IN PARTNERSHIP WITH:
CNRS

Université Denis Diderot (Paris 7)

Activity Report 2015

Project-Team PI.R2

Design, study and implementation of languages for proofs and programs

IN COLLABORATION WITH: Laboratoire Preuves, Programmes et Systèmes (PPS)

RESEARCH CENTER
Paris - Rocquencourt

THEME
Proofs and Verification

Table of contents

Project-Team PI.R2

Creation of the Team: 2009 January 01, updated into Project-Team: 2011 January 01

Keywords:

Computer Science and Digital Science:

2.1.1. - Semantics of programming languages
2.1.11. - Proof languages
2.1.3. - Functional programming
2.4.3. - Proofs
7.4. - Logic in Computer Science

Other Research Topics and Application Domains:

6.1. - Software industry
6.6. - Embedded systems

1. Members

Research Scientists

Pierre-Louis Curien [Team leader, CNRS, Senior Researcher, HdR]
Yves Guiraud [Inria, Researcher]
Hugo Herbelin [Inria, Senior Researcher, HdR]
Jean-Jacques Lévy [Inria, Senior Researcher (Emeritus), HdR]
Alexis Saurin [CNRS, Researcher]
Matthieu Sozeau [Inria, Researcher]

Faculty Members

Thierry Coquand [Univ. of Göteborg, Inria international chair]
Pierre Letouzey [Univ. Paris VII, Associate Professor]
Frédéric Loulergue [Univ. Orléans, Professor, until Aug 2015]
Yann Régis-Gianas [Univ. Paris VII, Associate Professor]

Engineers

Daniel de Rauglaudre [Inria]
Matej Kosik [Inria, from Oct 2015]

PhD Students

Cyrille Chenavier [Univ. Paris VII]
Maxime Lucas [Univ. Paris VII]
Guillaume Claret [Univ. Paris VII]
Thibaut Girka [granted by CIFRE (MITSUBISHI)]
Lourdes Del Carmen González Huesca [Univ. Paris VII, until Dec 2015, granted by ANR Paral ITP project]
Étienne Miquey [Univ. Paris VII]
Jovana Obradović [Univ. Paris VII]
Ludovic Patey [Univ. Paris VII]
Samanta Socci [Inria, until Feb 2015]
Pierre-Marie Pédrot [Univ. Paris VII, until Oct 2015]
Cyprien Mangin [Ecole Polytechnique]
Gabriel Lewertowski [Ecole Polytechnique]
Amina Doumane [Univ. Paris VII, funded by Ile de France]

Post-Doctoral Fellow

Marc Lasson [Inria, until Aug 2015]

Administrative Assistant

Lindsay Polienor [Inria]

Others

Akira Yoshimizu [PhD student at Tokyo Univ., Inria international internship, until Apr 2015]
Paul-André Melliès [External collaborator, CNRS, researcher]
Philippe Malbos [External collaborator, Univ. Lyon 1, associate professor]
Claudia Faggian [External collaborator, CNRS, researcher]

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. Research Program

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 [61] 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 [55], then by Howard and de Bruijn at the end of the 60's [66], [77], 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 and Huet's Calculus of Constructions [52], [53] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [75].

3.1.2. Towards the calculus of constructions

The λ-calculus, defined by Church [51], 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 [49].

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 [71].

In 1985, Coquand and Huet [52], [53] in the Formel team of Inria-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [62], [74]. 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 called OCaml and 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 [54] 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 moved to the new Inria Saclay research center. 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 the formalisation of mathematics in Coq, while contributing importantly to the development of Coq, in particular nowadays for what regards user interfaces.

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. Standard library

The standard library is 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 have their core 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 combining logic and programming 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 typed 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 exist – generally first-order ones.

3.4. Around and beyond the Curry-Howard correspondence

For two decades, the Curry-Howard correspondence has been limited to the intuitionistic case but since 1990, an important stimulus spurred on the community following Griffin's discovery that this correspondence was extensible to classical logic. The community then started to investigate unexplored potential connections between computer science and logic. One of these fields is the computational understanding of Gentzen's sequent calculus while another one is 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 [63] that some operators known as control operators were typable by the principle of double negation elimination ($\neg\neg A \Rightarrow A$), a principle that enables classical reasoning.

Control operators are used to jump from one location of a program to another. They were first considered in the 60's by Landin [70] and Reynolds [73] and started to be studied in an abstract way in the 80's by Felleisen *et al* [59], leading to Parigot's $\lambda\mu$-calculus [72], a reference calculus that is in close Curry-Howard correspondence with classical natural deduction. In this respect, control operators are fundamental pieces to establish a 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, for which two kinds of interpretations 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 rules. This line of work led in 2000 to 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 [69] and Krivine's abstract machine for call-by-name evaluation [68], [67]. 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 [60]: 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. Highlights of the Year

4.1. Highlights of the Year

4.1.1. Coq 8.5

Version 8.5 of Coq will remain as one of the most important versions of the history of Coq. It includes five big achievements affecting various components of the system: a new proof engine supporting multi-goal and deep backtracking by Arnaud Spiwack; a new asynchronous evaluation engine supporting efficient parallel development of interactive documents, parallel evaluation of tactics, modular compilation of files by Enrico Tassi; full universe polymorphism by Matthieu Sozeau; a new notion of primitive projections highlighting the negatively polarised view at record types by Matthieu Sozeau; a new evaluation machine by Maxime Dénès which works by compiling to OCaml.

The year 2015 was also a year of thinking on new ways to popularise Coq and further enhance the interaction between users and developers. In particular, a first Coq Coding Sprint gathered about 30 participants around about 10 developers.

4.1.2. EPIT 2015

This year, the French Spring School in Theoretical Computer Science (EPIT) was organised by Yann Régis-Gianas, Pierre Letouzey, Matthieu Sozeau and Pierre-Marie Pédrot in Fréjus (France). This CNRS "école thématique" was dedicated to the mechanisation of proofs of programs and of mathematical theorems in Coq. It was attended by 50 participants, coming from different research communities. Besides the courses introducing the basics of Coq and proof development in Coq, substantial efforts of formalisation in various areas such as formal language theory, number theory, or combinatorics were presented by their authors, and the attendants were encouraged to discuss their own formalisation projects with the Coq developers. The school has been sponsorised by the CNRS, the FIFP, the ADT Coq and the ANR Paral-ITP. The feedback from the participants was very positive.

5. New Software and Platforms

5.1. Coq

KEYWORDS: Proof - Certification - Formalisation
FUNCTIONAL DESCRIPTION

Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

- Participants: Benjamin Grégoire, Enrico Tassi, Bruno Barras, Yves Bertot, Pierre Boutillier, Xavier Clerc, Pierre Courtieu, Maxime Dénès, Stéphane Glondu, Vincent Gross, Hugo Herbelin, Pierre Letouzey, Assia Mahboubi, Julien Narboux, Jean-Marc Notin, Christine Paulin-Mohring, Pierre-Marie Pédrot, Loïc Pottier, Matthias Puech, Yann Régis-Gianas, François Ripault, Matthieu Sozeau, Arnaud Spiwack, Pierre-Yves Strub, Benjamin Werner, Guillaume Melquiond and Jean-Christophe Filliâtre
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Hugo Herbelin
- URL: http://coq.inria.fr/

5.1.1. Version 8.5

Cf. Highlights section. Version 8.5 includes as well a number of miscellaneous changes, at the level of tactics, of the specification language, of the Coq tools, of the standard library, altogether amounting to about 150 items in the change log of the version. In particular, Pierre-Marie Pédrot has been working on the overall optimisation of Coq, by tracking hotspots in the code. Coq v8.5 is currently much more efficient than its v8.4 counterpart, and is about as quick as v8.3, while having been expanded with a lot of additional features.

As a counterpart, the complexity of this new version induced a long phase of experimentation which included 3 different beta versions spanned over the whole 2015 year, with the final version being eventually released for the CoqPL workshop in January 2016.

5.1.2. Universes

Matthieu Sozeau followed up his work on universe polymorphism and uncovered important theoretical and practical problems regarding conversion and unification of universe polymorphic definitions in the presence of cumulativity and the Prop $\leq$ Type rule, as well as the invariants of the consistency checker. He also collaborated with Maxime Dénès and Benjamin Grégoire (Gallium and Marelle) on adapting the efficient conversion tests to universe polymorphism and with Enrico Tassi (Marelle) on the integration with the asynchronous proof development infrastructure. The universe polymorphic system is part of the 8.5 release.

5.1.3. The Equations plugin

Matthieu Sozeau continued work on the Equations plugin and fixed the remaining bugs preventing full automation of a middle-size example of formalisation – the normalisation proof of a predicate version of System F – together with Cyprien Mangin, during his master's internship. This involved finding a new termination proof for the calculus and making the dependent pattern-matching compilation more robust and axiom-free, using a different encoding of pattern-matching problems. This work was presented at LFMTP'15 in Berlin [29]. Since then, the system has been adapted to work with universe polymorphism and the new features of Coq 8.5.

5.1.4. Proof development in Coq

Pierre Letouzey developed a few new results about some Hofstadter sequences (see https://oeis.org/A005206 and https://oeis.org/A123070). These results have been proved in Coq, and they are presented in the technical report [39].

5.1.5. Proofs of algorithms on graphs

Chen Ran (ISCAS/SKLCS, Beijing) and Jean-Jacques Lévy pursued their work about producing readable formal proofs of graph algorithms. This work is performed in Why3 and partly in Coq. Graph algorithms are a good testbed for experimenting correctness proofs of programs with shared structures. We considered basic algorithms such as depth-first-search, random walk, acyclicity test, articulation points, strongly connected components, minimum spanning trees. In each case, the goal is to provide a simple proof as abstract as possible, although checked by computer. A longer term objective is to give formal proofs which could be inserted in algorithms textbooks. A progress work paper is under submission [41].

5.1.6. Development of programs for parallel and cloud computing

Frédéric Loulergue continued his work on the SyDPaCC framework. The goal of this framework is to ease the systematic development of correct parallel programs, in particular large-scale data-intensive applications. The parallel versions of the programs are written with a Coq axiomatisation of Bulk Synchronous Parallel ML (BSML) primitives. New results about SyDPaCC include the design and implementation of a new version of the core of the framework [21]. This new version has been used in a course of École des Jeunes Chercheur/se/s en Informatique Mathématique (EJCIM 2015) [38].

As the SyDPaCC framework currently mixes certified code extracted from Coq and unverified code, Frédéric Loulergue and Pierre Letouzey have worked on an extended extraction that generates, when possible, OCaml conditions for preconditions on function arguments. This part is still on-going work.

Frédéric Loulergue collaborated with Frédéric Dabrowski and Thomas Pinsard (Univ. Orléans) on the semantics and compilation of languages with nested atomic sections and thread escape. In [18], the focus is on the semantics of programming languages providing these features. The main contribution is the precise definition of atomicity, well-synchronisation and the proof that the latter implies the strong form of the former. A formalisation of the results in the Coq proof assistant is described.

In [27], the compilation of a language with nested atomic sections and thread escape towards a language with threads and locks is addressed. The design decisions of this compilation pass and of the target language were made with respect to the ultimate goal of a mechanised proof of semantic preservation.

Frédéric Loulergue collaborated with Allan Blanchard, Nikolai Kosmatov and Matthieu Lemerre (CEA LIST) on the verification of a critical component of a hypervisor. In [23], they present a case study on formal verification of the virtual memory system of the cloud hypervisor Anaxagoros, a microkernel designed for resource isolation and protection. The code under verification is specified and proven in the software verification framework, mostly using automatic theorem proving. The remaining properties are interactively proven with the Coq proof assistant.

Frédéric Loulergue collaborated with Asma Guesmi, Pascal Berthomé and Patrice Clemente (INSA Centre Val de Loire) on resources placement in the Cloud taking into account security requirements [28].

5.2. Other software developments

In collaboration with François Pottier (Inria Gallium), Yann Régis-Gianas maintained Menhir, an LR parser generator for OCaml. Yann Régis-Gianas develops the “Hacking Dojo”, a web platform to automatically grade programming exercises. The platform is now used in several courses of the University Paris Diderot. He gets help from the internship of Alexandre Ly, a master student of the Paris Diderot University. In collaboration with Beta Ziliani (LIIS, Cordoba, Argentine), Yann Régis-Gianas, Béatrice Carré and Jacques-Pascal Deplaix develop MetaCoq, an extension of Coq to use Coq as a metalanguage for itself.

6. New Results

6.1. Effects in proof theory and programming

Participants: Guillaume Claret, Pierre-Louis Curien, Hugo Herbelin, Étienne Miquey, Ludovic Patey, Pierre-Marie Pédrot, Yann Régis-Gianas, Alexis Saurin.

6.1.1. Axiom of dependent choice in classical arithmetic

In 2012, 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. Étienne Miquey has been working on a sequent-calculus version of this system, using Danvy's methodology of semantic artifacts, to progressively reduce the consistency of such a system to the normalisation of Girard-Reynold's system F. To achieve this goal, he incidentally proposed a way to get a dependently-typed sequent calculus, as well as a method to type a state-and-continuation-passing style translation of call-by-need calculus.

6.1.2. The computational contents of completeness proofs

Hugo Herbelin worked on the computational content of Gödel's completeness theorem, developing a proof with side-effects suitable for normalisation-by-evaluation.

6.1.3. Gödel's functional interpretation

Pierre-Marie Pédrot extended the proof-as-program interpretation of Gödel's Dialectica translation to the fully dependent setting, including dependent elimination [17].

6.1.4. Logical foundations of call-by-need evaluation

Alexis Saurin and Pierre-Marie Pédrot extended their reconstruction of call-by-need based on linear head reduction with control. They showed how linear head reduction could be adapted to the $\lambda\mu$-calculus. This classical linear head reduction lifts the usual properties of the intuitionistic one (with respect to σ-equivalence) to the $\lambda\mu$-calculus (and its σ-equivalence already formulated by Olivier Laurent in his PhD thesis). Moreover, they showed that substitution sequences of the $\lambda\mu$-calculus linear head reduction are in correspondence with the classical Krivine abstract machine substitution sequences, validating the known fact that the KAM implements linear head reduction. In a second step, they could lift to the $\lambda\mu$-calculus their three-step transformation from linear head reduction to call-by-need, and study the correspondence with Ariola, Herbelin and Saurin's classical call-by-need. This work appeared as one of the chapters of Pierre-Marie Pédrot's thesis and has been accepted for publication at ESOP'16 [30].

6.1.5. Call-by-name forcing

Pierre-Marie Pédrot studied variants of the forcing construction by decomposing it through call-by-push-value. In particular, the by-name decomposition behaves much more nicely w.r.t. the computational content of proofs and is a candidate for a dependently-typed extension. This work is partially reported on in his PhD [17].

6.1.6. A theory of effects and resources

In joint work with Marcelo Fiore and Guillaume Munch-Maccagnoni, Pierre-Louis Curien considered the Curry-Howard-Lambek correspondence for effectful computation and resource management, specifically proposing polarised calculi together with presheaf-enriched adjunction models as the starting point for a comprehensive semantic theory relating logical systems, typed calculi, and categorical models in this context. Our thesis is that the combination of effects and resources should be considered orthogonally. Model theoretically, this leads to an understanding of our categorical models from two complementary perspectives: (i) as a linearisation of CBPV (Call-by-Push-Value) adjunction models, and (ii) as an extension of linear/non-linear adjunction models with an adjoint resolution of computational effects. When the linear structure is cartesian and the resource structure is trivial, we recover Levy's notion of CBPV adjunction model, while when the effect structure is trivial, we have Benton's linear/non-linear adjunction models. Further instances of our model theory include the dialogue categories with a resource modality of Melliès and Tabareau, and the Enriched Effect Calculus models of Egger, Møgelberg and Simpson. Our development substantiates the

approach by providing a lifting theorem of linear models into cartesian ones. To each of our categorical models we systematically associate a typed term calculus, each of which corresponds to a variant of the sequent calculi LJ (Intuitionistic Logic) or ILL (Intuitionistic Linear Logic). The adjoint resolution of effects corresponds to polarisation whereby, syntactically, types locally determine a strict or lazy evaluation order and, semantically, the associativity of cuts is relaxed. In particular, our results show that polarisation provides a computational interpretation of CBPV in direct style. Further, we characterise depolarised models: those where the cut is associative, and where the evaluation order is unimportant. This work will be presented at POPL 2016 [26].

6.1.7. Coq as a programming language with effects

As part of his PhD thesis, Guillaume Claret defined a notion of effectful interactive computation as an embedded DSL in Coq (in the spirit of the works on algebraic effects), and used it to implement a web server. It is equipped with a dual notion of effectful interactive execution context. Using these two notions together, Guillaume Claret is able to specify and reason about interactive programs inside Coq. He submitted several papers about this line of work: one has been published [32], others will be part of his PhD manuscript.

6.2. Reasoning and programming with infinite data

Participants: Amina Doumane, Alexis Saurin, Pierre-Marie Pédrot, Yann Régis-Gianas.

This theme is part of the ANR project Rapido (see the National Initiatives section).

6.2.1. Interactive semantics for logic fixed-points and infinitary logics.

Amina Doumane and Alexis Saurin, in a joint work with David Baelde published at CSL 2015 [24], developed a game-semantics of $\mu MALL$ (Multiplicative Additive Linear Logic with least and greatest fixpoints).

This interactive semantics was worked out in computational ludics, benefitting from both the work by Clairambault on a HO style game semantics for an intuitionistic logic with least and greatest fixpoints and from the flexibility of Terui's computational ludics (in particular its ability to consider designs with cuts).

This framework is built around the notion of design, which can be seen as an analogue of the strategies of game semantics. The infinitary nature of designs makes them particularly well suited for representing computations over infinite data. We provided $\mu MALL$ with a denotational semantics (that is invariant by cut-elimination), interpreting proofs by designs and formulas by particular sets of designs called behaviours. Then a completeness result for a specific class of designs is proved, the class of "essentially finite designs", which are those designs performing a finite computation followed by a copycat. On the way to the previous completeness result, we investigate semantic inclusion, proving its decidability (given two formulas A and B, one can decide whether the semantics of A is included in the semantics of B) and completeness (if semantic inclusion holds, the corresponding implication is provable in $\mu MALL$).

6.2.2. Proof theory of circular proofs

In a collaboration with David Baelde, Amina Doumane and Alexis Saurin developed further the theory of infinite proofs. Studying the proof theory of circular proofs on MALL, they established a result of focalisation for these infinite proofs. The usual result of focalisation for linear logic can actually be extended to circular proofs but, contrarily to $\mu MALL$ where fixed-points operators can be given an arbitrary polarity, the least fixed-point must be set to be a positive construction and the greatest fixed-points to be negative, which is consistent with intuition from programming with inductive and co-inductive datatypes. An interesting phenomenon arising with focalisation is that some infinite but regular proofs may not have any regular focused proofs. This is similar to what happens for cut-elimination of regular proofs.

Works on cut-elimination for circular proofs are still ongoing.

6.2.2.1. Automata theory meets proof theory: proof certificates for Büchi inclusion

In a joint work with David Baelde and Lucca Hirschi, Amina Doumane and Alexis Saurin carried out a proof-theoretical investigation of the linear-time μ-calculus, proposing well-structured proof systems and showing constructively that they are complete for inclusions of Büchi automata suitably encoded as formulas.

They do so in a way that combines the advantages of two lines of previous work: Kaivola gave a proof of completeness for an axiomatisation that amounts to a finitary proof system, but his proof is non-constructive and yields no reasonable procedure. On the other hand, Dax, Hofmann and Lange recently gave a deductive system that is appropriate for algorithmic proof search, but their proofs require a global validity condition and do not have a well understood proof theory.

They work with well-structured proof systems, effectively constructing proofs in a finitary sequent calculus that enjoys local correctness and cut elimination.

This involves an intermediate circular proof system in which one can obtain proofs for all inclusions of parity automata, by adapting Safra's construction. In order to finally obtain finite proofs of Büchi inclusions, a translation result from circular to finite proofs is designed.

6.3. Effective higher dimensional algebra

Participants: Cyrille Chenavier, Pierre-Louis Curien, Yves Guiraud, Maxime Lucas, Philippe Malbos, Jovana Obradović.

6.3.1. Rewriting methods for Artin monoids

With Stéphane Gaussent (ICJ, Univ. Saint-Étienne), Yves Guiraud and Philippe Malbos have used higher-dimensional rewriting methods for the study of Artin monoids, a class of monoids that is fundamental in algebra and geometry. This work formulates 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 monoid [76], and one by Deligne about the actions of Artin monoids on categories [58], both originally proved by geometrical and non-constructive methods. An improved completion procedure, called the homotopical completion-reduction procedure (see also [8]), is formalised and used to give constructive proofs of (improved versions of) both theorems. This work has been published in Compositio Mathematica [19] and has been implemented in a Python library (http://www.pps.univ-paris-diderot.fr/~guiraud/cox/cox.zip).

6.3.2. Rewriting and Garside theory

Yves Guiraud has collaborated with Patrick Dehornoy (LNO, Univ. Caen) to develop an axiomatic setting for monoids with a special notion of quadratic normalisation map with good computational properties. This theory generalises the normalisation procedure known for monoids that admit a special family of generators called a Garside family [57] to a much wider class that also includes the plactic monoids. It is proved that good quadratic normalisation maps correspond to quadratic convergent presentations, together with a sufficient condition for this to happen, based on the shape of the normalisation paths on length-three words. This work has been submitted for publication to the Journal de l'École Polytechnique — Mathématiques [44].

Building on this last article, Yves Guiraud currently collaborates with Matthieu Picantin (Automates team, LIAFA, Univ. Paris 7) to generalise the main results of [19] to monoids with a Garside family. This will allow an extension of the field of application of the rewriting methods to other geometrically interesting classes of monoids, such as the dual braid monoids.

6.3.3. Higher-dimensional linear rewriting

With Eric Hoffbeck (LAGA, Univ. Paris 13), Yves Guiraud and Philippe Malbos have introduced in [64] the setting of linear polygraphs to formalise a theory of linear rewriting, generalising Gröbner bases. They have adapted the computational method of [7] to compute polygraphic resolutions of associative algebras, with applications to the decision of the Koszul homological property. They are currently engaged into a major overhaul of this work, whose main goal is to ease the adaptation of the results to other algebraic varieties, like commutative algebras or Lie algebras.

6.3.4. Theory of reduction operators

Cyrille Chenavier, supervised by Yves Guiraud and Philippe Malbos, explores the use of Berger's theory of reduction operators [50] to design new rewriting methods in algebra. In [42], he proposed a construction of a contracting homotopy for the Koszul complex of an algebra (a complex characterising the homological property of Koszulness): when an algebra admits a side-confluent presentation (a strong hypothesis of confluence), he gave a candidate for the contracting homotopy, built using specific representations of confluence algebras; when the presentation satisfies an additional condition, called the extra-condition, it turns out that this candidate works.

6.3.5. Rewriting methods for coherence

In [45], Maxime Lucas, supervised by Yves Guiraud and Pierre-Louis Curien, has applied the rewriting techniques of [65] to prove coherence theorems for bicategories and pseudofunctors. He obtained a coherence theorem for pseudonatural transformations thanks to a new theoretical result, improving on the former techniques, that relates the properties of rewriting in 1- and 2-categories.

6.3.6. Wiring structure of operads and operad-like structures

Building on recent ideas of Marcelo Fiore on the one hand, and of François Lamarche on the other hand, Pierre-Louis Curien and Jovana Obradović developed a syntactic approach, using some of the kit of Curien-Herbelin's duality of computation and its polarised versions by Munch and Curien, to the definition of various structures that have appeared in algebra under the names of operads, cyclic operads, dioperads, properads, modular and wheeled operads, permutads, etc. These structures are defined in the literature in different flavours. The goal is to formalise the proofs of equivalence between these different styles of definition. This work is completed for cyclic operads and was presented at the conference Category Theory 2015 in Aveiro [43]. Further work will be to make these proofs modular, so as not to repeat them for each variation of the notion of operad.

6.3.7. A graphical proof of the Bénabou-Roubaud theorem

As a substantial development of reasoning with string diagrams, Jovana Obradović gave a complete proof of the Bénabou-Roubaud monadic descent theorem in [47]. One of the essential points concerning Grothendieck's original approach to descent theory consists of identifying the class of effective descent morphisms for a given fibration. In the special case of a bifibration satisfying Beck-Chevalley condition, Bénabou and Roubaud have given such a characterisation by means of monadicity. Due to the technically complicated calculations involving Grothendieck's cocycle condition, the categorical equivalence which reflects the comparison of the descent in fibered categories with monadic descent is usually not worked out in complete detail in the literature. Jovana Obradović linked the monadic and the original viewpoint via another possible definition of the category of descent data. This intermediate step, due to Janelidze and Tholen, involves constructions in internal categories and it provides an example on how one can stay in the world of string diagrams even when dealing with morphisms which do not have an explicit string diagram definition.

6.4. Incrementality

Participants: Yann Régis-Gianas, Lourdes Del Carmen González Huesca, Thibaut Girka.

An optimisation to perform incremental computations was developed by Lourdes del Carmen González Huesca and Yann Régis-Gianas, providing a mechanism to achieve efficiency. Incrementality as a way to propagate an input change into a corresponding output change is guided by formal change descriptions over terms and dynamic differentiation of functions. The data-changes are represented by displaceable types, a general framework to displace terms directed by types. An extension of the simply-typed lambda-calculus with differentials and partial derivatives offers a language to reason about incrementality. The basic system, λ-diff, was enriched with expressions for fixed-points and data-types together with their corresponding derivatives to analyse incrementality over them. The above results are reported in the second part of Lourdes González Huesca PhD thesis [16].

In collaboration with Paolo Giarrusso and Yufei Cai (Univ Marburg, Allemagne), Yann Régis-Gianas developed a new method to incrementalise higher-order programs using formal derivatives and static caching. A paper is in preparation.

In collaboration with David Mentré (Mitsubishi), Thibaut Girka and Yann Régis-Gianas designed and certified a new algorithm for correlating program generation: such a program is used to characterise the differences between two close programs. (Therefore, a correlating program is a good input for an incremental static analyser.) Before their work, only one algorithm existed in the literature and it was unsound. The new algorithm is sound and certified in Coq. This work has been published in the ATVA conference. Thibaut Girka has presented this work [33] at ATVA 2015.

In collaboration with David Mentré (Mitsubishi), Thibaut Girka and Yann Régis-Gianas are developing a theoretical framework to define a notion of differential operational semantics: a general mathematical object to characterise the difference of behavior of two close programs.

6.5. Metatheory and development of Coq

Participants: Pierre-Louis Curien, Hugo Herbelin, Pierre Letouzey, Yann Régis-Gianas, Matthieu Sozeau.

6.5.1. Models of type theory

Simplicial sets and their extensions as Kan complexes can serve as models of homotopy type theory. Hugo Herbelin extended his concrete type-theoretic formalisation of semi-simplicial sets [20] to simplicial sets.

6.5.2. Unification

Matthieu Sozeau is working in collaboration with Beta Ziliani (PhD at MPI-Saarbrücken, now assistant professor at Cordoba, Argentina) on formalising the unification algorithm used in Coq, which is central for working with advanced type inference features like Canonical Structures. This is the first precise formalisation of all the rules of unification including the ones used for canonical structure resolution and universes. The presentation includes a careful study of the heuristics used in the existing Coq algorithms, which can be added or removed from the new implementation modularly. This work has been presented at the ICFP'15 conference [31].

6.5.3. Nominal techniques

Matthieu Sozeau cosupervised the internship of Gabriel Lewertowski with Nicolas Tabareau (Ascola team, Nantes), on the development of a library for nominal reasoning in Coq/Ssreflect. The goal of this internship was to study the use of nominal sets to ease the formalisation of programming language (meta-)theory. A library based on the Mathematical Components formalisation of finite sets and effective quotients was built, providing generic definitions of substitution and elimination operators for simple descriptions of programming language syntax as a grammar. This work was done in collaboration with Assia Mahboubi (Specfun) and Cyril Cohen (Marelle). It forms the basis for the formalisation of cubical type theory, a new type theory using name abstraction that implements an axiom-free version of Homotopy Type Theory.

7. Partnerships and Cooperations

7.1. National Initiatives

Alexis Saurin (coordinator) and Yann Régis-Gianas are members of the four-year RAPIDO ANR project accepted in 2014 and starting in January 2015. RAPIDO aims at investigating the use of proof-theoretical methods to reason and program on infinite data objects. The goal of the project is to develop logical systems capturing infinite proofs (proof systems with least and greatest fixed points as well as infinitary proof systems), to design and to study programming languages for manipulating infinite data such as streams both from a syntactical and semantical point of view. Moreover, the ambition of the project is to apply the fundamental results obtained from the proof-theoretical investigations (i) to the development of software tools dedicated to

the reasoning about programs computing on infinite data, *e.g.* stream programs (more generally coinductive programs), and (ii) to the study of properties of automata on infinite words and trees from a proof-theoretical perspective with an eye towards model-checking problems. Other permanent members of the project are Christine Tasson from PPS, David Baelde from LSV, ENS-Cachan, and Pierre Clairambault, Damien Pous and Colin Riba from LIP, ENS-Lyon.

Pierre-Louis Curien (coordinator), Yves Guiraud and Philippe Malbos are members of the three-year Focal project of the IDEX Sorbonne Paris Cité, started in June 2013. This project, giving the support for the PhD grant of Cyrille Chenavier, concerns the interactions between higher-dimensional rewriting and combinatorial algebra. This project is joint with members of the LAGA (Laboratory of Mathematics, Univ. Paris 13).

Pierre-Louis Curien (coordinator), Yves Guiraud and Philippe Malbos are members of the four-year Cathre ANR project, started in January 2014. This project, giving the support for the PhD grant of Maxime Lucas, investigates the general theory of higher-dimensional rewriting, the development of a general-purpose library for higher-dimensional rewriting, and applications in the fields of combinatorial algebra, combinatorial group theory and theoretical computer science. This project is joint with members of the LAGA (Univ. Paris 13), the LIX (École Polytechnique), the ICJ (Univ. Lyon 1 and Univ. Saint-Étienne), the I2M (Univ. Aix-Marseille) and the IMT (Univ. Toulouse 3).

Pierre-Louis Curien, Yves Guiraud (local coordinator) and Matthieu Sozeau are members of the Groupement de Recherche Topologie Algébrique, federating French researchers working on classical topics of algebraic topology and homological algebra, such as homotopy theory, group homology, K-theory, deformation theory, and on more recent interactions of topology with other themes, such as higher categories, motivic homotopy, string theory.

Matthieu Sozeau, Hugo Herbelin, Lourdes del Carmen González Huesca and Yann Régis-Gianas were members of the ANR Paral-ITP, which started in November 2011 and ended in June 2015, and aimed at 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 and will end mid 2016. Récré is about realisability and rewriting, with applications to proving with side-effects and concurrency.

Yann Régis-Gianas collaborates with Mitsubishi Rennes on the topic of differential semantics. This collaboration led to the CIFRE grant for the PhD of Thibaut Girka.

Yann Régis-Gianas is a member of the ANR COLIS dedicated to the verification of Linux Distribution installation scripts. This project is joint with members of VALS (Univ Paris Sud) and LIFL (Univ Lille).

Matthieu Sozeau is a member of the CoqHoTT project led by Nicolas Tabareau (Ascola team, École des Mines de Nantes), funded by an ERC Starting Grant. The PhD grant of Gabriel Lewertowski is funded by the CoqHoTT ERC.

7.2. European Initiatives

7.2.1. Collaborations in European Programs, except FP7 & H2020

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 not involved in an Inria International Labs

Pierre-Louis Curien and Claudia Faggian (external collaborator) participate to the Associated Team CRECOGI (Concurrent, Resourceful and Effectful Computation, by Geometry of Interaction) between the project-team Focus (Bologna) and the University of Tokyo (principal investigators Ugo dal Lago and Ichiro Hasuo) (started in 2015).

7.3.2. Inria International Partners

7.3.2.1. Informal International Partners

The project-team has collaborations with University of Aarhus (Denmark), University of Oregon, University of Tokyo, University of Sovi Sad and the Institute of Mathematics of the Serbian Academy of Sciences, University of Nottingham, Institute of Advanced Study, MIT, the University of Cambridge, and Universidad Nacional de Córdoba.

7.3.3. Participation In other International Programs

Pierre-Louis Curien participates to the ANR International French-Chinese project LOCALI (Logical Approach to Novel Computational Paradigms), coordinated by Gilles Dowek.

7.4. International Research Visitors

7.4.1. Visits of International Scientists

Andrej Bauer (University of Novi Sad) visited πr^2 and PPS for one month in September 2015 to collaborate with Matthieu Sozeau.

7.4.2. Internships

Akira Yoshimizu had a six-month Inria international internship (Nov 2014 - April 2015). He worked on abstract machines for the geometry of synchronisation, a variation of Girard's geometry of interaction that incorporates synchronisation and that is fit for dealing with quantum primitives added to a functional language, and coauthored a paper at LICS 2015 with Ugo Dal Lago, Claudia Faggian, and Benoît Valiron [56].

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific events organisation

8.1.1.1. Member of the organising committees

Yves Guiraud and Philippe Malbos organised with Samuel Mimram (LIX, École Polytechnique) the first edition of the Higher-Dimensional Rewriting and Applications (HDRA) workshop of the International Conference on Rewriting, Deduction, and Programming (RDP), held in Warsaw in June-July 2015. The second edition of HDRA will be held in June 2016 in Porto, during the new Formal Structures for Computation and Deduction (FSCD) conference.

Yann Régis-Gianas, Pierre Letouzey, Matthieu Sozeau were the organisers of the "Ecole de Printemps d'Informatique Théorique 2015" about proof of programs (cf. Highlights section).

Matthieu Sozeau co-organised and chaired the first Coq for Programming Languages (CoqPL) workshop, collocated with POPL'15 in Mumbai, India, in January 2015.

8.1.2. Scientific events selection

8.1.2.1. Member of the conference program committees

Frédéric Loulergue is member of of the program committee of the International Conference on Computational Science (ICCS'15), and is co-chair of the special session "Formal Approaches to Parallel and Distributed Systems" (4PAD) of the Euromicro PDP 2016 conference.

8.1.2.2. Member of the conference steering committees

Pierre-Louis Curien and Hugo Herbelin were members of the steering committee of the conference *Typed Lambda Calculi and Applications* (TLCA) until its merge with the conference *Rewriting Techniques and Application* in the new conference *Formal Structures for Computation and Deduction* (FSCD), whose first edition will be held in Porto in 2016.

Pierre-Louis Curien is member of the steering committee of the international workshop Games for Logic and Programming Languages (GaLop).

Matthieu Sozeau is member of the steering committee of the Dependently Typed Programming international workshop (DTP).

Frédéric Loulergue is a member of the steering committee of the international symposium on High-Level Parallel Programming and Applications (HLPP).

8.1.3. Journal

8.1.3.1. Member of the editorial boards

Pierre-Louis Curien is co-editor in chief (and becomes editor in chief in January 2016) of Mathematical Structures in Computer Science.

Hugo Herbelin, Pierre Letouzey and Matthieu Sozeau were co-editors of the post-proceedings of the conference TYPES 2014. These post-proceedings were published in October 2015 as LIPICS volume 39.

8.1.3.2. Reviewer - Reviewing activities

The members of the team reviewed numerous papers for numerous journals and international conferences.

8.1.4. Invited talks

Pierre-Louis Curien gave an invited talk at the GALOP'15 (Games in Logic and Programming) Workshop, in London (April 11-12).

Matthieu Sozeau participated to the HoTT/UF workshop in Warsaw, Poland (June 29-30). He gave an invited lecture on "Coq support for Homotopy Type Theory".

Alexis Saurin gave an invited talk, in the form of a distilled tutorial, at WoC'2015, affiliated to ETAPS 2015 on "Logical by need".

8.1.5. Hiring committees

Pierre-Louis Curien has been member of the "Comité de Sélection" for a professor position in mathematics at the University Paris Diderot.

8.1.6. Scientific expertise

Pierre-Louis Curien is a member of the Scientific Committee of the CIRM (since June 2013).

8.1.7. Presentation of papers

Amina Doumane presented [24] at CSL (sep 2015).

Matthieu Sozeau has presented [31] at ICFP 2015.

Cyprien Mangin has presented [29] at LFMTP 2015.

Yann Régis-Gianas has presented [32] at Formalize 2015.

Thibaut Girka has presented [33] at ATVA 2015.

Jovana Obradović has presented [43] at Category Theory 2015.

8.1.8. Other presentations

Matthieu Sozeau gave a talk on the development of Coq at the CoqPL workshop in Mumbai (Jan 2015).

Amina Doumane gave a talk at GALOP'15 on Least and Greastest Fixed Points in Ludics.

Étienne Miquey gave a talk at GALOP'15 Workshop on realisability games for arithmetical formulæ [37], and a talk on a classical sequent calculus with dependent types at TYPES'15 [34].

Hugo Herbelin gave a talk at TYPES'15 on a proof of Gödel's completeness using side-effects.

Hugo Herbelin gave a talk at the workshop HoTT-UF15 on an extension of his construction of semi-simplicial types to simplicial types.

Pierre-Louis Curien and Jovana Obradović gave talks at the workshop Logic and Applications 2015 http://imft.ftn.uns.ac.rs/math/cms/LAP2015, Dubrovnik (September 21-25).

Cyrille Chenavier and Jovana Obradović gave talks at the workshop HDRA, Warsaw (cf. Scientific events organisation).

8.1.9. Talks in seminars

Yves Guiraud gave talks on various aspects of Squier's theory and its applications in algebra at the LCR seminar (Jan. 2015, LIPN, Univ. Paris 13), the Logique et Interactions seminar (Nov. 2015, I2M, Univ. Aix-Marseille) and the Chocola monthly meeting (Dec. 2015, ENS Lyon).

Étienne Miquey gave talks on the classical realisability and arithmetical formulæ at the Chocola monthly meeting (May 2015, ENS de Lyon) and the Logique et Intéractions seminar (Jul. 2015, I2M, Univ. Aix-Marseille).

Matthieu Sozeau gave a talk on a new unification algorithm for Coq at the MIT seminar in Boston (April 2015).

Jean-Jacques Lévy gave a talk about Readable proofs of Depth First Search in graphs using Why3, at VALS seminar, Plateau de Saclay (December), and a talk about The cost of usage in the lambda-calculus, at Deducteam seminar, Inria (April), and Journées Chocola at ENS-Lyon (November).

8.1.10. Attendance to conferences, workshops, schools,...

Matthieu Sozeau attended POPL'15, RDP'15, LFMTP'15 and ICFP'15.

Étienne Miquey attended GALOP'15, the "École Jeunes Chercheurs 2015" of the GDR-IM, TYPES'15 and the EPIT spring school.

Hugo Herbelin attended TYPES'15, RDP'15 and ITP'15.

Pierre-Marie Pédrot attended JFLA'15 and LICS'15.

Hugo Herbelin, Pierre Letouzey, Pierre-Marie Pédrot and Matthieu Sozeau attended the first Coq coding sprint as developers communicating their expertise to the participants. Cyprien Mangin attended as participant.

8.1.11. 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. Matthieu Sozeau joined the organisation in 2014.

The speakers in 2015 were Danko Ilik (Eliminating control operators from classical realisability), Ali Assaf (Tarski and Coq), Maxime Dénès (Coqonut: a formally verified JIT compiler for Coq), Tomer Libal (Regularity in higher-order unification), Andrew Polonsky (Defining equality by induction on type structure), Sergei Soloviev (Isomorphism of Dependent Products in Type Theory: Specifics and Scientific Context), Noam Zeilberger (Functors are Type Refinement Systems), Gabriel Scherer (Which types have a unique inhabitant?), Andrej Bauer (A sound and complete language for type theory with equality reflection), Nicolas Tabareau (Vers un analogue de l'axiome de Giraud en HoTT), Alexandre Miquel (An axiomatic presentation of forcing, or: forcing for the dummies), Danko Ilik (The exp-log normal form of formulas).

8.1.12. Groupe de travail Catégories supérieures, polygraphes et homotopie

Pierre-Louis Curien and Yves Guiraud, together with Cyrille Chenavier, Maxime Lucas and Jovana Obradović actively participate in this weekly working group of PPS, organised by François Métayer since 2009.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Master: Pierre-Louis Curien teaches in the course Models of programming languages: domains, categories, games of the MPRI (together with Thomas Ehrhard and Paul-André Melliès).

Alexis Saurin chairs LMFI M2 since September 2013.

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

In addition to his regular teaching duties for the Computer Science department of Paris 7 (M1 and M2 Pro), Pierre Letouzey taught 2 courses in the LMFI M2 in 2015 : initiation to formal proofs on computer (24h) and models of programming (24h).

Matthieu Sozeau teaches in the MPRI course on Advanced uses of Proof Assistants (12 hours + a project), together with Assia Mahboubi (Inria SpecFun).

Licence and Master: Lourdes González had a temporary research and teaching position (A.T.E.R) at University Paris 7 for the academic year 2014–2015. During the second semester (Jan - Aug 2015) she was in charge of TP (Travaux pratiques, 44 hours) on the subjects “Projet Informatique” (L2) and “Introduction aux environnements de dévelopement et génie logiciel” (L3). During the second semester (Jan - Aug 2015) she was in charge of TP (Travaux pratiques, 24 hours) on the subject “Programmation système” (M1).

Licence: Étienne Miquey was in charge of practical sessions in the course “Analyse de données structurées” (36h, L2) and “Programmation Orientée Objet et Interface Graphique” (60h, L2) at University Paris 7.

MOOC: In collaboration with Roberto DiCosmo and Ralf Treinen, Yann Régis-Gianas has created a MOOC about the OCaml programming language.

8.2.2. Supervision

Internship: Yann Régis-Gianas has supervised the M2 internships of Béatrice Carré and Jacques Pascal Deplaix, and the M1 internships of Lélio Brun and of Loïc Runarvot.

Internship: Matthieu Sozeau has supervised the M2 internships of Gabriel Lewertowsky (with Nicolas Tabareau), who is now starting a PhD under their joint supervision, and of Cyprien Mangin, who is starting a PhD under the supervision of Bruno Barras and Matthieu Sozeau.

Internship: Yves Guiraud has supervised the M2 internship of Pierre Giraud.

PhD in progress: Guillaume Claret, supervised by Hugo Herbelin and Yann Régis-Gianas.

PhD in progress: Thibaut Girka, supervised by Roberto DiCosmo and Yann Régis-Gianas.

PhD in progress: Cyrille Chenavier, supervised by Yves Guiraud and Philippe Malbos.

PhD in progress: Maxime Lucas, supervised by Yves Guiraud and Pierre-Louis Curien.

PhD in progress: Jovana Obradović, supervised by Pierre-Louis Curien.

PhD in progress: Amina Doumane, supervised by Alexis Saurin, David Baelde and Pierre-Louis Curien.

PhD in progress: Étienne Miquey, supervised by Hugo Herbelin and Alexandre Miquel.

Pierre-Marie Pédrot and Lourdes González defended their PhD in 2015.

8.2.3. Juries

PhD Juries: Pierre-Louis Curien has been reviewer (rapporteur) for the PhD thesis of Fabio Zanasi (Lyon, October 2015).

Examinator tasks: In June and July 2015, Pierre Letouzey has participated to the jury of the competitive examination for the entrance to Ecole Normale Supérieure (ENS Ulm). More precisely, he has been responsible for the oral test of computer science and has been examiner for about forty candidates, one hour each.

8.3. Popularisation

Étienne Miquey took part in the animation of several activities about mathematics in primary and high schools of Paris.

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 organised, and Étienne Miquey took part in the animation of the (computer science part of the) “Fête de la Science” event at the University Paris 7. Yann Régis-Gianas gave several presentations about “What is programming?” in primary and high schools of Paris.

Jean-Jacques Lévy gave a presentation about “Science et Informatique” at the primary school le Coteau, Vaucresson (November).

9. Bibliography

Major publications by the team in recent years

[1] R. M. AMADIO, Y. RÉGIS-GIANAS. *Certifying and reasoning about cost annotations of functional programs*, in "Higher-Order and Symbolic Computation", January 2013, https://hal.inria.fr/inria-00629473

[2] 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

[3] P.-L. CURIEN. *Operads, clones, and distributive laws*, in "Operads and Universal Algebra : Proceedings of China-France Summer Conference", Tianjin, China, L. G. CHENGMING BAI, J.-L. LODAY (editors), Nankai Series in Pure, Applied Mathematics and Theoretical Physics, Vol. 9, World Scientific, July 2010, pp. 25-50, https://hal.archives-ouvertes.fr/hal-00697065

[4] P.-L. CURIEN, R. GARNER, M. HOFMANN. *Revisiting the categorical interpretation of dependent type theory*, in "Theoretical computer Science", 2014, vol. 546, pp. 99-119, http://dx.doi.org/10.1007/s10990-007-9006-0

[5] 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, pp. 233–243 [*DOI :* 10.1145/351240.351262], http://hal.archives-ouvertes.fr/inria-00156377/en/

[6] P.-L. CURIEN, H. HERBELIN. *Abstract machines for dialogue games*, in "Interactive models of computation and program behavior", Panoramas et Synthèses, Société Mathématique de France, 2009, pp. 231-275, https://hal.archives-ouvertes.fr/hal-00155295

[7] Y. GUIRAUD, P. MALBOS. *Higher-dimensional normalisation strategies for acyclicity*, in "Advances in Mathematics", 2012, vol. 231, n^{o} 3-4, pp. 2294-2351 [*DOI :* 10.1016/J.AIM.2012.05.010], https://hal.archives-ouvertes.fr/hal-00531242

[8] Y. GUIRAUD, P. MALBOS, S. MIMRAM. *A Homotopical Completion Procedure with Applications to Coherence of Monoids*, in "RTA - 24th International Conference on Rewriting Techniques and Applications - 2013", Eindhoven, Netherlands, F. VAN RAAMSDONK (editor), Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, June 2013, vol. 21, pp. 223-238 [*DOI :* 10.4230/LIPICS.RTA.2013.223], https://hal.inria.fr/hal-00818253

[9] 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/

[10] 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, Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, 25-28 June 2012, Dubrovnik, Croatia, IEEE Computer Society, June 2012, pp. 365-374, https://hal.inria.fr/hal-00697240

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

[12] 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, pp. 409–423

[13] 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, pp. 305–335, http://gallium.inria.fr/~fpottier/publis/regis-gianas-pottier-hoarefp.ps.gz

[14] 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, pp. 356-365

[15] 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, pp. 278-293

Publications of the year

Doctoral Dissertations and Habilitation Theses

[16] L. D. C. GONZALEZ HUESCA. *Incrementality and effect simulation in the simply typed lambda calculus*, Universite Paris Diderot-Paris VII, November 2015, https://hal.inria.fr/tel-01248531

[17] P.-M. PÉDROT. *A Materialist Dialectica*, Paris Diderot, September 2015, https://hal.archives-ouvertes.fr/tel-01247085

Articles in International Peer-Reviewed Journals

[18] F. DABROWSKI, F. LOULERGUE, T. PINSARD. *A formal semantics of nested atomic sections with thread escape*, in "Computer Languages, Systems and Structures", 2015, 24 p. , https://hal.inria.fr/hal-01143199

[19] S. GAUSSENT, Y. GUIRAUD, P. MALBOS. *Coherent presentations of Artin monoids*, in "Compositio Mathematica", 2015, vol. 151, n^{o} 5, pp. 957-998 [*DOI :* 10.1112/S0010437X14007842], https://hal.archives-ouvertes.fr/hal-00682233

[20] H. HERBELIN. *A dependently-typed construction of semi-simplicial types*, in "Mathematical Structures in Computer Science", June 2015, vol. 25, n^{o} Special issue 05, 16 p. [*DOI :* 10.1017/S0960129514000528], https://hal.inria.fr/hal-00935446

[21] F. LOULERGUE, W. BOUSDIRA, J. TESSON. *Calculating Parallel Programs in Coq using List Homomorphisms*, in "International Journal of Parallel Programming", 2016, 20 p. , https://hal.inria.fr/hal-01159182

[22] É. MIQUEY, M. GUILLERMO. *Classical realizability and arithmetical formulæ*, in "Mathematical Structures in Computer Science", 2016 [*DOI :* 10.1017/S0960129515000559], https://hal.inria.fr/hal-01247989

International Conferences with Proceedings

[23] B. ALLAN, N. KOSMATOV, M. LEMERRE, F. LOULERGUE. *A case study on formal verification of the Anaxagoros hypervisor paging system with Frama-C*, in "International Workshop on Formal Methods for Industrial Critical Systems (FMICS)", Oslo, Norway, Springer, June 2015, https://hal.inria.fr/hal-01140271

[24] D. BAELDE, A. DOUMANE, A. SAURIN. *Least and Greatest Fixed Points in Ludics* , in "24th EACSL Annual Conference on Computer Science Logic (CSL 2015)", Berlin, Germany, 24th EACSL Annual Conference on Computer Science Logic (CSL 2015), Leibniz International Proceedings in Informatics (LIPIcs), September 2015, vol. 41, pp. 549–566 [*DOI :* 10.4230/LIPICS.CSL.2015.549], https://hal.archives-ouvertes.fr/hal-01256773

[25] M. BAGNOL, A. DOUMANE, A. SAURIN. *On the dependencies of logical rules*, in "FOSSACS, 18th International Conference on Foundations of Software Science and Computation Structures", London, United Kingdom, April 2015, https://hal.archives-ouvertes.fr/hal-01110340

[26] P.-L. CURIEN, M. FIORE, G. MUNCH-MACCAGNONI. *A theory of effects and resources: adjunction models and polarised calculi*, in "Principles of Programmming Languages", Saint-Petersbourg, Florida, United States, Proceedings POPL 2016, January 2016 [*DOI :* 10.1145/2837614.2837652], https://hal.archives-ouvertes.fr/hal-01256092

[27] F. DABROWSKI, F. LOULERGUE, T. PINSARD. *Nested atomic sections with thread escape: Compilation to threads and locks*, in "ACM Symposium on Applied Computing (SAC)", Salamanca, Spain, ACM, April 2015, https://hal.inria.fr/hal-01105093

[28] A. GUESMI, P. CLEMENTE, F. LOULERGUE, P. BERTHOMÉ. *Cloud Resources Placement Based on Functional and Non-Functional Requirements*, in "SECRYPT", Colmar, France, SCITEPRESS, July 2015, https://hal.inria.fr/hal-01158134

[29] C. MANGIN, M. SOZEAU. *Equations for Hereditary Substitution in Leivant's Predicative System F: A Case Study*, in "Tenth International Workshop on Logical Frameworks and Meta Languages: Theory and Practice", Berlin, Germany, EPTCS, August 2015, vol. 185 [*DOI :* 10.4204/EPTCS.185.5], https://hal.inria.fr/hal-01248807

[30] P.-M. PÉDROT, A. SAURIN. *Classical by-need*, in "European Symposium on Programming", Eindhoven, Netherlands, European Symposium on Programming, April 2016, https://hal.archives-ouvertes.fr/hal-01257348

[31] B. ZILIANI, M. SOZEAU. *A unification algorithm for Coq featuring universe polymorphism and overloading*, in "20th ACM SIGPLAN International Conference on Functional Programming", Vancouver, Canada, September 2015 [*DOI :* 10.1145/2784731.2784751], https://hal.inria.fr/hal-01248804

Conferences without Proceedings

[32] G. CLARET, Y. RÉGIS-GIANAS. *Mechanical Verification of Interactive Programs Specified by Use Cases*, in "3rd IEEE/ACM FME Workshop on Formal Methods in Software Engineering", FLoren, France, May 2015 [*DOI :* 10.1109/FORMALISE.2015.17], https://hal.inria.fr/hal-01255107

[33] T. GIRKA, D. MENTRÉ, Y. RÉGIS-GIANAS. *A Mechanically Checked Generation of Correlating Programs Directed by Structured Syntactic Differences*, in "Automated Technology for Verification and Analysis", Shanghai, China, Lecture Notes in Computer Science, October 2015, vol. 9364 [*DOI :* 10.1007/978-3-319-24953-7_6], https://hal.inria.fr/hal-01238704

[34] H. HERBELIN, É. MIQUEY. *Toward dependent choice: a classical sequent calculus with dependent types*, in "TYPES 2015", Tallinn, Estonia, May 2015, https://hal.inria.fr/hal-01247998

[35] F. LOULERGUE. *Construction de programmes parallèles en Coq avec des homomorphismes de listes*, in "14èmes journées Approches Formelles dans l'Assistance au Développement Logiciel (AFADL)", Bordeaux, France, 2015, https://hal.inria.fr/hal-01158137

[36] F. LOULERGUE. *Modèles fonctionnels de MapReduce en Coq*, in "Journées Nationales du GdR GPL", Bordeaux, France, 2015, https://hal.inria.fr/hal-01158138

[37] É. MIQUEY. *Realizability games for arithmetical formulæ* , in "Worskshop GaLoP2015", Londres, United Kingdom, April 2015, https://hal.inria.fr/hal-01248093

Scientific Books (or Scientific Book chapters)

[38] F. LOULERGUE, W. BOUSDIRA, J. TESSON. *Calcul de programmes parallèles avec Coq*, in "Informatique Mathématique", collection Alpha, CNRS Éditions, March 2015, https://hal.inria.fr/hal-01107296

Research Reports

[39] P. LETOUZEY. *Hofstadter's problem for curious readers*, Université Paris Diderot ; Inria Paris-Rocquencourt, September 2015, 29 p. , https://hal.inria.fr/hal-01195587

Other Publications

[40] D. BAELDE, A. DOUMANE, A. SAURIN. *Least and Greatest Fixed Points in Ludics*, July 2015, This document corresponds to the long version of a paper accepted for publication at CSL 2015, https://hal.inria.fr/hal-01178396

[41] R. CHEN, J.-J. LEVY. *Readable semi-automatic formal proofs of Depth-First Search in graphs using Why3*, November 2015, working paper or preprint, https://hal.inria.fr/hal-01253136

[42] C. CHENAVIER. *Confluence algebras and acyclicity of the Koszul complex*, April 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01141738

[43] P.-L. CURIEN, J. OBRADOVIC. *On the various definitions of cyclic operads*, January 2016, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01254649

[44] P. DEHORNOY, Y. GUIRAUD. *Quadratic normalisation in monoids*, April 2015, 27 pages, https://hal.archives-ouvertes.fr/hal-01141226

[45] M. LUCAS. *A coherence theorem for pseudonatural transformations*, September 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01191867

[46] C. MANGIN. *Eliminating Dependent Pattern-Matching in Coq*, Université Paris 7 Denis Diderot, August 2015, https://hal.inria.fr/hal-01250855

[47] J. OBRADOVIC. *The Bénabou-Roubaud monadic descent theorem via string diagrams*, January 2016, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01254637

[48] T. ZIMMERMANN, H. HERBELIN. *Automatic and Transparent Transfer of Theorems along Isomorphisms in the Coq Proof Assistant*, May 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01152588

References in notes

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

[50] R. BERGER. *Confluence and Koszulity*, in "J. Algebra", 1998, vol. 201, n^{o} 1, pp. 243–283

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

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

[53] 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

[54] 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

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

[56] U. DAL LAGO, C. FAGGIAN, B. VALIRON, A. YOSHIMIZU. *Parallelism and Synchronization in an Infinitary Context*, in "Proc. LICS 2015", 2015

[57] P. DEHORNOY, F. DIGNE, E. GODELLE, D. KRAMMER, J. MICHEL. *Foundations of Garside theory*, EMS Tracts Math., European Mathematical Society, 2015, vol. 22, xiv + 689 pages p.

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

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

[60] 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, pp. 446-457

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

[62] 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° 63, pp. 63-92

[63] 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, pp. 47–57

[64] Y. GUIRAUD, E. HOFFBECK, P. MALBOS. *Linear polygraphs and Koszulity of algebras*, June 2014, 42 pages, https://hal.archives-ouvertes.fr/hal-01006220

[65] Y. GUIRAUD, P. MALBOS. *Coherence in monoidal track categories*, in "Math. Structures Comput. Sci.", 2012, vol. 22, n° 6, pp. 931–969

[66] 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

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

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

[69] P. LANDIN. *The mechanical evaluation of expressions*, in "The Computer Journal", January 1964, vol. 6, n° 4, pp. 308–320

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

[71] P. MARTIN-LÖF. *A theory of types*, University of Stockholm, 1971, n° 71-3

[72] 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, pp. 361-380, http://www.informatik.uni-trier.de/~ley/pers/hd/p/Parigot:Michel.html

[73] 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, pp. 717–740

[74] 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, pp. 408-423

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

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

[77] N. DE BRUIJN. *AUTOMATH, a language for mathematics*, Technological University Eindhoven, November 1968, n° 66-WSK-05