas Tabareau and Guilhem Jaber (Inria Ascola team, École des mines de Nantes), Matthieu Sozeau investigated an internalisation of the presheaf model of the Calculus of Inductive Constructions (CIC). They published their work at LICS'12 [23]. This work corresponds to adapting the idea of Forcing due to Cohen in Type Theory. An internal model construction allows to enrich the logical type theory with new modalities and define their semantics by translation to CIC. The usual Cohen forcing can be realised using this framework to show the independence of the continuum hypothesis in CIC, but more practical applications are possible as well. Notably, the step-indexed technique for building models of imperative languages with rich type structure can be phrased as a forcing/presheaf construction. Sozeau, Tabareau and Jaber developed a plugin that can handle this example [32] which relies on a modified Coq version implementing proof-irrelevance and eta-rules for records.

6.2.6. Proof irrelevance, eta-rules

Matthieu Sozeau continued his work on proof-irrelevance by implementing a variant of Werner's proof-irrelevant CIC in Coq [72]. An article describing this work is in preparation. The new system also handles the extensional eta-rules for records, extending the technique implemented by Hugo Herbelin to handle eta-expansion of functions in Coq.

6.2.7. Unification

The unification algorithm of Coq now essentially dwells in the λ-calculus part of the language. Pierre Boutillier started a refactoring of the code in order to deal with algebraic datatypes. Hugo Herbelin and Pierre Boutillier investigated how to reformulate unification on top of an abstract machine (i.e. on top of sequent calculus). Hugo Herbelin added various heuristics to the unification algorithm of Coq, making them both more powerful and customisable.

Matthieu Sozeau is continuing work in collaboration with Beta Ziliani (PhD student of Derek Dreyer at MPI Saarbrücken, two one week visits in 2012), and Aleksandar Nanevski (Researcher at IMDEA Madrid) on giving a clear formalisation for the unification algorithm of Coq. This will help understand better the working of advanced features like Canonical Structures and Type Classes that are heavily used in big developments, as the spectacular recently completed formalisation of the proof of Feit-Thompson's Odd theorem by the Mathematical Components team.

Matthieu Sozeau adapted the existing unification algorithm to be universe-aware, resulting in more predictability and earlier error-reporting in both the type inference and tactic unification algorithms of Coq.

6.3. Homotopy of rewriting systems

Participants: Pierre-Louis Curien, Yves Guiraud, Philippe Malbos.

6.3.1. Coherence in monoidal structures

Yves Guiraud and Philippe Malbos have applied the Squier's homotopical theorem [70], which they had generalised to higher-dimensional rewriting systems [52], to several types of categories with monoidal structures. This work develops a formal setting to produce constructive proofs of coherence conditions, applied to the cases of monoidal categories, symmetric monoidal categories and braided categories. These results have been published in Mathematical Structures in Computer Science [12].

6.3.2. Computation of resolutions of monoids

Yves Guiraud and Philippe Malbos have extended Squier's homotopical theory to the higher dimensions of presentations of monoids to get an algorithm transforming a convergent word rewriting system into a polygraphic resolution of the presented monoid, in the setting of Métayer [63]. From this polygraphic resolution, this work gives an explicit procedure to recover several of the known Abelian resolutions of the monoid, generalising and relating algebraic invariants of monoids. Moreover, a polygraphic resolution corresponds to the normalisation strategies of rewriting systems and they contain all the proofs of equality

between elements, together with the proofs of equality of those proofs of equality, and so on. This work has been published in Advances in Mathematics [13]. By nature, polygraphic resolutions bear many similarities with the higher-dimensional groupoids that appear in homotopical type theory when one considers the towers of identity types: this connection will be investigated by Pierre-Louis Curien, Yves Guiraud, Hugo Herbelin and Matthieu Sozeau.

6.3.3. Coherent presentations of Artin groups

With Stéphane Gaussent (Institut Camille Jordan, Université de Saint-Étienne), Yves Guiraud and Philippe Malbos are currently finishing an article on the rewriting properties and coherence issues in Artin groups, a class of groups that is fundamental in algebra and geometry. This work uses the formal setting of coherent presentations (a truncation of polygraphic resolutions at the level above relations) to formulate, in a common language, several known results in combinatorial group theory: one by Tits about the fundamental group of a graph associated to an Artin group [71], and one by Deligne about the actions of Artin groups on categories [44], both proved by geometrical and non-constructive methods. In this work, an improvement of Knuth-Bendix's completion procedure is introduced, called the homotopical completion-reduction procedure, and it is used to give a constructive proof of both those theorems. In fact, the method even improves the formerly known results: for example, it generalises Deligne's result to cases where his geometrical proof cannot be applied. A preliminary version of this work is available online [31]. The next objective of this collaboration is to extend the formal setting and methods to compute polygraphic resolutions of Artin groups, with a view towards two open problems of combinatorial group theory with respect to Artin groups: the decidability of the word problem and the verification that a precise topological space is a classifying space.

6.3.4. Higher-dimensional linear rewriting

With Samuel Mimram (CEA Saclay) and Pierre-Louis Curien, Yves Guiraud and Philippe Malbos investigate the links between set-theoretic rewriting theory and the computational methods known in symbolic algebra, such as Gröbner bases [36]. In particular, this work is interested in extending the setting of higher-dimensional rewriting to include "linear rewriting" and, as a consequence, to be able to apply its methods in symbolic computation. One particular direction is to understand Anick's resolution [33], and to improve it with the completion-reduction methodology, in order to get better algorithms to compute homological invariants and to prove important properties such as Koszulness. This research direction has been presented to the first call for projects of the IDEX Sorbonne-Paris-Cité, together with Eric Hoffbeck and Muriel Livernet (LAGA, Université Paris 13) and François Métayer (PPS, Université Paris 7).

6.4. Coq as a functional programming language

Participants: Nicolas Ayache, Pierre Boutillier, François Bobot, Guillaume Claret, Lourdes del Carmen Gonzalez Huesca, Tim Griffin, Hugo Herbelin, Pierre Letouzey, Matthias Puech, Yann Régis-Gianas, Matthieu Sozeau.

6.4.1. Type classes and libraries

Pierre Castéran from Inria Bordeaux and Matthieu Sozeau published a tutorial on the use of type classes [30] that was used as the basis of an invited lecture by M. Sozeau at the JFLA conference in February 2012. It will be published as part of the new version of the Coq'Art book.

6.4.2. Dependent pattern-matching

Pierre Boutillier experimented about how to integrate gently in Coq the compilation process he came up with last year to simulate Agda-style dependent pattern-matching. As a consequence, pattern grammar in Gallina has changed, much more notations can be used and users can write patterns instead of simple abstractions in the pattern-matching return clause.

Matthieu Sozeau continued maintenance and polishing of the Equations plugin that allows dependent pattern-matching on inductive families. A first official release is planned for the beginning of 2013.

6.4.3. Modularised arithmetical libraries

The modularised arithmetical libraries elaborated by Pierre Letouzey during the previous year(s) have been released as part of Coq 8.4. They provided greater uniformity of available functions and lemmas across the various Coq numerical datatypes. These libraries seem to work quite well, the only remaining issue is the documentation: due to this complex modular organisation, it is currently tedious for the user to browse the available functions and results. We expect to tackle this issue next year, by providing various documentation views, either the external summary of all available elements at once, or the internal layout of these elements.

6.4.4. Library of finite sets

Pierre Letouzey has integrated an additional Coq implementation in the MSets library of finite sets. This additional implementation is an improved version of the Red-Black-Tree library contributed by Andrew Appel. Using these RBT instead of the previously available AVL could be more efficient, at least in Coq, since they trigger no computations of integer numbers coding the tree depth.

6.4.5. Library on XPath processing

As part of the ANR Typex (http://typex.lri.fr), Matthieu Sozeau is developing a library for the certification of efficient XPath/XQuery engines in collaboration with Kim Nguyen (LRI) and Alan Schmitt (Inria Grenoble).

6.4.6. Mathematics of routing

Tim Griffin's primary focus during his visit to πr^2 was the development of a "metalanguage" for algebraic structures using Coq. Since he was something of a beginner with Coq, this involved learning the basics as well as more advanced work on representing algebraic structures. He made very good progress on this while in Paris and is now continuing this work in Cambridge.

6.4.7. Incrementality in proof languages

Matthias Puech and Yann Régis-Gianas worked on incremental type checking. This preliminary work has been presented during a contributed talk at TLDI 2012 [25]. It sets the ground for an incremental proof development and checking system, by means of a representation language for repositories of proofs and proof changes.

The traditional interaction with a proof-checker is a batch process. Coq (among others) refines this model by providing a read-eval print loop with a global undo system, implemented in an ad-hoc way. A more general approach to incrementality is being developed by means of a finer-grained analysis of dependencies. The approach developed is adaptable to any typed formal language: the language is specified in a meta-language close to the Edinburgh Logical Framework, in which subsequent versions of a development can be type-checked incrementally. Applications of this framework are: proof language for proof assistants, integrated development environments for proof or programming languages, typed version control systems.

6.4.8. Proofs of programs in Coq

As part of the CerCo European project, in collaboration with Roberto Amadio (PPS, Paris 7), François Bobot, Nicolas Ayache and Yann Régis-Gianas maintained a prototype compiler for a large subset of the C language whose specificity is to annotate input source programs with information about the worst-case execution cost of their machine code. Yann Régis-Gianas started a mechanised version of the proof technique used to prove the correctness of such an annotating compiler.

Yann Régis-Gianas maintained another compiler for Core ML that uses a generalisation of CerCo technique in order to obtain certified worst case execution time bounds on functional programs. This compiler produces proof obligations in Coq. The corresponding paper is published in January 2012 in the proceedings of FOPARA 2011 [19].

Nicolas Ayache developed a Frama-C plugin distributed in the CerCo software suite whose role is to synthesize cost annotations out of C programs. François Bobot developed a new version of this plugin. In particular, this new version handles C programs that manipulate pointers.

In collaboration with Roberto Amadio, Yann Régis-Gianas extended the cost annotating compilation chain of the FOPARA paper to handle the cost of memory management. A journal paper is about to be published in HOSC.

6.4.9. Lightweight proof-by-reflection

In the context of the ANR project Paral-ITP, Lourdes del Carmen Gonzalez Huesca, Guillaume Claret and Yann Régis-Gianas developed a new technique for proof-by-reflection based on a notion of *a posteriori* simulation of effectful computations in Coq.

7. Partnerships and Cooperations

7.1. National Initiatives

Matthieu Sozeau, Hugo Herbelin, Lourdes del Carmen Gonzalez Huesca and Yann Régis-Gianas are members of the ANR Paral-ITP started November 2011. Paral-ITP is about preparing the Coq and Isabelle interactive theorem provers to a new generation of user interfaces thanks to massive parallelism and incremental type-checking.

Hugo Herbelin is the coordinator of the PPS site for the ANR Récré accepted in 2011, which started in January 2012. Récré is about realisability and rewriting, with applications to proving with side-effects and concurrency.

Matthieu Sozeau is member of the ANR Typex project (Types and certification for XML) and is coordinator of one of the tasks of the project on formalisation and certification of XML tools. The project kicked-off on January 8th, 2012 and is a joint project with LRI, PPS and Inria Grenoble.

7.2. European Initiatives

7.2.1. FP7 Projects

Yann Régis-Gianas is a participant of the EU-FP7 Certified Complexity project (CerCo). This European project started in February 2010 as a collaboration between Bologna university (Asperti, Sacerdoti Coen), Edinburgh university (Stark) and Paris 7 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. François Bobot's postdoc is funded by this project.

7.2.2. Collaborations in European Programs, except FP7

Hugo Herbelin is participating to a PHC Pavle Savić with the university of Novi Sad in Serbia, the mathematical institute of Belgrade, ENS Lyon and the university of Turin. This project, called TLIT and headed by Silvia Ghilezan on the Serbian side, is about the properties of resource λGtz calculus; subject reduction for the $\overline{\lambda}\mu\widetilde{\mu}$-calculus; explicit substitutions and confluence; the diagrams and termination for ${}^*X$ calculus; introducing imperative features in classical logic; the $\lambda\mu$ calculus and its properties; the symmetries in classical logics.

Pierre-Louis Curien, Yves Guiraud and Philippe Malbos are collaborators of the Applied and Computational Algebraic Topology (ACAT) networking programme of the European Science Foundation.

7.3. International Initiatives

7.3.1. Inria Associate Teams

Title: Proof theory and functional programming languages (SEMACODE)

Inria principal investigator: Alexis SAURIN

International Partner:

Institution: University of Oregon (United States)

Laboratory: Computer and Information Science Department

Researcher: Zena ARIOLA

International Partner:

Institution: University of Novi Sad

Laboratory: Faculty of Engineering

Researcher: Silvia GHILEZAN

Duration: 2011 - 2013

See also: http://www.pps.univ-paris-diderot.fr/~saurin/EA-SEMACODE

Activity report: http://www.pps.univ-paris-diderot.fr/~saurin/EA-SEMACODE/en/activite.html

Cross-fertilisation between logic and programming languages theory is at the root of many striking developments in programming concepts as well as tools for formal analysis of programs. Our associated team project aims at gathering senior and young researchers from both sites in order to put a joint research effort on the following research themes: formalising particular evaluation strategies of functional languages based on logical techniques coming from sequent calculi. More specifically, we shall be interested in incorporating control operators directly in call-by-need and in developing a uniform framework for call-by-value and call-by-name calculi with delimited control, in particular to unveil the logical interpretation of delimited control (that is its logical counterpart with respect to Curry-Howard correspondence), and developing connections between delimited control and stream calculi; developing the logical content of realistic abstract machines and associated formal analysis tools for realistic abstract machines building on Curien-Herbelin $\overline{\lambda}$ calculi. The project will gather πr^2 expertise in proof theory and in the logical foundations of functional programming languages, the expertise of the Oregonian group on call-by-need evaluation and delimited control as well as respective crucial inputs of Gaboardi and Ghilezan on stream calculi, delimited control, semantics and type theory. The project will in particular allow to have the Inria and American students and post-docs involved in the project (7 out of 13 people involved) to travel between both sites and to organise joint workshops (one such workshop is planned in June 2011).

7.3.2. PHC

Hugo Herbelin started a PHC STAR with Gyesik Lee and Sungwoo Park in Korea on reverse mathematics and Coq, and on the role that polarisation can play in this respect.

7.3.3. Inria International Partners

πr^2 has strong relations with the following universities: Cambridge (Tim Griffin), Nottingham (Thorsten Altenkirch), Münich (Andreas Abel, Martin Hofmann), Strathclyde (Conor McBride), Chalmers in Göteborg (Thierry Coquand, Peter Dybjer), Technical University in Tallinn (Tarmo Uustalu, Keiko Nakata), Yale University (Zhong Shao), Harvard University (Greg Morrisett).

7.4. International Research Visitors

7.4.1. Visits of International Scientists

Thorsten Altenkirch (University of Nottingham) visited πr^2 for one month April 2012.

Conor McBride (University of Strathclyde) visited πr^2 for three weeks April-May 2012.

Keiko Nakata (University of Tallin) visited πr^2 for 4 days in September and worked with Zena Ariola and Hugo Herbelin on typing the continuation-passing-style semantics of call-by-need λ-calculus.

Tim Griffin visited πr^2 from January to June 2012 (and was funded 3 months by the Inria Paris-Rocquencourt invitation programme). He worked on the formalisation of routing protocols in Coq, and had many exchanges with Coq and ssreflect implementors.

Zena Ariola is visiting πr^2 during the whole academic year 2012-2013. She works on call-by-need, continuation-passing translations and related subjects.

Beta Ziliani (MPI Saarbrücken) visited πr^2 for one week in January 2012 and one week in July 2012 to work with Matthieu Sozeau on formalising the unification algorithm of Coq.

Jael Kriener (University of Kent) visited πr^2 for one week in January 2012 to work with Matthieu Sozeau on proof-search for Type Classes.

7.4.2. Internships

We host Paul Downen (PhD student of Zena Ariola, University of Oregon), during the entire academic year 2012/2013.

7.4.3. Visits to International Teams

Hugo Herbelin and Matthieu Sozeau have spent three months at the IAS as part of the special year on Univalent Foundations (October-December 2012).

7.4.4. Shorter International Visits Abroad

Pierre-Louis Curien visited Zena Ariola (Univ. of Oregon) for 2 weeks in May-June, and Tarmo Uustalu (Institute of Cybernetics of Technical University, Tallinn) for 2 weeks in December.

Hugo Herbelin visited Silvia Ghilezan at the University of Novi Sad in Serbia for one week in January 2012. He visited Predrag Janičić at the University of Belgrade for 2 days. He visited Danko Ilik in Skopje for one week.

Hugo Herbelin visited 4 days Gyesik Lee in Seoul and 3 days Sungwoo Park in Pohang in May as part of their joint project on Reverse Mathematics in Coq.

Guillaume Munch-Maccagnoni also visited Seoul (as part of the PHC STAR programme) 10 days in December.

Matthieu Sozeau was invited by the French Ministry of Foreign Affairs to visit Keiko Nakata and Tarmo Uustalu (IoC, Tallinn) for 4 days in June 2012. He gave a seminar on Equations there.

8. Dissemination

8.1. Scientific Animation

8.1.1. Collective responsibilities

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 Higher-Order and Symbolic Computation.

8.1.3. Program committees and organising committees

Pierre-Louis Curien was member of the PC for ICALP 2012.

Pierre-Louis Curien, Yves Guiraud, Philippe Malbos and Alexis Saurin have been members of the organising and scientific committee of the Logic and Interactions weeks held in the CIRM, Marseille, in the first trimester of 2012.

Yann Régis-Gianas served on the PC for the conference “Journees Francophones des Langages Applicatifs” (2013).

Hugo Herbelin was in the PC of the conference ITP 2012 and MFCS 2012. He was in the PC of the TYPES 2011 conference post-proceedings.

Hugo Herbelin organised the “Automation in Proof Assistant” satellite workshop of ETAPS 2012 which was shared with the organisation of the first COST 0901 meeting of year 2012.

Matthieu Sozeau was in the PC of the MSFP'12 (colocated with ETAPS'12) and LFMTP'12 (colocated with ICFP'12) workshops. He was in the PC of the PLPV'12 workshop, colocated with POPL'12, for which he was an external reviewer as well. Matthieu Sozeau is in the PC of JFLA'13.

8.1.4. Jury participation

In June and July 2012, Alexis Saurin has been member of the Jury for entrance exams at ENS Paris and examined the candidates for the three ENS for the “Informatique Fondamentale” Oral exam.

In September 2012, Alexis Saurin has been member of the Jury for LMFI Master.

8.1.5. Invited talks

Matthieu Sozeau gave an invited lecture on type classes at JFLA'12 in Carnac (January 2012) and at U. Penn, in October 2012.

Pierre-Louis Curien was invited speaker at the annual meeting of the GDR Topologie et Applications, Lille (“Interprétation catégorique de la théorie des types : questions de cohérence”), in October 2012.

Pierre Letouzey gave an invited talk in the ML workshop in Copenhagen in September 2012, about the interactions between Coq and OCaml.

Alexis Saurin gave an invited talk at the ASL annual meeting held in Madison, Wisconsin, in April 2012 (special session on Structural Proof Theory and Computing).

8.1.6. Presentation of papers

Matthias Puech has presented his paper [25] at the TLDI 2012 workshop.

Yann Régis-Gianas has presented the paper [21] at the MIPS 2012 workshop (best paper award).

Pierre Boutillier has presented his paper at JFLA 2012.

Hugo Herbelin and Matthieu Sozeau have presented their papers at LICS 2012 in Dubrovnik, Croatia.

Hugo Herbelin has presented his joint paper with Zena Ariola, Paul Downen, Keiko Nakata and Alexis Saurin at FLOPS 2012 in Kobe, Japan.

8.1.7. Other presentations

Pierre Boutillier gave a talk at Logic and Interaction weeks in February.

Pierre Boutillier gave a talk about compiling Agda style dependent pattern-matching in CIC during AIPA workshop in Tallinn.

Jaime Gaspar gave talks at Colloquium Logicum 2012, Paderborn, in September, and Collegium Logicum 2012: Structural Proof Theory, at École Polytechnique, in November.

Tim Griffin gave the following presentations: “Mathematics of Inter-domain routing", at Internet Engineering Task Force (IETF), Paris, March; “Routing in Equilibrium" for the "Paris Networking Group" (http://www.paris-networking.org/display_event.php?dispID=2221), Paris, and at Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL) (http://algotel2012.ens-lyon.fr), Montpellier.

Hugo Herbelin gave a talk on his proof-as-program interpretation of dependent choice in classical logic to the Réalisabilité à Chambéry #5 workshop in June.

Hugo Herbelin gave a talk on his proof with side-effects of Gödel's completeness theorem to the Workshop on Formal and Automated Theorem Proving and Applications (FATPA'12) held at the university of Belgrade.

Pierre-Marie Pédrot gave a presentation at the Logic and Interactions weeks in February, entitled "Double-glueing and orthogonality: refining models of linear logic through realizability".

8.1.8. Talks in seminars

Nicolas Ayache gave a talk about the achievements of the CerCo project at the seminar of the ENSTA, at the seminar of the EPITA and at the "GT programmation".

Pierre-Louis Curien gave a talk "De la logique classique (polarisée) à la logique linéaire (tensorielle)" at the Groupe de Travail Sémantique des calculs classiques, PPS, Paris 7, in November.

Pierre-Louis Curien gave talks on "System L syntax for sequent calculi" at the University of Bath, in February, at the École Polytechnique, in May, and at the Institute of Cybernetics, Tallinn, in December.

Jaime Gaspar gave talks at the Seminar of the Deducteam team, and at the Mathematical Logic Seminar, Lisbon.

Tim Griffin gave the following talks: "Routing in Equilibrium", University of Rome-3, April; "Routing in Equilibrium with help from ssreflect", Marelle team at École Polytechnique, May; "Mathematics of Routing", DIMACS working group on Abstractions for Network Services, Rutgers University, New Jersey, in May; "Mathematics of Routing", Université d'Évry, in June.

Yves Guiraud gave two talks on "Théorie de Squier et résolutions polygraphiques" at the Groupe de Travail Catégories supérieures, polygraphes et homotopie, PPS, Paris 7, in June.

Hugo Herbelin gave a talk on the formalisation of semi-simplicial sets in type theory at the Univalent Foundations seminar of the Institute for Advanced Study, Princeton, NJ, in December.

Guillaume Munch-Maccagnoni gave a talk about "Runnable monads in models of the λ calculus" at the ANR Logoi meeting in Luminy, in June.

Guillaume Munch-Maccagnoni gave a talk "On the computational contents of an involutive negation" at the Groupe de Travail Sémantique des calculs classiques, PPS, Paris 7, in November.

Pierre-Marie Pédrot gave a talk on a "Dependent delimited control monad" at the working group of the "Logique de la Programmation" team, from Institut Mathématique de Luminy, in December.

Matthieu Sozeau gave a talk in Tallinn about "Equations: A Dependent Pattern-Matching Suite" in June 2012.

Matthieu Sozeau gave a talk and a tutorial on "Coq with Classes" at the Institute for Advanced Study, Princeton, NJ in October and December 2012.

Matthieu Sozeau gave a talk about "Universe Polymorphism and Inference in Coq" at the Institute for Advanced Study, Princeton, NJ in December 2012.

8.1.9. Attendance to conferences, workshops, schools,...

Pierre Boutillier attended the Journées Francophones des Langages Applicatifs, Carnac, in February.

Tim Griffin attended the ssreflect/Inria MAP Spring School, Sophia Antipolis, in March.

Hugo Herbelin attended the FATPA workshop at the University of Belgrade in January.

Hugo Herbelin attended the ETAPS 2012 conference and the AIPA workshop in Tallinn in March-April.

Hugo Herbelin attended the FLOPS 2012 conference in Kobe in May.

Hugo Herbelin attended the ITP 2012 conference at Princeton University in August.

Hugo Herbelin attended the Coq workshop 2012 at Princeton University in August.

Guillaume Munch-Maccagnoni attended APLAS, Kyoto, in December.

Matthias Puech attended POPL 2012 in January.

Yann Régis-Gianas attended FMICS 2012, Paris, in August.

Matthieu Sozeau attended the POPL 2012 conference in January.

Matthieu Sozeau attended the JFLA 2012 conference in February.

8.1.10. 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. Yves Guiraud took over from Hugo Herbelin for the organisation of the seminar while Hugo was at IAS (October-December).

The web site is http://www.pps.univ-paris-diderot.fr/gdt-types-realisabilite. The programme in 2012 included talks by Pierre-Louis Curien, Arnaud Spiwack, Hugo Herbelin, Matthias Puech, Pierre Boutillier (twice), Federico Aschieri, Jaime Gaspar, Pierre-Marie Pédrot, François Bobot.

8.2. Teaching - Supervision - Jury participation

8.2.1. Teaching

(We do not include here the standard teaching duties of Pierre Letouzey and Yann Régis-Gianas as “Maîtres de conférences”.)

Licence: Pierre Boutillier is teaching assistant at University Paris 7. He taught this year Unix-system (beginner), algorithms and databases (beginner in the bioinformatics master).

Licence: Matthias Puech has a temporary research and teaching position (A.T.E.R) at University Paris 7 for the academic year 2012–2013. He teaches this year logic (L2) and functional programming (L3) and abstract machines (L3).

Master: Yann Régis-Gianas took part in the MPRI course entitled “Type systems”: he gave a 12 hours course about generalised algebraic data types, higher-order Hoare logic and dependently typed programming.

Master: Alexis Saurin taught the LMFI course entitled “Lambda-calcul et preuves” together with Thomas Ehrhard (January-May). Each of them taught half of this 48H master course.

Master: Pierre-Louis Curien gave a 10 hours course at the Department of Computer Science of the University of Oregon on proof theory (May).

Doctorat: Pierre-Louis Curien gave lectures at the 11th Oregon Summer School on Programming Languages (July).

Training: Pierre Letouzey participated to a training program for high-school teachers, introducing them to computer science. These teachers are meant to participate later to the new “Spécialité” ISN (Informatique et Sciences du Numérique) in “Terminale” grade.

8.2.2. Supervision

Internship: Yann Régis-Gianas supervised the internship of François Ripault.

Internship: Yann Régis-Gianas supervised the internship of Arthur Guillon.

PhD: Matthias Puech, Certificates for incremental type checking, Università di Bologna et Université Paris 7, December 2012, Hugo Herbelin, Andrea Asperti, and Yann-Régis Gianas.

PhD in progress: Pierre Boutillier, Représentation des effets et inférence de type dans le cadre du développement d’un langage de programmation à types riches, September 2010, Hugo Herbelin.

PhD in progress: Lourdes del Carmen Gonzalez Huesca, Un langage de tactiques typées pour Coq, December 2011, Hugo Herbelin and Yann-Régis Gianas.

PhD in progress: Guillaume Munch-Maccagnoni, Polarités, réalisabilité classique, contrôle délimité, September 2009, Pierre-Louis Curien and Thomas Ehrhard.

PhD in progress: Pierre-Marie Pédrot, Types dépendants et linéarité, October 2012, Hugo Herbelin and Alexis Saurin.

PhD in progress: Guillaume Claret, October 2012, Yann Régis Gianas.

8.2.3. PhD or habilitation jury participation

Pierre-Louis Curien was a referee for the PhD theses of Martin Churchill (“Imperative programs as proofs via game semantics”, Univ. of Bath), Benedikt Ahrens (“Initiality for typed syntax and semantics”, Université Nice Sophia Antipolis), Silvia Vecchiato (“Semantics of CBN dual proofs into control categories”, Università di Siena), and Thomas Seiller (“Logique dans le facteur hyperfini: Géométrie de l’interaction et complexité”, Université Aix-Marseille). He was member of the habilitation jury of Micaela Mayero (“Problèmes critiques et preuves formelles”, Université Paris 13).

8.3. Popularisation

Yann Régis-Gianas organised the "Fête de la Science" event for the computer science department of the University Paris 7. François Bobot, Guillaume Claret, Lourdes Gonzalez-Huesca and Pierre-Marie Pédrot also took part in the animation.

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

Yann Régis-Gianas co-organised the “Robocup”, a robotic and programming contest between undergraduate students of three universities of Paris (UPD, UP13, UP5).

Pierre Letouzey and Pierre Boutillier took part in the animation of the “Fête de la science” event at Inria Rocquencourt.

Yann Régis-Gianas gave one of the lectures to high-school teachers of the Académie de Paris in the context of the new curriculum “Informatique et Science du Numérique” (ISN).

9. Bibliography

Major publications by the team in recent years

[1] Z. ARIOLA, H. HERBELIN, A. SABRY. *A Type-Theoretic Foundation of Delimited Continuations*, in "Higher Order and Symbolic Computation", 2007, http://dx.doi.org/10.1007/s10990-007-9006-0.

[2] P.-L. CURIEN. *Substitution up to isomorphism*, in "Fundamenta Informaticae", 1993, vol. 19, p. 51-85.

[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, SIGPLAN Notices 35(9), ACM, September 18-21 2000, p. 233–243 [*DOI :* 10.1145/351240.351262], http://hal.archives-ouvertes.fr/inria-00156377/en/.

[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, G. C. NECULA, P. WADLER (editors), ACM, January 7-12 2008, p. 383-394.

[5] H. HERBELIN. *An intuitionistic logic that proves Markov’s principle*, in "Logic In Computer Science", Edinburgh, Royaume-Uni, IEEE Computer Society, 2010, http://hal.inria.fr/inria-00481815/en/.

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

[7] 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, Springer, July 2008, vol. 5133, p. 305–335, http://gallium.inria.fr/~fpottier/publis/regis-gianas-pottier-hoarefp.ps.gz.

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

[9] A. SAURIN. *On the Relations between the Syntactic Theories of $\lambda\mu$-Calculi*, in "17th Annual Conference of the EACSL 17th EACSL Annual Conference on Computer Science Logic - CSL 2008", Bertinoro Italie, Lecture notes in computer science, Springer, 2008, vol. 5213, p. 154-168 [*DOI :* 10.1007/978-3-540-87531-4_13], http://hal.archives-ouvertes.fr/hal-00527930/en/.

[10] M. SOZEAU, N. OURY. *First-Class Type Classes*, in "Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings", O. A. MOHAMED, C. MUÑOZ, S. TAHAR (editors), Lecture Notes in Computer Science, Springer, 2008, vol. 5170, p. 278-293.

Publications of the year

Articles in International Peer-Reviewed Journals

[11] A. BOVE, A. KRAUSS, M. SOZEAU. *Partiality and Recursion in Interactive Theorem Provers - An Overview*, in "Mathematical Structures in Computer Science", January 2012, To appear, http://hal.inria.fr/hal-00691459.

[12] Y. GUIRAUD, P. MALBOS. *Coherence in monoidal track categories*, in "Mathematical Structures in Computer Science", 2012, vol. 22, n° 6, p. 931-969, http://hal.inria.fr/hal-00470795.

[13] Y. GUIRAUD, P. MALBOS. *Higher-dimensional normalisation strategies for acyclicity*, in "Advances in Mathematics", 2012, vol. 231, n° 3-4, p. 2294-2351 [*DOI :* 10.1016/J.AIM.2012.05.010], http://hal.inria.fr/hal-00531242.

[14] D. ILIK. *Continuation-passing Style Models Complete for Intuitionistic Logic*, in "Annals of Pure and Applied Logic", May 2012 [*DOI :* 10.1016/J.APAL.2012.05.003], http://hal.inria.fr/hal-00647390.

[15] D. ILIK. *Delimited control operators prove Double-negation Shift*, in "Annals of Pure and Applied Logic", November 2012, vol. 163, n° 11, p. 1549-1559 [*DOI :* 10.1016/J.APAL.2011.12.008], http://hal.inria.fr/hal-00647389.

[16] A. SAURIN. *Böhm theorem and Böhm trees for the Lambda-mu-calculus*, in "Theoretical Computer Science", June 2012, vol. 435, p. 106-138 [*DOI :* 10.1016/J.TCS.2012.02.027], http://hal.inria.fr/hal-00695534.

[17] V. SILES, H. HERBELIN. *Pure Type System conversion is always typable*, in "Journal of Functional Programming", May 2012, vol. 22, n° 2, p. 153 - 180 [*DOI :* 10.1017/S0956796812000044], http://hal.inria.fr/inria-00497177.

Invited Conferences

[18] M. SOZEAU. *Coq avec Classes*, in "JFLA - Journées Françaises des Langages Applicatifs", Carnac, France, February 2012, http://hal.inria.fr/hal-00699595.

International Conferences with Proceedings

[19] R. AMADIO, Y. RÉGIS-GIANAS. *Certifying and reasoning on cost annotations of functional programs*, in "Foundational and Practical Aspects of Resource Analysis", Madrid, Spain, R. PEÑA (editor), Lecture Notes in Computer Science, Springer, June 2012, vol. 7177, p. 72-88, http://hal.inria.fr/inria-00629473.

[20] Z. ARIOLA, P. DOWNEN, H. HERBELIN, K. NAKATA, A. SAURIN. *Classical call-by-need sequent calculi : The unity of semantic artifacts*, in "FLOPS 2012 - 11th International Symposium on Functional and Logic Programming", Kobe, Japan, T. SCHRIJVERS, P. THIEMANN (editors), Lecture Notes in Computer Science, Springer, 2012, vol. 7294, p. 32-46 [*DOI :* 10.1007/978-3-642-29822-6], http://hal.inria.fr/hal-00697241.

[21] N. AYACHE, R. AMADIO, Y. RÉGIS-GIANAS. *Certifying and reasoning on cost annotations in C programs*, in "FMICS 2012 - 17th International Workshop on Formal Methods for Industrial Critical Systems", Paris, France, August 2012, http://hal.inria.fr/hal-00702665.

[22] H. HERBELIN. *A Constructive Proof of Dependent Choice, Compatible with Classical Logic*, in "LICS 2012 - 27th Annual ACM/IEEE Symposium on Logic in Computer Science", Dubrovnik, Croatia, IEEE Computer Society, 2012, p. 365-374, 1ère version rédigée en janvier 2011. Nombreuses corrections, et raffinements, appliqués après coup., http://hal.inria.fr/hal-00697240.

[23] G. JABER, N. TABAREAU, M. SOZEAU. *Extending Type Theory with Forcing*, in "LICS 2012 : Logic In Computer Science", Dubrovnik, Croatia, June 2012, http://hal.inria.fr/hal-00685150.

National Conferences with Proceeding

[24] P. BOUTILLIER. *A relaxation of Coq's guard condition*, in "JFLA - Journées Francophones des langages applicatifs - 2012", Carnac, France, February 2012, p. 1 - 14, http://hal.inria.fr/hal-00651780.

Conferences without Proceedings

[25] M. PUECH, Y. RÉGIS-GIANAS. *Safe Incremental Type Checking*, in "TLDI 2012 - Seventh ACM SIGPLAN Workshop on Types in Language Design and Implementation", Philadelphia, United States, January 2012, 2 pages, http://hal.inria.fr/hal-00650341.

Scientific Books (or Scientific Book chapters)

[26] P.-L. CURIEN. *Operads, clones, and distributive laws*, in "Operads and Universal Algebra : Proceedings of China-France Summer Conference", L. G. CHENGMING BAI, J.-L. LODAY (editors), Nankai Series in Pure, Applied Mathematics and Theoretical Physics, Vol. 9, World Scientific, Tianjin, China, 2012, p. 25-50, http://hal.inria.fr/hal-00697065.

Research Reports

[27] F. ASCHIERI. *Interactive Realizability for Classical Peano Arithmetic with Skolem Axioms*, Inria, April 2012, http://hal.inria.fr/hal-00685360.

[28] F. ASCHIERI. *Interactive Realizability for Second-Order Heyting Arithmetic with EM1 and SK1*, Inria, 2012, http://hal.inria.fr/hal-00657054.

[29] F. ASCHIERI, M. ZORZI. *Eliminating Skolem Functions in Peano Arithmetic with Interactive Realizability*, Inria, April 2012, http://hal.inria.fr/hal-00690270.

Other Publications

[30] P. CASTÉRAN, M. SOZEAU. *A gentle introduction to type classes and relations in Coq*, May 2012, This document presents the main features of type classes and user-defined relations in the Coq proof assistant. Available at http://www.labri.fr/perso/casteran/CoqArt/TypeClassesTut/typeclassestut.pdf, http://hal.inria.fr/hal-00702455.

[31] S. GAUSSENT, Y. GUIRAUD, P. MALBOS. *Coherent presentations and actions on categories*, 2012, 66 pages, http://hal.inria.fr/hal-00682233.

[32] M. SOZEAU, N. TABAREAU, G. JABER. *Forcing in Coq*, 2012, http://github.com/mattam82/Forcing.

References in notes

[33] D. J. ANICK. *On the Homology of Associative Algebras*, in "Trans. Amer. Math. Soc.", 1986, vol. 296, n^{o} 2, p. 641–659.

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

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

[36] B. BUCHBERGER. *An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal*, in "J. Symbolic Comput.", 2006, vol. 41, n^{o} 3-4, p. 475–511, Translated from the 1965 German original by Michael P. Abramson.

[37] A. CHLIPALA. *An Introduction to Programming and Proving with Dependent Types in Coq*, in "Journal of Formalized Reasoning", 2010, vol. 3, n^{o} 2, p. 1–93.

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

[39] COQ DEVELOPMENT TEAM, THE. *The Coq Reference Manual, version 8.2*, September 2008, http://coq.inria.fr/doc.

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

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

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

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

[44] P. DELIGNE. *Action du groupe des tresses sur une catégorie*, in "Invent. Math.", 1997, vol. 128, n^{o} 1, p. 159–175.

[45] 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.

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

[47] C. FÜHRMANN. *Direct Models for the Computational Lambda Calculus*, in "Electr. Notes Theor. Comput. Sci.", 1999, vol. 20, p. 245-292.

[48] F. GARILLOT. *Generic Proof Tools and Finite Group Theory*, École Polytechnique, December 2011.

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

[50] 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, North Holland, 1971, n^{o} 63, p. 63-92.

[51] 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.

[52] Y. GUIRAUD, P. MALBOS. *Higher-dimensional categories with finite derivation type*, in "Theory Appl. Categ.", 2009, vol. 22, n^{o} 18, p. 420-478.

[53] R. HARPER, R. POLLACK. *Type Checking with Universes*, in "Theor. Comput. Sci.", 1991, vol. 89, n^{o} 1, p. 107-136.

[54] H. HERBELIN. *An intuitionistic logic that proves Markov's principle*, in "Logic In Computer Science", United Kingdom Edinburgh, IEEE Computer Society, 2010, http://hal.inria.fr/inria-00481815/en.

[55] M. HOFMANN. *On the Interpretation of Type Theory in Locally Cartesian Closed Categories*, in "Computer Science Logic (CSL'94)", Springer Lecture Notes in Computer Science 933, 1994, p. 427-441.

[56] 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.

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

[58] J.-L. KRIVINE. *Structures de réalisabilité, RAM et ultrafiltre sur N*, in "CoRR", 2008, vol. abs/0809.2394.

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

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

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

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

[63] F. MÉTAYER. *Resolutions by polygraphs*, in "Theory Appl. Categ.", 2003, vol. 11, p. 148–184.

[64] M. PARIGOT. *Free Deduction: An Analysis of "Computations" in Classical Logic.*, in "Logic Programming, Second Russian Conference on Logic Programming", St. Petersburg, Russia, A. VORONKOV (editor), Lecture Notes in Computer Science, Springer, September 11-16 1991, vol. 592, p. 361-380, http://dblp.uni-trier.de.

[65] 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.

[66] B. C. PIERCE. *Proof Assistants as Teaching Assistants: A View from the Trenches*, in "ITP", M. KAUFMANN, L. C. PAULSON (editors), Lecture Notes in Computer Science, Springer, 2010, vol. 6172, 8.

[67] 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.

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

[69] B. SPITTERS, E. VAN DER WEEGEN. *Type classes for mathematics in type theory*, in "Mathematical Structures in Computer Science", 2011, vol. 21, n^{o} 4, p. 795-825.

[70] C. SQUIER, F. OTTO, Y. KOBAYASHI. *A finiteness condition for rewriting systems*, in "Theoret. Comput. Sci.", 1994, vol. 131, n^{o} 2, p. 271–294.

[71] J. TITS. *A local approach to buildings*, in "The geometric vein", New York, Springer, New York, 1981, p. 519–547.

[72] B. WERNER. *On the Strength of Proof-Irrelevant Type Theories*, in "Journal of Automated Reasoning", 2006, p. 604–618.

[73] N. DE BRUIJN. *AUTOMATH, a language for mathematics*, Technological University Eindhoven, November 1968, n^{o} 66-WSK-05.