IN PARTNERSHIP WITH:
CNRS

Université Denis Diderot (Paris 7)

Activity Report 2018

Project-Team PI.R2

Design, study and implementation of languages for proofs and programs

IN COLLABORATION WITH: Institut de Recherche en Informatique Fondamentale

RESEARCH CENTER
Paris

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:

A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.11. - Proof languages
A2.4.3. - Proofs
A7.2. - Logic in Computer Science
A8.1. - Discrete mathematics, combinatorics
A8.4. - Computer Algebra

Other Research Topics and Application Domains:

B6.1. - Software industry
B6.6. - Embedded systems

1. Team, Visitors, External Collaborators

Research Scientists

Thierry Coquand [University of Gothenburg, Senior Researcher]
Pierre-Louis Curien [Team leader, CNRS, Senior Researcher, HDR]
Yves Guiraud [Inria, Researcher]
Hugo Herbelin [Inria, Senior Researcher, HDR]
Jean-Jacques Lévy [Inria, Emeritus, HDR]
Alexis Saurin [CNRS, Researcher]
Matthieu Sozeau [Inria, Researcher]

Faculty Members

Pierre Letouzey [Univ Denis Diderot, Associate Professor]
Yann Régis-Gianas [Univ Denis Diderot, Associate Professor]

Post-Doctoral Fellows

Eric Finster [Inria]
Kailiang Ji [Inria]
Exequiel Rivas Gadda [Inria]

PhD Students

Antoine Allioux [Inria, from Sep 2018]
Cédric Ho Thanh [Univ Denis Diderot]
Cyprien Mangin [Ecole polytechnique, until Aug 2018]
Théo Zimmermann [Univ Denis Diderot]

Technical staff

Daniel de Rauglaudre [Inria, from Jul 2015]
Thierry Martinez [Inria]

Administrative Assistants

Mathieu Mourey [Inria, from Sep 2018]
Sandrine Verges [Inria, until Jun 2018]

Visiting Scientist

Ying Jiang [Chinese Academy of Sciences, from Nov 2018]

External Collaborators

Thibaut Girka [Univ Denis Diderot, until Sep 2018]
Jovana Obradović [Univ Denis Diderot]

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 [83] 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 [78], then by Howard and de Bruijn at the end of the 60's [95], [114], 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 [75], [76] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [113].

3.1.2. Towards the calculus of constructions

The λ-calculus, defined by Church [73], 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 [66].

To explain 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 [105].

In 1985, Coquand and Huet [75], [76] in the Formel team of Inria-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [84], [109]. 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 lists). 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 [77] 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

During 1984-2012 period, 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 was six years ago. 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 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 mainly by employees of Inria, the CNAM, Paris 7 and MINES.

In the last six years, Hugo Herbelin and Matthieu Sozeau coordinated the development of the system, the official coordinator hat passed from Hugo to Matthieu in August 2016. The ecosystem and development model changed greatly during this period, with a move towards an entirely distributed development model, integrating contributions from all over the world. While the system had always been open-source, its development team was relatively small, well-knit and gathered regularly at Coq working groups, and many developments on Coq were still discussed only by the few interested experts.

The last years saw a big increase in opening the development to external scrutiny and contributions. This was supported by the "core" team which started moving development to the open GitHub platform (including since 2017 its bug-tracker [60] and wiki), made its development process public, starting to use public pull requests to track the work of developers, organising yearly hackatons/coding-sprints for the dissemination of expertise and developers & users meetings like the Coq Workshop and CoqPL, and, perhaps more anecdotally, retransmitting Coq working groups on a public YouTube channel.

This move was also supported by the hiring of Maxime Dénès in 2016 as an Inria research engineer (in Sophia-Antipolis), and the work of Matej Košík (2-year research engineer). Their work involved making the development process more predictable and streamlined and to provide a higher level of quality to the whole system. In September 2018, a second engineer, Vincent Laporte, was hired. Yves Bertot, Maxime Dénès and Vincent Laporte are developing the Coq consortium, which aims to become the incarnation of the global Coq community and to offer support for our users.

Today, the development of Coq involves participants from the Inria project-teams pi.r2 (Paris), Marelle (Sophia-Antipolis), Toccata (Saclay), Gallinette (Nantes), Gallium (Paris), and Camus (Strasboug), the LIX at École Polytechnique and the CRI Mines-ParisTech. Apart from those, active collaborators include members from MPI-Saarbrucken (D. Dreyer's group), KU Leuven (B. Jacobs group), MIT CSAIL (A. Chlipala's group, which hosted an Inria/MIT engineer, and N. Zeldovich's group), the Institute for Advanced Study in Princeton (from S. Awodey, T. Coquand and V. Voevodsky's Univalent Foundations program) and Intel (M. Soegtrop). The latest released version Coq 8.8.0 had 40 contributors (counted from the start of 8.8 development) and the upcoming Coq 8.9 has 54.

On top of the developer community, there is a much wider user community, as Coq is being used in many different fields. The Software Foundations series, authored by academics from the USA, along with the reference Coq'Art book by Bertot and Castéran [67], the more advanced Certified Programming with Dependent Types book by Chlipala [72] and the recent book on the Mathematical Components library by Mahboubi, Tassi et al. provide resources for gradually learning the tool.

In the programming languages community, Coq is being taught in two summer schools, OPLSS and the DeepSpec summer school. For more mathematically inclined users, there are regular Winter Schools in Nice and in 2017 there was a school on the use of the Univalent Foundations library in Birmingham.

Since 2016, Coq also provides a central repository for Coq packages, the Coq opam archive, relying on the OCaml opam package manager and including around 250 packages contributed by users. It would be too long to make a detailed list of the uses of Coq in the wild. We only highlight four research projects relying heavily on Coq. The Mathematical Components library has its origins in the formal proof of the Four Colour Theorem and has grown to cover many areas of mathematics in Coq using the now integrated (since Coq 8.7) SSReflect proof language. The DeepSpec project is an NSF Expedition project led by A. Appel whose aim is full-stack verification of a software system, from machine-checked proofs of circuits to an operating system to a web-browser, entirely written in Coq and integrating many large projects into one. The ERC CoqHoTT project led by N. Tabareau aims to use logical tools to extend the expressive power of Coq, dealing with the univalence axiom and effects. The ERC RustBelt project led by D. Dreyer concerns the development of rigorous formal foundations for the Rust programming language, using the Iris Higher-Order Concurrent Separation Logic Framework in Coq.

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 language.

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, Iris at MPI-Saarbrucken). 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 [85] 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 [102] and Reynolds [108] and started to be studied in an abstract way in the 80's by Felleisen *et al* [81], leading to Parigot's $\lambda\mu$-calculus [106], 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 [101] and Krivine's abstract machine for call-by-name evaluation [98], [97]. 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 [82]: 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.

3.5. Effective higher-dimensional algebra

3.5.1. Higher-dimensional algebra

Like ordinary categories, higher-dimensional categorical structures originate in algebraic topology. Indeed, ∞-groupoids have been initially considered as a unified point of view for all the information contained in the homotopy groups of a topological space X: the *fundamental ∞-groupoid* $\Pi(X)$ of X contains the elements of X as 0-dimensional cells, continuous paths in X as 1-cells, homotopies between continuous paths as 2-cells, and so on. This point of view translates a topological problem (to determine if two given spaces X and Y are homotopically equivalent) into an algebraic problem (to determine if the fundamental groupoids $\Pi(X)$ and $\Pi(Y)$ are equivalent).

In the last decades, the importance of higher-dimensional categories has grown fast, mainly with the new trend of *categorification* that currently touches algebra and the surrounding fields of mathematics. Categorification is an informal process that consists in the study of higher-dimensional versions of known algebraic objects (such as higher Lie algebras in mathematical physics [65]) and/or of "weakened" versions of those objects, where equations hold only up to suitable equivalences (such as weak actions of monoids and groups in representation theory [80]).

Since a few years, the categorification process has reached logic, with the introduction of homotopy type theory. After a preliminary result that had identified categorical structures in type theory [94], it has been observed recently that the so-called "identity types" are naturally equiped with a structure of ∞-groupoid: the 1-cells are the proofs of equality, the 2-cells are the proofs of equality between proofs of equality, and so on. The striking resemblance with the fundamental ∞-groupoid of a topological space led to the conjecture that homotopy type theory could serve as a replacement of set theory as a foundational language for different fields of mathematics, and homotopical algebra in particular.

3.5.2. Higher-dimensional rewriting

Higher-dimensional categories are algebraic structures that contain, in essence, computational aspects. This has been recognised by Street [112], and independently by Burroni [71], when they have introduced the concept of *computad* or *polygraph* as combinatorial descriptions of higher categories. Those are directed presentations of higher-dimensional categories, generalising word and term rewriting systems.

In the recent years, the algebraic structure of polygraph has led to a new theory of rewriting, called *higher-dimensional rewriting*, as a unifying point of view for usual rewriting paradigms, namely abstract, word and term rewriting [99], [104], [86], [87], and beyond: Petri nets [89] and formal proofs of classical and linear logic have been expressed in this framework [88]. Higher-dimensional rewriting has developed its own methods to analyse computational properties of polygraphs, using in particular algebraic tools such as derivations to prove termination, which in turn led to new tools for complexity analysis [68].

3.5.3. Squier theory

The homotopical properties of higher categories, as studied in mathematics, are in fact deeply related to the computational properties of their polygraphic presentations. This connection has its roots in a tradition of using rewriting-like methods in algebra, and more specifically in the work of Anick [63] and Squier [111], [110] in the 1980s: Squier has proved that, if a monoid M can be presented by a *finite*, *terminating* and *confluent* rewriting system, then its third integral homology group $H_3(M, \mathbb{Z})$ is finitely generated and the monoid M has *finite derivation type* (a property of homotopical nature). This allowed him to conclude that finite convergent rewriting systems were not a universal solution to decide the word problem of finitely generated monoids. Since then, Yves Guiraud and Philippe Malbos have shown that this connection was part of a deeper unified theory when formulated in the higher-dimensional setting [14], [15], [91], [92], [93].

In particular, the computational content of Squier's proof has led to a constructive methodology to produce, from a convergent presentation, *coherent presentations* and *polygraphic resolutions* of algebraic structures, such as monoids [14] and algebras [31]. A coherent presentation of a monoid M is a 3-dimensional combinatorial object that contains not only a presentation of M (generators and relations), but also higher-dimensional cells, each of which corresponding to two fundamentally different proofs of the same equality: this is, in essence, the same as the proofs of equality of proofs of equality in homotopy type theory. When this process of "unfolding" proofs of equalities is pursued in every dimension, one gets a polygraphic resolution of the starting monoid M. This object has the following desirable qualities: it is free and homotopically equivalent to M (in the canonical model structure of higher categories [100], [64]). A polygraphic resolution of an algebraic object X is a faithful formalisation of X on which one can perform computations, such as homotopical or homological invariants of X. In particular, this has led to new algorithms and proofs in representation theory [10], and in homological algebra [90][31].

4. Highlights of the Year

4.1. Highlights of the Year

4.1.1. Awards

Matthieu Sozeau received a Distinguished Paper award at ICFP 2018 for his work on "Equivalences for Free!"[36], together with co-authors Nicolas Tabareau and Eric Tanter.

Amina Doumane received in January 2018 the best paper award given by *La Recherche* for her paper in LICS 2017 entitled *Constructive Completeness for the Linear-Time mu-Calculus* for which she already received the Kleene Award from the LICS conference in 2017.

Amina Doumane received the Ackermann Award from the EACSL committee. As a result, she was invited to give a lecture at CSL 2018.

5. New Software and Platforms

5.1. Coq

The Coq Proof Assistant
KEYWORDS: Proof - Certification - Formalisation
SCIENTIFIC DESCRIPTION: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.
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, ...

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.8.2 contains the result of refinements and stabilization of features and deprecations, cleanups of the internals of the system along with a few new features.

Summary of changes:

Kernel: fix a subject reduction failure due to allowing fixpoints on non-recursive values (#407), by Matthieu Sozeau. Handling of evars in the VM (#935) by Pierre-Marie Pédrot.

Notations: many improvements on recursive notations and support for destructuring patterns in the syntax of notations by Hugo Herbelin.

Proof language: tacticals for profiling, timing and checking success or failure of tactics by Jason Gross. The focusing bracket { supports single-numbered goal selectors, e.g. 2:{, (#6551) by Théo Zimmermann.

Vernacular: cleanup of definition commands (#6653) by Vincent Laporte and more uniform handling of the Local flag (#1049), by Maxime Dénès. Experimental Show Extraction command (#6926) by Pierre Letouzey. Coercion now accepts Prop or Type as a source (#6480) by Arthur Charguéraud. Export modifier for options allowing to export the option to modules that Import and not only Require a module (#6923), by Pierre-Marie Pédrot.

Universes: many user-level and API level enhancements: qualified naming and printing, variance annotations for cumulative inductive types, more general constraints and enhancements of the minimization heuristics, interaction with modules by Gaëtan Gilbert, Pierre-Marie Pédrot and Matthieu Sozeau.

Library: Decimal Numbers library (#6599) by Pierre Letouzey and various small improvements.

Documentation: a large community effort resulted in the migration of the reference manual to the Sphinx documentation tool. The new documentation infrastructure (based on Sphinx) is by Clément Pit-Claudel. The migration was coordinated by Maxime Dénès and Paul Steckler, with some help of Théo Zimmermann during the final integration phase. The 14 people who ported the manual are Calvin Beck, Heiko Becker, Yves Bertot, Maxime Dénès, Richard Ford, Pierre Letouzey, Assia Mahboubi, Clément Pit-Claudel, Laurence Rideau, Matthieu Sozeau, Paul Steckler, Enrico Tassi, Laurent Théry, Nikita Zyuzin.

Tools: experimental -mangle-names option to coqtop/coqc for linting proof scripts (#6582), by Jasper Hugunin. Main changes:

Critical soundness bugs were fixed between versions 8.8.0 and 8.8.2, and a PDF version of the reference manual was made available. The Windows installer also includes many more external packages that can be individually selected for installation.

On the implementation side, the dev/doc/changes.md file documents the numerous changes to the implementation and improvements of interfaces. The file provides guidelines on porting a plugin to the new version.

More information can be found in the CHANGES file. Feedback and bug reports are extremely welcome.

Distribution Installers for Windows 32 bits (i686), Windows 64 bits (x8_64) and macOS are available. They come bundled with CoqIDE. Windows binaries now include the Bignums library.

Complete sources of the files installed by the Windows installers are made available, to comply with license requirements.
NEWS OF THE YEAR: Version 8.8.0 was released in April 2018 and version 8.8.2 in September 2018. This is the third release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.7 and was based on a public road-map. It attracted many external contributions. Code reviews and continuous integration testing were systematically used before integration of new features, with an important focus given to compatibility and performance issues.

The main advances in this version are cleanups and fixes in the many different components of the system, ranging from low level kernel fixes to advances in the support of notations and tacticals for selecting goals. A large community effort was made to move the documentation to the Sphinx format, providing a more accessible online ressource to users.

- Participants: Abhishek Anand, C. J. Bell, Yves Bertot, Frédéric Besson, Tej Chajed, Pierre Courtieu, Maxime Denes, Julien Forest, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Benjamin Grégoire, Jason Gross, Hugo Herbelin, Ralf Jung, Matej Kosik, Sam Pablo Kuper, Xavier Leroy, Pierre Letouzey, Assia Mahboubi, Cyprien Mangin, Érik Martin-Dorel, Olivier Marty, Guillaume Melquiond, Pierre-Marie Pédrot, Benjamin C. Pierce, Lars Rasmusson, Yann Régis-Gianas, Lionel Rieg, Valentin

Robert, Thomas Sibut-Pinote, Michael Soegtrop, Matthieu Sozeau, Arnaud Spiwack, Paul Steckler, George Stelle, Pierre-Yves Strub, Enrico Tassi, Hendrik Tews, Laurent Théry, Amin Timany, Vadim Zaliva and Théo Zimmermann

- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- Publication: The Coq Proof Assistant, version 8.8.0
- URL: http://coq.inria.fr/

5.2. Equations

KEYWORDS: Coq - Dependent Pattern-Matching - Proof assistant - Functional programming
SCIENTIFIC DESCRIPTION: Equations is a tool designed to help with the definition of programs in the setting of dependent type theory, as implemented in the Coq proof assistant. Equations provides a syntax for defining programs by dependent pattern-matching and well-founded recursion and compiles them down to the core type theory of Coq, using the primitive eliminators for inductive types, accessibility and equality. In addition to the definitions of programs, it also automatically derives useful reasoning principles in the form of propositional equations describing the functions, and an elimination principle for calls to this function. It realizes this using a purely definitional translation of high-level definitions to core terms, without changing the core calculus in any way, or using axioms.
FUNCTIONAL DESCRIPTION: Equations is a function definition plugin for Coq (supporting Coq 8.6 and 8.7), that allows the definition of functions by dependent pattern-matching and well-founded, mutual or nested structural recursion and compiles them into core terms. It automatically derives the clauses equations, the graph of the function and its associated elimination principle.

Equations is based on a simplification engine for the dependent equalities appearing in dependent eliminations that is also usable as a separate tactic, providing an axiom-free variant of dependent destruction. The main features of Equations include:

Dependent pattern-matching in the style of Agda/Epigram, with inaccessible patterns, with and where clauses. The use of the K axiom or a proof of K is configurable.

Support for well-founded recursion using by rec annotations, and automatic derivation of the subterm relation for inductive families.

Support for mutual and nested structural recursion using with and where auxilliary definitions, allowing to factor multiple uses of the same nested fixpoint definition. It proves the expected elimination principles for mutual and nested definitions.

Automatic generation of the defining equations as rewrite rules for every definition.

Automatic generation of the unfolding lemma for well-founded definitions (requiring only functional extensionality).

Automatic derivation of the graph of the function and its elimination principle. In case the automation fails to prove these principles, the user is asked to provide a proof.

A new dependent elimination tactic based on the same splitting tree compilation scheme that can advantageously replace dependent destruction and sometimes inversion as well. The as clause of dependent elimination allows to specify exactly the patterns and naming of new variables needed for an elimination.

A set of Derive commands for automatic derivation of constructions from an inductive type: its signature, no-confusion property, well-founded subterm relation and decidable equality proof, if applicable.

NEWS OF THE YEAR: Equations 1.0 was released in december this year, after 7 years of (non-continuous) development. It provides the first feature-full version of the software. It has been tried and tested on small to medium scale examples (available on the website). Equations was presented at the Type Theory Tools EUTypes meeting in January 2017 in Paris, and another demo/presentation will be given at PEPM 2018 in Los Angeles in January 2018.

- Participants: Matthieu Sozeau and Cyprien Mangin
- Contact: Matthieu Sozeau
- Publications: Equations reloaded - Equations for Hereditary Substitution in Leivant's Predicative System F: A Case Study - Equations: A Dependent Pattern-Matching Compiler
- URL: http://mattam82.github.io/Coq-Equations/

5.3. Rewr

Rewriting methods in algebra
KEYWORDS: Computer algebra system (CAS) - Rewriting systems - Algebra
FUNCTIONAL DESCRIPTION: Rewr is a prototype of computer algebra system, using rewriting methods to compute resolutions and homotopical invariants of monoids. The library implements various classical constructions of rewriting theory (such as completion), improved by experimental features coming from Garside theory, and allows homotopical algebra computations based on Squier theory. Specific functionalities have been developed for usual classes of monoids, such as Artin monoids and plactic monoids.
NEWS OF THE YEAR: Rewr has been extended with the experimental KGB completion algorithm, based on Knuth-Bendix completion procedure improved by techniques coming from Garside theory.

- Participants: Yves Guiraud and Samuel Mimram
- Contact: Yves Guiraud
- Publications: Higher-dimensional categories with finite derivation type - Higher-dimensional normalisation strategies for acyclicity - Coherent presentations of Artin monoids - A Homotopical Completion Procedure with Applications to Coherence of Monoids - Polygraphs of finite derivation type - Quadratic normalisation in monoids
- URL: http://www.lix.polytechnique.fr/Labo/Samuel.Mimram/rewr

5.4. Catex

KEYWORDS: LaTeX - String diagram - Algebra
FUNCTIONAL DESCRIPTION: Catex is a Latex package and an external tool to typeset string diagrams easily from their algebraic expression. Catex works similarly to Bibtex.
NEWS OF THE YEAR: It is now possible to add labels to objects and morphisms

- Participant: Yves Guiraud
- Contact: Yves Guiraud
- URL: https://www.irif.fr/~guiraud/catex/catex.zip

5.5. Cox

KEYWORDS: Computer algebra system (CAS) - Rewriting systems - Algebra
FUNCTIONAL DESCRIPTION: Cox is a Python library for the computation of coherent presentations of Artin monoids, with experimental features to compute the lower dimensions of the Salvetti complex.

- Participant: Yves Guiraud
- Contact: Yves Guiraud
- Publications: Coherent presentations of Artin monoids - A Homotopical Completion Procedure with Applications to Coherence of Monoids
- URL: https://www.irif.fr/~guiraud/cox/cox.zip

6. New Results

6.1. Effects in proof theory and programming

Participants: Hugo Herbelin, Yann Régis-Gianas, Alexis Saurin, Exequiel Rivas Gadda.

6.1.1. Interfaces for computational effects

Exequiel Rivas studied the relation between interfaces for computational effects in programming languages: arrows, idioms and monads. Building on previous results of Lindley, Yallop and Wadler, a categorical account was developed by means of monoidal adjunctions. This work was presented in MSFP 2018 [40] and later in SYCO I. Together with Ruben Pieters and Tom Schrijvers, a journal version of the article is currently being prepared that includes this work and previous work on non-monadic handlers. It will be submitted to the Journal of Functional Programming.

6.1.2. Monads with merging

In collaboration with Mauro Jaskelioff, Exequiel Rivas developed monads with merge-like operators. These operators are based on two well-known algebraic theories for concurrency: classic process algebras and the more recent concurrent monoids. This resulted in an article submitted to FoSSaCS.

6.1.3. Relative effects: coherence for skew structures

In joint work with Mauro Jaskelioff, Tarmo Uustalu and Niccolò Veltri, Exequiel Rivas developed coherence theorems in the setting of categories with skew structures: skew monoidal categories, skew near-rig categories, skew semigroup categories. These skew structures are motivated by the study of relative effects in programming languages, where the primary example are relative monads. The results are formalised in the programming language Agda. A journal article is currently being written.

6.1.4. Effectful proving

Hugo Herbelin started a program of reconstruction of different levels of computational strength of logic by means of translation to a core logic of polarised linear connectives.

6.1.5. On the computational strength of choice axioms

With the goal of transferring the effectful computational contents of the dependent choice to other forms of choice or bar induction axioms, Hugo Herbelin worked at clarifying the folklore regarding the strengths of various forms of choice and of bar induction.

In collaboration with Boban Velickovic, Alexis Saurin advised the LMFI master internship of Ikram Cherigui on classical realisability and forcing in set theory.

6.1.6. Effectful systems in Coq

In collaboration with Thomas Letan (Agence Nationale pour la Sécurité des Systèmes Informatiques), Pierre Chifflier (ANSSI) and Guillaume Hiet (Centrale Supélec), Yann Régis-Gianas developed a new approach to model and verify effectful systems in Coq. This work has been presented at FM 2018 [38].

6.2. Reasoning and programming with infinite data

Participants: Yann Régis-Gianas, Alexis Saurin, Abhishek De, Luc Pellissier, Xavier Onfroy.

This theme is part of the ANR project Rapido (see the National Initiatives section) which goes until end of september 2019.

6.2.1. Proof theory of infinitary and circular proofs

In collaboration with David Baelde, Amina Doumane, Guilhem Jaber and Denis Kuperberg, Alexis Saurin extended the proof theory of infinite and circular proofs for fixed-point logics in various directions by relaxing the validity condition necessary to distinguish sound proofs from invalid ones. The original validity condition considered by Baelde, Doumane and Saurin in CSL 2016 rules out lots of proofs which are computationally and semantically sound and does not account for the cut-axiom interaction in sequent proofs.

In the setting of sequent calculus, Saurin introduced together with Baelde, Doumane and Jaber a relaxed validity condition to allow infinite branches to be supported by threads bouncing on axioms and cuts. This allows for a much more flexible criterion, inspired from Girard's geometry of interaction. The most general form of this criterion does not ensure productivity due to a discrepancy between the sequential nature of proofs in sequent calculus and the parallel nature of threads. Several directions of research have therefore been investigated from that point:

- In sequent calculus, Baelde, Doumane and Saurin provided a slight restriction of the full bouncing validity which grants productivity and validity of the cut-elimination process. This restriction still strictly extends previous notions of validity and is actually expressive enough to be undecidable as proved together with Kuperberg. Decidability can be recovered by constraining the shapes of bounces. Doumane and Saurin were able in the fall 2018 to generalise the CSL proof technique to be applicable to bouncing threads. Those results are currently being written targetting a submission early 2019.
- In the setting of natural deduction, Saurin and Jaber introduced a validity criterion aiming at ensuring productivity of a circular λ-calculus with inductive and coinductive types.
- In the fall 2018, Abhishek De started his PhD under Saurin's supervision. The first part of his PhD work is dedicated to lifting the proof theory of circular and infinitary proofs to the setting of proof nets, in which the bouncing criterion will be much more convenient to work with since the discrepancy between sequent proofs and parallel threads will be dealt with.

6.2.2. Brotherston-Simpson's conjecture: Finitising circular proofs

An important and most active research topic on circular proofs is the comparison of circular proof systems with usual proof systems with induction and co-induction rules à la Park. This can be viewed as comparing the proof-theoretical power of usual induction reasoning with that of Fermat's infinite descent method. Berardi and Tatsuta, as well as Simpson, obtained in 2017 important results in this direction for logics with inductive predicates à la Martin-Löf. Those frameworks, however, are weaker than those of fixpoint logic which can express and mix least and greatest fixpoints by interleaving μ and ν statements. New results on this topics followed in 2018.

In a work with Nollet and Tasson, Saurin published in CSL 2018 a new validity condition which is quite straightfoward to check (it can be checked at the level of elementary cycles of the circular proofs, while the other criteria need to check a condition on every infinite branch) and still capture all circular proofs obtained from $\mu MALL$ finite proofs [46]. The condition for cycling in those proofs is more constrained than that of Baelde, Doumane and Saurin, but the proof contains more information which can be used to exctract inductive invariants. With this validity condition which can be useful for proof search for circular proofs, they obtained partial finitisation results and are currently aiming at solving the most general Brotherston-Simpson's conjecture.

6.2.3. Streams and classical logic

Luc Pellissier started a post-doc in december 2018 funded by the RAPIDO project and started working with Alexis Saurin on the stream interpretation of $\Lambda\mu$-calculi by investigating the connection between $\Lambda\mu$-calculus and the parsimonious λ-calculus.

6.2.4. Formalising circular proofs and their validity condition

During the spring and summer 2018, Saurin started with Xavier Onfroy a formalisation of circular proofs in Coq. Until now, Onfroy formalised parity-automata and their meta-theory as a first step to capture the decidability condition of circular proofs. Preliminary formalisations of circular proofs have been considered by Onfroy but shall still be pursued in order to fit into the picture.

6.3. Effective higher-dimensional algebra

Participants: Antoine Allioux, Pierre-Louis Curien, Eric Finster, Yves Guiraud, Cédric Ho Thanh, Matthieu Sozeau.

6.3.1. Rewriting methods in algebra

Yves Guiraud has written with Philippe Malbos (Univ. Lyon 1) a survey on the use of rewriting methods in algebra, centered on a formulation of Squier's homotopical and homological theorems in the modern language of higher-dimensional categories. This article is intended as an introduction to the domain, mainly for graduate students, and has appeared in Mathematical Structures in Computer Science [32].

Yves Guiraud has completed a four-year collaboration with Eric Hoffbeck (Univ. Paris 13) and Philippe Malbos (Univ. Lyon 1), whose aim was to develop a theory of rewriting in associative algebras, with a view towards applications in homological algebra. They adapted the known notion of polygraph [71] to higher-dimensional associative algebras, and used these objects to develop a rewriting theory on associative algebras that generalises the two major tools for computations in algebras: Gröbner bases [70] and Poincaré-Birkhoff-Witt bases [107]. Then, they transposed the construction of [14], based on an extension of Squier's theorem [110] in higher dimensions, to compute small polygraphic resolutions of associative algebras from convergent presentations. Finally, this construction has been related to the Koszul homological property, yielding necessary or sufficient conditions for an algebra to be Koszul. The resulting work will appear in Mathematische Zeitschrift [31].

Yves Guiraud has written his "Habilitation à diriger des recherches" manuscript, as a survey on rewriting methods in algebra based on Squier theory [13]. The defense is planned for Spring 2019.

Yves Guiraud works with Dimitri Ara (Univ. Aix-Marseille), Albert Burroni, Philippe Malbos (Univ. Lyon 1), François Métayer (Univ. Nanterre) and Samuel Mimram (École Polytechnique) on a reference book on the theory of polygraphs and higher-dimensional categories, and their applications in rewriting theory and homotopical algebra.

Yves Guiraud works with Marcelo Fiore (Univ. Cambridge) on the theoretical foundations of higher-dimensional algebra, in order to develop a common setting to develop rewriting methods for various algebraic structures at the same time. Practically, they aim at a definition of polygraphic resolutions of monoids in monoidal categories, based on the recent notion of n-oid in an n-oidal category. This theory will subsume the known cases of monoids and associative algebras, and encompass a wide range of objects, such as Lawvere theories (for term rewriting), operads (for Gröbner bases) or higher-order theories (for the λ-calculus).

Opetopes are a formalisation of higher many-to-one operations leading to one of the approaches for defining weak ω-categories. Opetopes were originally defined by Baez and Dolan. A reformulation (leading to a more carefully crafted definition) has been later provided by Batanin, Joyal, Kock and Mascari, based on the notion of polynomial functor. Pierre-Louis Curien, Cédric Ho Thanh and Samuel Mimram have developped (in several variants) a type-theoretical treatment of opetopes and finite opetopic sets, and have shown that the models of their type theory are indeed the opetopic sets as defined mathematically by the above authors. This work is being submitted to an international conference. Also, Cédric Ho Thanh has given a direct precise proof of the equivalence between many-to-one polygraphs and opetopic sets, thus establishing a connection with the theory of polygraphs [57].

6.3.2. Garside methods in algebra and rewriting

Building on [9], Yves Guiraud is currently finishing with Matthieu Picantin (Univ. Paris 7) a work that generalises already known constructions such as the bar resolution, several resolutions defined by Dehornoy and Lafont [79], and the main results of Gaussent, Guiraud and Malbos on coherent presentations of Artin monoids [10], to monoids with a Garside family. This allows an extension of the field of application of the rewriting methods to other geometrically interesting classes of monoids, such as the dual braid monoids.

Still with Matthieu Picantin, Yves Guiraud develops an improvement of the classical Knuth-Bendix completion procedure, called the KGB (for Knuth-Bendix-Garside) completion procedure. The original algorithm tries to compute, from an arbitrary terminating rewriting system, a finite convergent presentation, by adding relations to solve confluence issues. Unfortunately, this algorithm fails on standard examples, like most Artin monoids with their usual presentations. The KGB procedure uses the theory of Tietze transformations, together with Garside theory, to also add new generators to the presentation, trying to reach the convergent Garside presentation identified in [9]. The KGB completion procedure is partially implemented in the prototype Rewr, developed by Yves Guiraud and Samuel Mimram.

6.3.3. Foundations and formalisation of higher algebra

Antoine Allioux (PhD started in February), Eric Finster, Yves Guiraud and Matthieu Sozeau are exploring the development of higher algebra in type theory. To formalise higher algebra, one needs a new source of coherent structure in type theory. Finster has developed an internalisation of polynomial monads (of which opetopes and ∞-categories are instances) in type theory, which ought to provide such a coherent algebraic structure, inspired by the work of Kock et al [96]. Antoine Allioux is focusing on building an equivalence of types between categories seen as polynomial monads and the standard univalent categories in Homotopy Type Theory [22]. Another result that should follow is the ability to define simplicial types in Homotopy Type Theory, a long standing open problem in the field. An article on this subject is in preparation. Once armed with such a definition mechanism for higher algebraic structures and their algebras, it should be possible to internalise results from higher rewriting theory in type theory, which was the initial goal of this project.

6.3.4. Type Theory and Higher Topos Theory

Eric Finster explored the connections between intensional type theory and the theory of higher topoi, as developed in the works on Joyal and Lurie [103]. In particular, in collaboration with Mathieu Anel, André Joyal and Georg Biedermann, he gave a proof of a new result about the generation of left exact modalities in higher topoi, which has a corresponding internalisation in Homotopy Type Theory. Applications of this result to the Goodwillie Calculus, an advanced technique in abstract homotopy theory, resulted in the article [28].

6.4. Incrementality

Participants: Thibaut Girka, Yann Régis-Gianas.

6.4.1. Incrementality in proof languages

In collaboration with Paolo Giarrusso, Philipp Shuster and Yufei Cai (Univ Marburg, Allemagne), Yann Régis-Gianas developed a new method to incrementalise higher-order programs using formal derivatives and static caching. Yann Régis-Gianas has developed a mechanised proof for this transformation as well as a prototype language featuring efficient derivatives for functional programs. A paper has been submitted to ESOP 2019.

In collaboration with Olivier Martinot (Paris Diderot), Yann Régis-Gianas studied a new technique to implement incrementalised operations on lists. A paper is to be submitted to ICFP 2019.

6.4.2. Difference languages

Kostia Chardonnet and Yann Régis-Gianas started the formalisation of difference languages for Java, using the framework developed by Thibaut Girka. In particular, Kostia Chardonnet implemented a mechanised small step operational semantics for a large subset of Java. A paper is in preparation.

6.5. Metatheory and development of Coq

Participants: Hugo Herbelin, Pierre Letouzey, Yann Régis-Gianas, Matthieu Sozeau, Gaëtan Gilbert, Cyprien Mangin, Théo Winterhalter, Théo Zimmermann, Thierry Martinez.

6.5.1. Homotopy type theory

Hugo Herbelin developed the syntax for a variant of Cohen, Coquand, Huber and Mörtberg's Cubical Type Theory where equality on types is defined to be equivalence of types, thus satisfying univalence by construction.

6.5.2. Proof irrelevance and Homotopy Type Theory

Gaëtan Gilbert (PhD student of N. Tabareau, Gallinette and M. Sozeau) continued developing the theory and implementation of *strict* propositions in the calculus of inductive constructions. In collaboration with Jesper Cockx (Chalmers), they developed this notion in full in an article at POPL 19 [30]. Strict propositions enjoy definitional proof-irrelevance and are compatible with both Univalence and Uniqueness of Identity Proofs, providing a foundation for further research in both directions: dealing with strict structures in homotopy type theory, and improving the support for programming with dependent types and proofs. They have shown in particular how to translate inductive types that can be seen as strict propositions into recursively defined types, providing a fix to the "singleton elimination" criterion used in Coq to treat the interaction of propositions (in Prop) and informative objects (in Type). Together with Pierre Letouzey, Matthieu Sozeau is pursuing an adaptation of the Prop sort informed by this new result. In particular, Pierre Letouzey is now experimenting with alternative ways to handle the accessibility arguments of Coq general fixpoints during extraction. Historically, the elimination of these arguments was a consequence of the accessibility inductive type being in Prop. But this can actually be seen as a more general dead-code elimination method. This leverages the need for accessibility to be in sort Prop, and hence opens new prospects concerning the Prop universe and the proof irrelevance.

6.5.3. Extensionality and Intensionality in Type Theory

Théo Winterhalter, Nicolas Tabareau and Matthieu Sozeau studied and formalised a complete translation from Extensional to Intensional Type Theory in Coq, now published at CPP 2019 [43]. They show that, contrary to the original paper proof of Oury, the target intensional type theory only needs to be extended with the Uniqueness of Identity Proofs principle and Functional Extensionality, settling concretely and formally a question that was studied semantically and up-to now only on paper by Hofmann and Altenkirch [61]. The translation was formalised using the Template-Coq framework and gives rise to an executable translation from partial terms of ETT into terms of Coq annotated with transports of equalities. This provides a simple way to justify the consistency of type theories extending the definitional equality relation by provable propositional equalities, and shows the equivalence of 2-level type theory [62] and the Homotopy Type System proposed by Voevodsky.

6.5.4. Dependent pattern-matching and recursion

Cyprien Mangin and Matthieu Sozeau have continued work on the Equations plugin of Coq, Equations now provides means to define nested, mutual and well-founded recursive definitions, together with a definitional compilation of dependent-pattern matching avoiding the use of axioms. In recent work, Matthieu Sozeau uncovered a new way to deal with dependent pattern-matching on inductive families avoiding more uses of the K axiom, inspired by the work of Cockx [74], that integrates well with the simplification engine developed for Equations. An article describing this work is in revision [58].

Thierry Martinez continued the implementation of a dependent pattern-matching compilation algorithm in Coq based on the PhD thesis work of Pierre Boutillier and on the internship work of Meven Bertrand. The algorithm based on small inversion and generalisation is the object of a paper to be submitted to the TYPES post-proceedings.

6.5.5. Explicit Cumulativity

Pierre Letouzey continued exploring with the help of Matthieu Sozeau a version of Coq's logic (CIC) where the cumulativity rule is explicit. This cumulativity rule is a form of coercion between Coq universes, and is done silently in Coq up to now. Having a version of CIC where the use of the cumulativity bewteen Prop and Type is traceable would be of great interest. In particular this would lead to a solid ground for the Coq extraction tool and solve some of its current limitations. Moreover, an explicit cumulativity would also help significantly the studies of Coq theoretical models. A prototype version of Coq is now available, but only a fragment of the standard library has been adapted to explicit cumulativity. In particular, the equalities of equalities currently need some amending, and this process is quite cumbersome.

6.5.6. Cumulativity for Inductive Types

Together with Amin Timany, Matthieu Sozeau developed the Calculus of Cumulative Inductive Constructions which extends the cumulativity relation of universes to universe polymorphic inductive types. This work was presented at FSCD 2018 [42]. The development of the model of this calculus suggested a refinement of the implementation which was integrated in Coq 8.8, providing a more flexible subtyping relation on inductive types in Coq. Notably, this work shrinks the gap to emulate the so-called "template" polymorphism of Coq with cumulative universe polymorphism. Cumulative Inductive Types also provide an apropriate basis to formalise the notions of small and large categories in type theory, avoiding the introduction of coercions. In particular, it provides a way to define a well-behaved category of types and functions and constructions on it, like the Yoneda embedding, which would not be expressible without cumulativity. Finally, Cumulative Inductive Types allow the definition of syntactic models of type theories with cumulativity inside Coq, as pioneered by Boulier *et al* [69].

6.5.7. Mathematical notations in Coq

Hugo Herbelin developed new extensions of the system of mathematical notation of Coq: support for autonomous auxiliary grammars, support for binders over arbitrary patterns, support for generic notations for applications.

6.5.8. Software engineering aspects of the development of Coq

Théo Zimmermann has studied software engineering and open collaboration aspects of the development of Coq.

Following the migration of the Coq bug tracker from Bugzilla to GitHub which he conducted in 2017, he analyzed data (extracted through the GitHub API), in collaboration with Annalí Casanueva Artís from the Paris School of Economics. The results show an increased number of bugs by core developers and an increased diversity of the people commenting bug reports. These results validate *a posteriori* the usefulness of such a switch. A paper [60] has been written and has been presented at the EAQSE workshop (without proceedings). The current objective is to publish the paper in the MSR 2019 conference.

Following discussions dating back from the end of 2017, he has founded the coq-community GitHub organisation in July 2018. This is a project for a collaborative, community-driven effort for the long-term maintenance and advertisement of Coq packages. Already 10 pre-existing Coq projects (plugins and libraries) have been moved to this organisation since then (seven of them are former Coq contribs that were fixed from time to time by the Coq developers themselves – mostly by Hugo Herbelin). The organisation also hosts a "manifesto" repository for general discussion, documentation and advice to developers (including already a few reusable templates for Coq projects), and a docker-coq project to provide reusable Docker images with Coq. The next objectives are to get started on the collaborative documentation (starting with a work by Pierre Castéran from LaBRI) and to create an editorial committee. Théo Zimmermann and Yann Régis-Gianas are preparing an article of the model proposed by the various existing *-community GitHub organisations (including the elm-community organisation from which coq-community was inspired, and ocaml-community which was influenced by coq-community itself).

In addition, Théo Zimmermann has coordinated efforts to improve the documentation of Coq, has documented the release process that he had put in place with Maxime Dénès, and has developed a GitHub / GitLab bot (in OCaml) that is used to automatise many useful functions for the Coq development (continuous integration and backporting of pull requests in particular). The goal is to make this bot modular and reusable for other projects.

6.5.9. Coordination of the development of Coq

The amount of contributions to the Coq system increased significantly in the recent years (around 50 pull-requests are reviewed, discussed and merged each month, approximately). Hugo Herbelin, Matthieu Sozeau and Théo Zimmermann, helped by members from Gallinette (Nantes) and Marelle (Sophia-Antipolis), devoted an important part of their time to coordinate the development, to review propositions of extensions of Coq from external and/or young contributors, and to propose themselves extensions (see the corresponding paragraphs).

6.6. Formalisation and verification

Participants: Pierre-Louis Curien, Kailiang Ji, Pierre Letouzey, Jean-Jacques Lévy, Cyprien Mangin, Daniel de Rauglaudre, Matthieu Sozeau.

6.6.1. Proofs and surfaces

Following ideas of J. Richter-Gebert, Pierre-Louis Curien, together with Jovana Obradović (former PhD student of the team and now postdoc in Prague), joined a project with Zoran Petrić and other Serbian colleagues on formalising proofs of incidence theorems (arising by repeated use of Menelaus theorem) by means of a cyclic sequent calculus, by which is meant that a (proof of a) sequent $\vdash \Gamma$ stands for the conjunction of all (proofs of) traditional sequents $\Gamma \setminus \psi \vdash \psi$. We have designed a proof system, showed its soundness, and experimented it on an extended set of examples from elementary projective geometry. A paper is being written.

6.6.2. Hofstadter nested recursive functions and Coq

Pierre Letouzey continued this year the study of a family of nested recursive functions proposed by D. Hofstadter in his book “Gödel Escher Bach”. This is a generalisation of the earlier work [20], bringing a large number of new insights as well as many new conjectures. Most of the work is already certified in Coq, with generalised and/or nicer proofs, see https://www.irif.fr/~letouzey/hofstadter_g/. Many interactions with Fibonacci numbers or similar recursive sequence have been found. Pierre Letouzey even stumbled upon a Rauzy fractal during this investigation, which is still ongoing.

6.6.3. Real Numbers in Coq

The present Coq library of real numbers is made of 17 axioms. Daniel de Rauglaudre has been studying the possibility of making an implementation with one only axiom: the Limited Principle of Omniscience (LPO) which says that we can differentiate an infinite sequence of 0s from an infinite sequence holding something else than 0 (it seems obvious but it cannot be proved in constructive logic). This axiom had been already used in the formal proof of Puiseux’ theorem done some years ago (only axiom of this proof too).

Real numbers are defined by an infinite sequence of digits and the operations of addition and multiplication by algorithms using LPO.

It was tested in OCaml, the axiom being replaced by a function having a limit corresponding to the precision of the computation and it seems to work. But the proof in Coq that this implementation is a field stumbles on difficulties about the associativity of addition which is more complicated than expected. Several tracks have been experimented with Hugo Herbelin’s help.

6.6.4. Proofs of algorithms on graphs

Jean-Jacques Lévy and Chen Ran (a PhD student at the Institute of Software, Beijing) pursue their work about formal proofs of graph algorithms. Their goal is to provide proofs of algorithms checked by computer and human readable. If these kinds of proofs exist for algorithms on inductive structures or recursive algorithms on arrays, they seem less easy to design for combinatorial structures such as graphs. In 2016, they completed proofs for algorithms computing the strongly connected components in graphs (Kosaraju - 1978 and Tarjan - 1972). Their proofs use the multi-sorted first-order logic with inductive predicates of the Why3 system (research-team Toccata, Saclay). They also widely use the numerous automatic provers interfaced with Why3. A very minor part of these proofs is also achieved in Coq. The difficulty of this approach is to combine automatic provers and the intuitive design. Another point is to define the good level of abstraction in order to avoid too many implementation features while keeping an effective presentation.

In 2017, the same proofs were fully completed in Coq-ssreflect with the Mathematical Components library by Cohen and Théry (research-team Marelle, Sophia-Antipolis), and in Isabelle-HOL by Merz (research-team VeriDis, Nancy), both proofs with the assistance of J.-J. Lévy. These proofs are between a factor 3 to 8 in length with respect to the initial Why3 proofs, but more importantly they look less human readable, mainly because of the absence of automatic deduction and several technicalities about termination. On the way, this collaboration led to a new, better presentation of the Why3 proof.

Part of this work (Tarjan 1972) was presented at JFLA 2017, a more comprehensive version was presented at the VSTTE 2017 conference in Heidelberg. Scripts of proofs can be found at http://jeanjacqueslevy.net/why3, where other proofs of graph algorithms are also present: acyclicity test, articulation points, biconnected components. A proof of Tarjan's planarity test is also under design. A paper entitled "Formal Proofs of Tarjan's Algorithm in Why3, Coq and Isabelle" is under submission to a conference.

6.6.5. Certified compilation and meta-programming

Matthieu Sozeau participates to the CertiCoq project (https://www.cs.princeton.edu/~appel/certicoq) whose aim is to verify a compiler from Coq's Gallina language down to CompCert C-light which provides itself a certified compilation path to assembly language. Matthieu Sozeau focused on the front-end part of CertiCoq, providing formal proofs of the first two phases of the compiler. The first phase translates from Coq syntax to a more amenable representation for metatheoretical study, and the second phase performs extraction to an untyped lambda-calculus with datatypes and mutual (co-)fixpoints. These two phases are of general use and are now integrated and developed in the MetaCoq project. The CertiCoq team expects to release a first version of the compiler in the beginning of 2019, along with an article describing it.

MetaCoq is a project led by Matthieu Sozeau, in collaboration with Simon Boulier and Nicolas Tabareau in Nantes, Abhishek Anand and Gregory Malecha (BedRock Systems, Inc) and Yannick Forster in Saarbrucken. The project was born from the extension of the Template-Coq reification plugin of G. Malecha, which now contains:

- A specification of the typing rules of Coq and its basic metatheoretical properties (weakening, substitution). This specification is not entirely complete yet, as the positivity and guard-checking of definitions is missing. Cyprien Mangin has formalised the regular tree structure used by the guard checker, and a simple positivity check for inductive types. Its integration is ongoing.
- A (partial) proof of the correctness and completeness of a reference type-checker with respect to these rules.
- An implementation of the extraction phase of Coq, which is used in the CertiCoq project. The proof of "syntactic" correctness of this phase, that is the preservation of weak call-by-value reduction by extraction is ongoing.
- A monad giving the ability to program arbitrary plugins in Coq itself, in the style of MTac.

. The foundation of this project was published at ITP 2018 [37], and a journal article is in preparation.

In collaboration with Jan-Oliver Kaiser (MPI-SWS), Beta Ziliani (CONICET/FAMAF), Robbert Krebbers (ICIS) and Derek Dreyer (MPI-SWS), Yann Régis-Gianas participates in the Mtac2 project, a metaprogramming language for Coq. The new version of this language has been presented at ICFP 2018 [34]. It includes in particular in a depedently-typed variant of the LCF tactic typing discipline.

In collaboration with Xavier Denis (Paris Diderot), Yann Régis-Gianas is implementing a compiler for Mtac2.

6.6.6. *Equivalences for free!*

Nicolas Tabareau (Inria Nantes), Eric Tanter (U. Chile in Santiago) and Matthieu Sozeau developed a new parametricity translation for justifying the transport of programs and proofs by equivalences in type theory [36]. Inspired by the Univalence axiom, they show that every construction of type theory (minus inductive families indexed by universes) respect type equivalence, and provide a modified parametricity translation that can be used to construct the proof of invariance by equivalence of any term. This translation is engineered so that transports do not appear during this inference, allowing an easy implementation of a transfer metaprogram in type theory using type class inference. Using this metaprogram, one can automatically transport libraries of implementations and their proofs from one type to an equivalent one, including cases where dependent types are used. While the translation ultimately relies on the univalence axiom to treat universes, its use can be avoided in many cases, providing an effective translation that can be evaluated inside type theory.

6.6.7. *Detecting K-Synchronisability Violations*

Ahmed Bouajjani, Constantin Enea, Kailiang Ji and Shaz Qadeer introduced a bounded analysis that explores a special type of computations, called k-synchronous, for analyzing message passing programs. They gave a procedure for deciding k-*synchronisability* of a program, i.e., whether every computation is equivalent (has the same happens-before relation) to one of its k-synchronous computations. They also showed that reachability over k-synchronous computations and checking k-synchronisability are both PSPACE-complete. Furthermore, they introduced a class of programs called *flow-bounded* for which the problem of deciding whether there exists a $k > 0$ for which the program is k-synchronisable, is decidable. The k-*synchronisability* violation detection algorithm was implemented in Spin model checker. This work was published at CAV 2018 [48].

7. Partnerships and Cooperations

7.1. National Initiatives

Pierre-Louis Curien, Yves Guiraud, Hugo Herbelin, and Alexis Saurin are members of the GDR Informatique Mathématique, in the LHC (Logique, Homotopie, Catégories) and Scalp (Structures formelles pour le calcul et les preuves) working groups. Alexis Saurin is coordinator of the Scalp working group.

Pierre-Louis Curien, Yves Guiraud (local coordinator) and Matthieu Sozeau are members of the GDR 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 and theoretical computer science.

Yves Guiraud is member of the GDR Tresses, federating French researchers working on algebraic, algorithmic and topological aspects of braid groups, low-dimensional topology, and connected subjects.

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 collaborates with ANSSI on the topic of certified ful programming in Coq.

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).

Yann Régis-Gianas and Alexis Saurin (coordinator) are members of the four-year RAPIDO ANR project, started 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 fixpoints 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 IRIF (PPS team), David Baelde from LSV, ENS-Cachan, and Pierre Clairambault, Damien Pous and Colin Riba from LIP, ENS-Lyon.

Matthieu Sozeau is a member of the CoqHoTT project led by Nicolas Tabareau (Gallinette team, Inria Nantes & École des Mines de Nantes), funded by an ERC Starting Grant. The post-doctoral grant of Eric Finster is funded by the CoqHoTT ERC and Amin Timany's 2-month visit was funded on the ERC as well.

7.2. European Initiatives

7.2.1. Collaborations in European Programs, Except FP7 & H2020

Hugo Herbelin is a deputy representative of France in the COST action EUTYPES. The full name of the project (whose scientific leader is Herman Geuvers, from the University of Nijmegen) is "European research network on types for programming and verification".

Presentation of EUTYPES: Types are pervasive in programming and information technology. A type defines a formal interface between software components, allowing the automatic verification of their connections, and greatly enhancing the robustness and reliability of computations and communications. In rich dependent type theories, the full functional specification of a program can be expressed as a type. Type systems have rapidly evolved over the past years, becoming more sophisticated, capturing new aspects of the behaviour of programs and the dynamics of their execution. This COST Action will give a strong impetus to research on type theory and its many applications in computer science, by promoting (1) the synergy between theoretical computer scientists, logicians and mathematicians to develop new foundations for type theory, for example as based on the recent development of "homotopy type theory", (2) the joint development of type theoretic tools as proof assistants and integrated programming environments, (3) the study of dependent types for programming and its deployment in software development, (4) the study of dependent types for verification and its deployment in software analysis and verification. The action will also tie together these different areas and promote cross-fertilisation.

7.3. International Initiatives

7.3.1. IIL projects

Matthieu Sozeau is part of an international collaboration network CSEC "Certified Software Engineering in Coq" funded by Inria Chile, Conicyt and the CoqHoTT ERC, which officially started in early 2018. The participants include Eric Tanter (primary investigator) and Nicolas Tabareau.

7.3.2. Inria Associate Teams Not Involved in an Inria International Labs

7.3.2.1. Associate team

Pierre-Louis Curien and Claudia Faggian are members of the CRECOGI associate team, coordinated on one side by Ugo dal Lago (research-team FoCUS , Inria Sophia and Bologna), and on the other side by Ichiro Hasuo (NII, Tokyo). The full name of the project is Concurrent, Resourceful and full Computation, by Geometry of Interaction.

Presentation of CRECOGI: Game semantics and geometry of interaction (GoI) are two closely related frameworks whose strengh is to have the characters of both a denotational and an operational semantics. They offer a high-level, mathematical (denotational) interpretation, but are interactive in nature. The formalisation in terms of movements of tokens through which programs communicate with each other can actually be seen as a low-level program. The current limit of GoI is that the vast majority of the literature and of the software tools designed around it have a pure, sequential functional language as their source language. This project aims at investigating the application of GoI to concurrent, resourceful, and effectful computation, thus paving the way to the deployment of GoI-based correct-by-construction compilers in real-world software developments in fields like (massively parallel) high-performance computing, embedded and cyberphysical systems, and big data. The presence of both the Japanese GoI community (whose skills are centered around effects and coalgebras) and the French GoI community (more focused on linear logic and complexity analysis) bring essential, complementary, ingredients.

7.3.2.2. Joint Inria-CAS project

Pierre-Louis Curien is principal investigator on the French side for a joint project Inria - Chinese Academy of Sciences. The project's title is "Verification, Interaction, and Proofs". The principal investigator on the Chinese side is Ying Jiang, from the Institute of Software (ISCAS) in Beijing. The participants of the project on the French side are Pierre-Louis Curien and Jean-Jacques Lévy, as well as other members of IRIF (Thomas Ehrhard, Jean Krivine, Giovanni Bernardi, Ahmed Bouajjani, Mihaela Sighireanu, Constantin Enea, Gustavo Petri), and Gilles Dowek (Deducteam team of Inria Saclay). On the Chinese side, the participants are Ying Jiang, as well as other members of the ISCAS (Angsheng Li, Xinxin Liu, Yi Lü, Peng Wu, Yan Rongjie, Zhilin Wu, and Wenhui Zhang), and Yuxi Fu (from Shanghai Jiaotong University). The project funds the postdoc of Kailiang Ji at University Paris 7, that started in December 2017 and will end in Marche 2019.

Presentation of VIP: The line between "verification" and "proofs" is comparable to the one separating satisfiability and provability: in a formal system, a formula can be trusted either if it is satisfied in the intended model (for all of its instances), or if it can be proved formally by using the axioms and inference rules of some logical system. These two directions of work are called model-checking and proof-checking, respectively. One of the aims of the present project is to bring specialists of the two domains together and to tackle problems where model-checking and proof-checking can be combined (the "V" and the "P" of the acronym). Applications in the realm of distributed computation, or concurrency theory (the "I" of the acronym) are particularly targeted.

7.3.3. Inria International Partners

7.3.3.1. Informal International Partners

The project-team has collaborations with University of Aarhus (Denmark), KU Leuven, 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, University of Cambridge, Universidad Nacional de Córdoba, and Universidad de Chile.

7.4. International Research Visitors

7.4.1. Visits of International Scientists

Mauro Jaskelioff (National University of Rosario and CONICET, Argentina) visited the team for a week in May 2018.

Vadim Zaliva (PhD student at CMU) visited the team for one month in July 2018 and collaborated with Matthieu Sozeau on the use of Template-Coq to verify translations from shallow to deep embeddings.

7.4.2. Internships

Yann Régis-Gianas supervised the internship of Loïc Peyrot (Master 1, Paris Diderot) about the development of a tool to define exercises for the learn-ocaml platform in a single ML file.

Yann Régis-Gianas supervised the internship of Carine Morel (Master 1, Paris Diderot) about the development of a user-friendly teaching-oriented documentation for the learn-ocaml platform.

Yann Régis-Gianas supervised the internship of Olivier Martinot (Licence 3, Paris Diderot) about the implementation of a set of efficient incrementalised combinators for list processing in cache-transfer style.

Alexis Saurin co-supervised the internship of Ikram Cherigi (Master 2 LMFI, Paris Diderot) about classical realisability and forcing in set theory.

Alexis Saurin supervised the internship of Xavier Onfroy (Master 2 LMFI, Paris Diderot) on formalisation of circular proofs in fixed-point logics and the decidability of validity.

Alexis Saurin supervised the internship of Kostia Chardonnet (Master 1 MPRI, Paris Diderot) about call-by-need calculus, degrees of lazyness and probabilistic lambda calculus.

7.4.3. Research Stays Abroad

Pierre-Louis Curien visited East China Normal University for a month from mid-October to mid-November 2018 (collaborations with Yuxin Deng and Min Zhang) as invited professor.

Pierre-Louis Curien visited the Institute of Mathematics of the Serbian Academy of Sciences in Belgrade in September 2018 for a week (collaboration with Zoran Petrić and other coauthors).

Hugo Herbelin participated to the Types, Sets and Constructions Trimester Program at the Hausdorff Research Institute of Mathematics in Bonn, May-August 2018.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Organisation

8.1.1.1. General Chair, Scientific Chair

Pierre-Louis Curien organised a Day of Hommage to the memory of Maurice Nivat on February 6, 2018, at University Paris 7.

Alexis Saurin organised and co-chaired with David Baelde the Paris workshop in Oxford, UK, July 7-8th 2018, collocated with FLoC 2018.

Matthieu Sozeau co-organised and co-chaired with Nicolas Tabareau the Coq Workshop 2018 in Oxford, UK, July 8th 2018, collocated with FLoC 2018.

8.1.1.2. Member of the Organising Committees

Yves Guiraud organised with Philippe Malbos (Univ. Lyon 1) and Samuel Mimram (École Polytechnique) the fourth edition of the workshop HDRA (Higher-Dimensional Rewriting and Algebra) in July 2018 in Oxford.

8.1.2. Scientific Events Selection

8.1.2.1. Member of the Conference Program Committees

Hugo Herbelin was a member of the program committee of the conference POPL 2019.

Yann Régis-Gianas was a member of the program committee of the conference PPDP 2018.

Yann Régis-Gianas was a member of the program committee of the conference JFLA 2019.

Matthieu Sozeau was member of the program committee of the conference Interactive Theorem Proving 2018 which took place in Oxford during FLoC 2018 and the 13th Workshop on Logical and Semantic Frameworks with Applications, which took place in Fortaleza, Brazil, September 26-28, 2018.

8.1.2.2. Member of the Conference Steering Committees

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

Hugo Herbelin is a member of the steering committee of the conference TYPES.

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

8.1.3. Journal

8.1.3.1. Member of the Editorial Boards

Pierre-Louis Curien is editor in chief of the Cambridge University Press journal Mathematical Structures in Computer Science (since January 2016).

Alexis Saurin is editing a special issue of MSCS dedicated to contributions in honour of Dale Miller for his 60th birthday.

8.1.3.2. Reviewer - Reviewing Activities

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

8.1.4. Invited Talks

Pierre-Louis Curien gave talks on the legacy of Maurice Nivat at two special events organised to honour his memory: special sessions in the Journées du GDR IM (Ecole Polytechnique, May 2018), and at ICALP 2018 (Prague, July 2018).

Eric Finster gave an invited talk at the annual meeting of the GDR Topologie in Montpellier in October entitled "The Cotopological Tower".

Hugo Herbelin gave an invited talk on computing with Gôdel's completeness theorem using side effects at the workshop Proof, Computation and Complexity in Bonn, July 2018.

Yann Régis-Gianas gave an invited talk about copatterns in OCaml at the "Logique, Types et Preuves" workshop of the GDR GPL.

Matthieu Sozeau gave an invited talk on "The Predicative, Polymorphic, Cumulative Calculus of Inductive Constructions" at the TYPES 2018 International Conference on Types for Proofs and Programs in Braga, Portugal, 18-21 June 2018.

Matthieu Sozeau gave an invited seminar entitled "Programmer en Coq" at the Collège de France, on December 12th 2018, part of Xavier Leroy's lectures on the Curry-Howard Isomorphism.

Théo Zimmermann was invited to give a talk in the First international workshop on Empirical Answers to Questions of Software Engineering to present his work on the impact of switching bug trackers [60].

8.1.5. Scientific Expertise

Pierre-Louis Curien has been an expert for a hiring committee for an assistant professor position in Logic, Computation and Programming at Stockholm University (June 2018).

Hugo Herbelin has been a reviewer for FWF (Austrian research funding agency) and NKFI (Hungarian research funding agency).

8.1.6. Research Administration

Pierre-Louis Curien is a member of the Scientific Council of the CIRM (Centre International de Rencontres Mathématiques).

Pierre-Louis Curien and Yves Guiraud are members of the scientific council of the Computer Science department of University Paris 7.

Yves Guiraud is the head of the “Preuves, Programmes et Systèmes” (PPS) pole of the IRIF laboratory (since April 2016), and a member of the IRIF direction council (since September 2017).

Yann Régis-Gianas is a member of the Executive Comitee of the OCaml Foundation, acting as a representative of the teaching community.

In collaboration with Emmanuel Chailloux (UPMC), Yann Régis-Gianas is organising the next four years of IRILL, an initiative about innovation in free software.

8.1.7. Presentation of papers

Pierre-Louis Curien gave a talk at the Conference “Topology in Australia and South Korea 2018”, Pohang (https://cgp.ibs.re.kr/conferences/Topology_in_Australia_and_South_Korea) in April 2018 (‘A syntactic approach to opetopes’).

Yann Régis-Gianas gave talks to present "Morbig", a static parser for POSIX Shellat at FOSDEM 2018 in Brussels, at MiniDebConf 2018 and at SLE 2018 in Boston.

Yann Régis-Gianas gave a talk at OCaml 2018 in St Louis to present Learn-OCaml, a project to support the teaching of OCaml worldwide.

Yann Régis-Gianas gave a talk at JFLA 2018 to present his work about extending OCaml with Copatterns.

Exequiel Rivas gave a talk on relating interfaces for computational effects at the Seventh Workshop on Mathematically Structured Functional Programming (MSFP 2018) in July 2018.

Exequiel Rivas gave a talk on relating interfaces for computational effects at the First Symposium on Compositional Structures (SYCO I) in September 2018.

Matthieu Sozeau gave a talk and presented a poster at PEPM 2018 on Equations, gave a talk on Typed Template Coq at CoqPL 2018, along with the traditional Coq developer session. These events were co-located with POPL 2018 in Los Angeles, CA in January 2018.

8.1.8. Talks in seminars

Eric Finster gave a talk about the implementation of Catt, a proof assistant for Maltsinotis-style higher categories at the Journées pi.r2 (Fontainebleau, june 2018).

Eric Finster gave a talk on “Towards Higher Universal Algebra in Dependent Type Theory” in the working group on Higher Categories, Polygraphs and Homotopy, during the Journées PPS (November 2018).

Eric Finster gave a talk during the HoTTest Electronic Seminar on "Towards Higher Universal Algebra in Type Theory".

Eric Finster gave a talk entitled "Left Exact Modalities in Type Theory" at the Cambridge Logic and Semantics Seminar (March 2018).

Hugo Herbelin gave a talk on computing with Gödel’s completeness theorem at the seminar of the Logic team of the IMJ-PRG Paris 6 - Paris 7 lab.

Pierre Letouzey gave two talks on “Un problème d’Hofstadter pour ses lecteurs curieux” during the Journées pi.r2 (Fontainebleau, june 2018) and the Journées PPS (November 2018).

Jean-Jacques Lévy gave a talk at the IRIF Verification seminar (January 8) entitled "Proofs of graph algorithms with automation and their readability".

Jean-Jacques Lévy gave a talk at the VIP ISCAS-Inria workshop (Irif, November 19-22) entitled "Comparing a Formal Proof in Why3, Coq and Isabelle".

Yann Régis-Gianas gave a talk at Gallium seminar to present "Morbig", a static parser for POSIX Shell.

Exequiel Rivas gave a talk on “Arrows: from programming to semantics” at the Laboratoire d’Informatique de Paris Nord (LIPN), September 2018.

Exequiel Rivas gave a talk on “Interaction from monadic interfaces” during the Journées PPS, November 2018.

Exequiel Rivas gave a talk on “Interaction from monadic interfaces” at the Prosecco seminar, December 2018.

Alexis Saurin gave a talk at I2M seminar in the "logique de la programmation" group entitled "logical-by-need".

Alexis Saurin gave a talk at the VIP ISCAS-Inria workshop (Irif, November 19-22) entitled "On non-wellfounded proofs and cuts in linear logic with fixed points."

Matthieu Sozeau gave a talk on the MetaCoq Project at the VALS seminar, LRI, October 2018.

Matthieu Sozeau gave a talk on “A universe of strict propositions“ during the Journées PPS, November 2018.

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

Hugo Herbelin attended TYPES 2018 in Porto (June), the Coq Implementors Workskop in Nice (May 2018), FLoC in Oxford (July 2018), the GPL working group in Marseille (October 2018), the Scalp working group in Saclay (November 2018).

Hugo Herbelin gave a talk on cubical type theory at the workshop Types, Homotopy Type theory and Verification (June 2018), on computing with Markov’s principle at the workshop Proof and Computation (July 2018), on a constructive proof of the axiom of dependent choice compatible with classical logic at the workshop Constructive Mathematics (August 2018), all workshops of the special trimester on types, sets and constructions in Bonn. He gave a talk on the reverse mathematics of Gödel’s completeness theorem and on the computational contents of Henkin’s proof at the seminar of the trimester.

Hugo Herbelin gave a talk on cubical type theory at the TYPES conference (June 2018).

Hugo Herbelin gave a talk on the cubical type structure of cubical type theory at the HoTT-UF workshop in Oxford, July 2018.

Jean-Jacques Lévy attended the Coq Winter School 2018-2019 (SSReflect & MathComp) at Inria Sophia-Antipolis (November 2018).

Yann Régis-Gianas attended JFLA 2018, OCaml 2018, SPLASH 2018, FOSDEM 2018 and MiniDebConf 2018.

Exequiel Rivas attended to MSFP 2018 and SYCO I.

Alexis Saurin attended FLoC 2018 in Oxford.

Matthieu Sozeau attended POPL in Los Angeles, CA (January 2018), the Coq Implementors Workshop in Nice (May 2018), the TYPES Conference in Braga, Portugal (June 2018), FLoC in Oxford (July 2018) and ICFP in St Louis, MI (September 2018).

Théo Zimmermann attented FOSDEM in Brussels (February 2018), the Coq Implementors Workskop in Nice (May 2018), FLoC in Oxford (July 2018), OpenSym in Paris (August 2018) and the EAQSE workshop in Villebrumier (November 2018).

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 Matthieu Sozeau. The speakers in 2018 were Rodolphe Lepigre (Practical Curry-Style using Choice Operators, Local Subtyping and Circular proofs), Armaël Guéneau (A Fistful of Dollars: Formalising Asymptotic Complexity Claims via Deductive Program Verification), Jérôme Siméon (Specifying and compiling domain specific languages using Coq: Three case studies), Laura Fontanella ’Axiom of choice in classical realisabiity), Adrien Guatto (A Generalised Modality for Recursion), Hadrien Batmalle (Preservation of properties of the original model in classical realisability), Raphaël Cauderlier (Tactics and certificates in Meta Dedukti).

8.1.11. Groupe de travail Catégories supérieures, polygraphes et homotopie

Several members of the team participate actively in this weekly working group of PPS, organised by François Métayer (Univ. Nanterre) 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). Pierre-Louis Curien taught a course on the Foundations of Programming Languages at East China Normal University (12 hours, November 2018).

Master: Hugo Herbelin teaches with Nicolas Tabareau the course on Homotopy Type Theory at the LMFI.

Master: Pierre Letouzey teaches two short courses to the LMFI Master 2 students : “Programming in Coq” and “Introduction to computed-aided formal proofs”. These two courses come in addition to Pierre Letouzey’s regular duty as teacher in the Computer Science department of Paris 7 (including a course on Compilation to M2-Pro students and a course on computed-aided formal proofs to M1 students).

Master: 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.

Master: Alexis Saurin taught the proof theory and lambda-calculus part of the cours fondamental de logique in M2 “Logique Mathématique et Fondements de l’Informatique”, Université Paris 7.

Alexis Saurin chairs LMFI M2 since September 2013.

Master: Matthieu Sozeau taught the MPRI course on Advanced uses of proof assistants (12 hours + a project), together with Bruno Barras (Inria Deducteam).

Matthieu Sozeau gave a guest lecture on dependent pattern-matching and Equations at the University of Saarland in April 2018.

Matthieu Sozeau gave an introductory lecture on Dependent Type Theory at the EUTYPES summer school in Ohrid, Macedonia, in August 2018.

8.2.2. Supervision

Guillaume Claret defended his PhD on “Programmation avec effets en Coq” on 18 September 2018 (supervised by Hugo Herbelin and Yann Régis-Gianas). Note that the dissertation was completed in 2015 but Guillaume Claret moved in the meantime to a private company and the defense has been delayed.

PhD (completed): Thibaut Girka defended his PhD on "Differential Program Semantics" on the 3rd of July 2018, supervised by Roberto Di Cosmo and Yann Régis-Gianas.

PhD (abandoned): Cyprien Mangin, Dependent Pattern-Matching, induction-induction and higher inductive types (started in September 2015), supervised by Matthieu Sozeau and Bruno Barras. Cyprien Mangin left for industry.

PhD in progress: Théo Zimmermann (started in September 2016), supervised by Hugo Herbelin.

PhD in progress: Cédric Ho Thanh (started in September 2017), on Opetopes for higher-dimensional rewriting and koszulity, supervised by Pierre-Louis Curien and Samuel Mimram.

PhD in progress: Antoine Allioux (started in February 2018), on the formalisation of algebraic structures in type theory, supervised by Yves Guiraud and Matthieu Sozeau.

PhD in progress: Abhishek De (started in october 2018), on fixed point logics, structures for infinite proofs and their finite representations, supervised by Alexis Saurin.

The following are cosupervisions of PhD students who are not formally part of the team:

PhD in progress: Rémi Nollet, Functional reactive programming and temporal logics: their syntax and semantics - from discrete to continuous time (started in September 2016), supervised by Alexis Saurin and Christine Tasson.

PhD in progress: Gaëtan Gilbert (at Inria Nantes), Definitional proof-irrelevance in the Calculus of Inductive Constructions (started in September 2016), supervised by Nicolas Tabareau and Matthieu Sozeau.

PhD in progress: Simon Forest (at École Polytechnique), Rewriting in semistrict higher categories (started in September 2017), supervised by Yves Guiraud and Samuel Mimram.

PhD in progress: Théo Winterhalter (at Inria Nantes), Extensional to Intensional type theory and meta-theory of proof-irrelevance (started in September 2017), supervised by Nicolas Tabareau and Matthieu Sozeau.

8.2.3. Juries

Pierre-Louis Curien was member of the jury of the PhD thesis of Clovis Eberhard (Université Savoie Mont Blanc), defended in June 2018.

Hugo Herbelin was a member of the jury of the PhD thesis of Andrea Vezzosi (University of Göteborg, Sweden), defended in September 2018.

Hugo Herbelin was a member of the jury of the PhD thesis of Guillaume Claret (University of Paris-Diderot), defended in September 2018.

Hugo Herbelin was referee for the PhD thesis of Simon Boulier (University of Nantes), defended in November 2018.

Hugo Herbelin was president of the jury of the PhD thesis of Pierre Cagne (University of Paris Diderot), defended in December 2018.

Yann Régis-Gianas is a member of the jury of the competitive examination for the entrance to the Ecoles Normales Supérieures and the Ecole Polytechnique.

Matthieu Sozeau was member of the jury of the PhD thesis of Amin Timany (KU Leuven, Belgium), defended in April 2018.

8.3. Popularisation

Pierre-Louis Curien gave a talk in the Lycée Georges Dumézil (Vernon, Eure, May 2018) on computer bugs and their prevention, on the occasion of the 50th anniversary of this high school.

Jean-Jacques Lévy is member of the Inria-Alumni's executive committee (4 meetings in 2018) and organised the session about the Transparency of Algorithms (November 12).

Jean-Jacques Lévy was invited by the French Academy of Sciences to participate to the 2018 Hangzhou International Human Resources Exchanges and Cooperation Conference (Hangzhou, November 9-12).

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

8.3.1. Education

Yann Régis-Gianas is the project leader of the "Learn-OCaml" project whose purpose is to support teaching the OCaml programming language worldwide.

8.3.2. Internal action

- Science outreach towards services (DPEI, STIP...)

  Jean-Jacques Lévy talked about "L'informatique en 4 temps" at the Alumni-UniThé seminar at Inria Bordeaux (October 10).

9. Bibliography

Major publications by the team in recent years

[1] R. M. AMADIO, Y. REGIS-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] D. BAELDE, A. DOUMANE, A. SAURIN. *Infinitary proof theory : the multiplicative additive case* , in "Proceedings of CSL 2016", September 2016, https://hal.archives-ouvertes.fr/hal-01339037

[4] C. CHENAVIER. *The lattice of reduction operators: applications to noncommutative Gröbner bases and homological algebra*, Université paris Diderot, December 2016, https://tel.archives-ouvertes.fr/tel-01415910

[5] 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

[6] 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

[7] 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/

[8] 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

[9] P. DEHORNOY, Y. GUIRAUD. *Quadratic normalization in monoids*, in "Internat. J. Algebra Comput.", 2016, vol. 26, n^{o} 5, pp. 935–972, https://doi.org/10.1142/S0218196716500399

[10] 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

[11] G. GILBERT, J. COCKX, M. SOZEAU, N. TABAREAU. *Definitional Proof-Irrelevance without K*, in "46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2019", Lisbon, Portugal, POPL, January 2019, https://hal.inria.fr/hal-01859964

[12] T. GIRKA, D. MENTRÉ, Y. REGIS-GIANAS. *Oracle-based Dierential Operational Semantics (long version)*, Université Paris Diderot / Sorbonne Paris Cité, October 2016, https://hal.inria.fr/hal-01419860

[13] Y. GUIRAUD. *Rewriting methods in homotopical and higher-dimensional algebra*, Univ. Paris 7, 2019, Habilitation à diriger des recherches

[14] 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

[15] 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

[16] H. HERBELIN. *On the Degeneracy of Sigma-Types in Presence of Computational Classical Logic*, in "Proceedings of TLCA 2005", P. URZYCZYN (editor), Lecture Notes in Computer Science, Springer, 2005, vol. 3461, pp. 209–220

[17] 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/

[18] 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

[19] 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

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

[21] 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

[22] T. U. F. PROGRAM. *Homotopy type theory—univalent foundations of mathematics*, The Univalent Foundations Program, Princeton, NJ; Institute for Advanced Study (IAS), Princeton, NJ, 2013, xiv+589 p. , http://homotopytypetheory.org/book

[23] Y. REGIS-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

[24] 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

[25] B. ZILIANI, M. SOZEAU. *A comprehensible guide to a new unifier for CIC including universe polymorphism and overloading*, in "Journal of Functional Programming", 2017, vol. 27 [*DOI :* 10.1017/S0956796817000028], https://hal.inria.fr/hal-01671925

Publications of the year

Doctoral Dissertations and Habilitation Theses

[26] G. CLARET. *Program in Coq*, Université Paris Diderot - Paris 7, September 2018, https://hal.inria.fr/tel-01890983

[27] T. GIRKA. *Differential program semantics*, Université Paris Diderot, July 2018, https://hal.inria.fr/tel-01890508

Articles in International Peer-Reviewed Journals

[28] M. ANEL, G. BIEDERMANN, E. FINSTER, A. JOYAL. *Goodwillie's calculus of functors and higher topos theory*, in "Journal of topology", December 2018, vol. 11, n° 4, pp. 1100-1132 [*DOI :* 10.1112/TOPO.12082], https://hal.inria.fr/hal-01939906

[29] F. FAGES, T. MARTINEZ, D. A. ROSENBLUETH, S. SOLIMAN. *Influence Networks compared with Reaction Networks: Semantics, Expressivity and Attractors*, in "IEEE/ACM Transactions on Computational Biology and Bioinformatics", 2018, vol. PP, n° 99, pp. 1-14 [*DOI :* 10.1109/TCBB.2018.2805686], https://hal.inria.fr/hal-01510216

[30] G. GILBERT, J. COCKX, M. SOZEAU, N. TABAREAU. *Definitional Proof-Irrelevance without K*, in "Proceedings of the ACM on Programming Languages", January 2019, pp. 1-28 [*DOI :* 10.1145/329031610.1145/3290316], https://hal.inria.fr/hal-01859964

[31] Y. GUIRAUD, E. HOFFBECK, P. MALBOS. *Convergent presentations and polygraphic resolutions of associative algebras*, in "Mathematische Zeitschrift", 2018, 68 pages, https://hal.archives-ouvertes.fr/hal-01006220

[32] Y. GUIRAUD, P. MALBOS. *Polygraphs of finite derivation type*, in "Mathematical Structures in Computer Science", 2018, vol. 28, n° 2, pp. 155-201, https://arxiv.org/abs/1402.2587 [*DOI :* 10.1017/S0960129516000220], https://hal.archives-ouvertes.fr/hal-00932845

[33] Y. JIANG, J. LIU, G. DOWEK, K. JI. *Towards Combining Model Checking and Proof Checking*, in "The Computer Journal", 2019, https://hal.inria.fr/hal-01970274

[34] J.-O. KAISER, B. ZILIANI, R. KREBBERS, Y. RÉGIS-GIANAS, D. DREYER. *Mtac2: typed tactics for backward reasoning in Coq*, in "Proceedings of the ACM on Programming Languages", July 2018, vol. 2, n° ICFP, pp. 1 - 31 [*DOI :* 10.1145/3236773], https://hal.inria.fr/hal-01890511

[35] L. PATEY, K. YOKOYAMA. *The proof-theoretic strength of Ramsey's theorem for pairs and two colors*, in "Advances in Mathematics", May 2018, vol. 330, pp. 1034 - 1070 [*DOI :* 10.1016/J.AIM.2018.03.035], https://hal.archives-ouvertes.fr/hal-01888655

[36] N. TABAREAU, É. TANTER, M. SOZEAU. *Equivalences for Free: Univalent Parametricity for Effective Transport*, in "Proceedings of the ACM on Programming Languages", September 2018, pp. 1-29 [*DOI :* 10.1145/3234615], https://hal.inria.fr/hal-01559073

International Conferences with Proceedings

[37] A. ANAND, S. BOULIER, C. COHEN, M. SOZEAU, N. TABAREAU. *Towards Certified Meta-Programming with Typed Template-Coq*, in "ITP 2018 - 9th Conference on Interactive Theorem Proving", Oxford, United Kingdom, LNCS, Springer, July 2018, vol. 10895, pp. 20-39 [*DOI :* 10.1007/978-3-319-94821-8_2], https://hal.archives-ouvertes.fr/hal-01809681

[38] T. LETAN, Y. RÉGIS-GIANAS, P. CHIFFLIER, G. HIET. *Modular Verification of Programs with Effects and Effect Handlers in Coq*, in "FM 2018 - 22nd International Symposium on Formal Methods", Oxford, United Kingdom, LNCS, Springer, July 2018, vol. 10951, pp. 338-354 [*DOI :* 10.1007/978-3-319-95582-7_20], https://hal.inria.fr/hal-01799712

[39] É. MIQUEY, H. HERBELIN. *Realizability Interpretation and Normalization of Typed Call-by-Need λ-calculus With Control*, in "FOSSACS 18 - 21st International Conference on Foundations of Software Science and Computation Structures", Thessalonique, Greece, C. BAIER, U. D. LAGO (editors), LNCS, Springer, April 2018, vol. 10803, pp. 276-292, https://arxiv.org/abs/1803.00914 [*DOI :* 10.1007/978-3-319-89366-2_15], https://hal.inria.fr/hal-01624839

[40] E. RIVAS. *Relating Idioms, Arrows and Monads from Monoidal Adjunctions*, in "Seventh Workshop on Mathematically Structured Functional Programming- EPTCS", Oxford, United Kingdom, July 2018, vol. 275, pp. 18-33, https://hal.inria.fr/hal-01946996

[41] Y. RÉGIS-GIANAS, N. JEANNEROD, R. TREINEN. *Morbig: A Static Parser for POSIX Shell*, in "SLE 2018 - ACM SIGPLAN International Conference on Software Language Engineering", Boston, United States, November 2018 [*DOI :* 10.1145/3276604.3276615], https://hal.archives-ouvertes.fr/hal-01890044

[42] A. TIMANY, M. SOZEAU. *Cumulative Inductive Types in Coq*, in "FSCD 2018 - 3rd International Conference on Formal Structures for Computation and Deduction", Oxford, United Kingdom, July 2018 [*DOI :* 10.4230/LIPICS.FSCD.2018.29], https://hal.inria.fr/hal-01952037

[43] T. WINTERHALTER, M. SOZEAU, N. TABAREAU. *Eliminating Reflection from Type Theory: To the Legacy of Martin Hofmann*, in "CPP 2019 - The 8th ACM SIGPLAN International Conference on Certified Programs and Proofs", Lisbonne, Portugal, ACM, January 2019, pp. 91-103 [*DOI :* 10.1145/3293880.3294095], https://hal.archives-ouvertes.fr/hal-01849166

National Conferences with Proceedings

[44] P. LAFORGUE, Y. RÉGIS-GIANAS. *OCaml étendu avec du filtrage par comotifs*, in "JFLA 2018 - Journées Francophones des Langages Applicatifs", Banyuls sur mer, France, January 2018, https://hal.inria.fr/hal-01897456

Conferences without Proceedings

[45] A. ANAND, S. BOULIER, N. TABAREAU, M. SOZEAU. *Typed Template Coq – Certified Meta-Programming in Coq*, in "CoqPL 2018 - The Fourth International Workshop on Coq for Programming Languages", Los Angeles, CA, United States, January 2018, pp. 1-2, https://hal.inria.fr/hal-01671948

[46] R. NOLLET, A. SAURIN, C. TASSON. *Local validity for circular proofs in linear logic with fixed points: extended version*, in "Computer Science Logic", Birmingham, United Kingdom, September 2018, https://hal.archives-ouvertes.fr/hal-01825477

[47] T. ZIMMERMANN. *Challenges in the collaborative development of a complex mathematical software and its ecosystem*, in "OpenSym 2018 - 14th International Symposium on Open Collaboration", Paris, France, August 2018, vol. 2018 [*DOI :* 10.1145/3233391.3233966], https://hal.inria.fr/hal-01951322

Scientific Books (or Scientific Book chapters)

[48] A. BOUAJJANI, C. ENEA, K. JI, S. QADEER. *On the Completeness of Verifying Message Passing Programs Under Bounded Asynchrony*, in "International Conference on Computer Aided Verification, CAV 2018: Computer Aided Verification", Springer International Publishing, July 2018, pp. 372-391, https://hal.archives-ouvertes.fr/hal-01947855

Other Publications

[49] R. CHEN, C. COHEN, J.-J. LEVY, S. MERZ, L. THERY. *Formal Proofs of Tarjan's Algorithm in Why3, Coq, and Isabelle*, October 2018, https://arxiv.org/abs/1810.11979 - working paper or preprint, https://hal.inria.fr/hal-01906155

[50] C. CHENAVIER. *A Lattice Formulation of the F 4 Completion Procedure*, January 2018, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01489200

[51] C. CHENAVIER. *Syzygies among reduction operators*, April 2018, https://arxiv.org/abs/1708.08709 - working paper or preprint, https://hal.archives-ouvertes.fr/hal-01578555

[52] T. COQ DEVELOPMENT TEAM. *The Coq Proof Assistant, version 8.8.0*, April 2018, Software [*DOI :* 10.5281/ZENODO.1219885], https://hal.inria.fr/hal-01954564

[53] P.-L. CURIEN, J. OBRADOVIC. *Categorified cyclic operads*, January 2018, https://arxiv.org/abs/1706.06788 - working paper or preprint, https://hal.archives-ouvertes.fr/hal-01679682

[54] T. GIRKA. *correlating_program*, July 2018, https://archive.softwareheritage.org/swh:1:rev:c8fca417ee9eefe25683042192da6747
Software, https://hal.inria.fr/hal-01831364

[55] T. GIRKA, Y. RÉGIS-GIANAS. *Correlating Oracles*, July 2018, https://archive.softwareheritage.org/swh:1:rev:cccf789c12617208
Software, https://hal.inria.fr/hal-01831369

[56] H. HERBELIN, É. MIQUEY. *Continuation-and-environment-passing style translations: a focus on call-by-need*, January 2019, working paper or preprint, https://hal.inria.fr/hal-01972846

[57] C. HO THANH. *The equivalence between many-to-one polygraphs and opetopic sets*, July 2018, https://arxiv.org/abs/1806.08645 - working paper or preprint, https://hal.archives-ouvertes.fr/hal-01946918

[58] C. MANGIN, M. SOZEAU. *Equations reloaded*, July 2018, working paper or preprint, https://hal.inria.fr/hal-01671777

[59] Y. REGIS-GIANAS, N. JEANNEROD, R. TREINEN. *Morbig*, October 2018, https://archive.softwareheritage.org/swh:1:dir:eb7770e
Software, https://hal.inria.fr/hal-01897572

[60] T. ZIMMERMANN, A. CASANUEVA ARTÍS. *Impact of switching bug trackers: a case study*, December 2018, working paper or preprint, https://hal.inria.fr/hal-01951176

References in notes

[61] T. ALTENKIRCH. *Extensional Equality in Intensional Type Theory*, in "14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999", IEEE Computer Society, 1999, pp. 412–420, https://doi.org/10.1109/LICS.1999.782636

[62] T. ALTENKIRCH, P. CAPRIOTTI, N. KRAUS. *Extending Homotopy Type Theory with Strict Equality*, in "CSL", LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, vol. 62, pp. 21:1–21:17

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

[64] D. ARA, F. MÉTAYER. *The Brown-Golasiński Model Structure on strict ∞-groupoids revisited*, in "Homology, Homotopy and Applications", 2011, vol. 13, n^{o} 1, pp. 121–142

[65] J. BAEZ, A. CRANS. *Higher-dimensional algebra. VI. Lie 2-algebras*, in "Theory Appl. Categ.", 2004, vol. 12, pp. 492–538

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

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

[68] G. BONFANTE, Y. GUIRAUD. *Polygraphic Programs and Polynomial-Time Functions*, in "Logical Methods in Computer Science", 2009, vol. 5, n^{o} 2, pp. 1–37

[69] S. BOULIER, P.-M. PÉDROT, N. TABAREAU. *The next 700 syntactical models of type theory*, in "Certified Programs and Proofs (CPP 2017)", Paris, France, January 2017, pp. 182 - 194 [*DOI :* 10.1145/3018610.3018620], https://hal.inria.fr/hal-01445835

[70] B. BUCHBERGER. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)*, Mathematical Institute, University of Innsbruck, Austria, 1965

[71] A. BURRONI. *Higher-dimensional word problems with applications to equational logic*, in "Theoretical Computer Science", jul 1993, vol. 115, n^{o} 1, pp. 43–62

[72] A. CHLIPALA. *Certified Programming with Dependent Types - A Pragmatic Introduction to the Coq Proof Assistant*, MIT Press, 2013, http://mitpress.mit.edu/books/certified-programming-dependent-types

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

[74] J. COCKX. *Dependent Pattern Matching and Proof-Relevant Unification*, Katholieke Universiteit Leuven, Belgium, 2017, https://lirias.kuleuven.be/handle/123456789/583556

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

[76] 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

[77] 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

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

[79] P. DEHORNOY, Y. LAFONT. *Homology of Gaussian groups*, in "Ann. Inst. Fourier (Grenoble)", 2003, vol. 53, n^{o} 2, pp. 489–540, http://aif.cedram.org/item?id=AIF_2003__53_2_489_0

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

[81] 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

[82] 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

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

[84] 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, pp. 63-92

[85] 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

[86] Y. GUIRAUD. *Présentations d'opérades et systèmes de réécriture*, Univ. Montpellier 2, 2004

[87] Y. GUIRAUD. *Termination Orders for 3-Dimensional Rewriting*, in "Journal of Pure and Applied Algebra", 2006, vol. 207, n^{o} 2, pp. 341–371

[88] Y. GUIRAUD. *The Three Dimensions of Proofs*, in "Annals of Pure and Applied Logic", 2006, vol. 141, n^{o} 1–2, pp. 266–295

[89] Y. GUIRAUD. *Two Polygraphic Presentations of Petri Nets*, in "Theoretical Computer Science", 2006, vol. 360, n^{o} 1–3, pp. 124–146

[90] Y. GUIRAUD, E. HOFFBECK, P. MALBOS. *Confluence of linear rewriting and homology of algebras*, in "3rd International Workshop on Confluence", Vienna, Austria, July 2014, https://hal.archives-ouvertes.fr/hal-01105087

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

[92] Y. GUIRAUD, P. MALBOS. *Identities among relations for higher-dimensional rewriting systems*, in "Séminaires et Congrès, Société Mathématique de France", 2011, vol. 26, pp. 145-161

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

[94] M. HOFMANN, T. STREICHER. *The groupoid interpretation of type theory*, in "Twenty-five years of constructive type theory (Venice, 1995)", Oxford Logic Guides, Oxford Univ. Press, New York, 1998, vol. 36, pp. 83–111

[95] 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

[96] J. KOCK, A. JOYAL, M. BATANIN, J.-F. MASCARI. *Polynomial functors and opetopes*, in "Advances in Mathematics", 2010, vol. 224, n^{o} 6, pp. 2690 - 2737 [*DOI :* 10.1016/J.AIM.2010.02.012], http://www.sciencedirect.com/science/article/pii/S0001870810000769

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

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

[99] Y. LAFONT. *Towards an Algebraic Theory of Boolean Circuits*, in "Journal of Pure and Applied Algebra", 2003, vol. 184, pp. 257-310

[100] Y. LAFONT, F. MÉTAYER, K. WORYTKIEWICZ. *A Folk Model Structure on Omega-Cat*, in "Advances in Mathematics", 2010, vol. 224, n^{o} 3, pp. 1183–1231

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

[102] 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

[103] J. LURIE. *Higher topos theory*, Annals of Mathematics Studies, Princeton University Press, Princeton, NJ, 2009, vol. 170, xviii+925 p.

[104] P. MALBOS. *Critères de finitude homologique pour la non convergence des systèmes de réécriture de termes*, Univ. Montpellier 2, 2004

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

[106] 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

[107] S. B. PRIDDY. *Koszul resolutions*, in "Trans. Amer. Math. Soc.", 1970, vol. 152, pp. 39–60

[108] 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

[109] 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

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

[111] C. C. SQUIER. *Word problems and a homological finiteness condition for monoids*, in "J. Pure Appl. Algebra", 1987, vol. 49, n^{o} 1-2, pp. 201–217

[112] R. STREET. *Limits Indexed by Category-Valued 2-Functors*, in "Journal of Pure and Applied Algebra", 1976, vol. 8, pp. 149–181

[113] T. C. D. TEAM. *The Coq Proof Assistant, version 8.7.1*, December 2017, https://doi.org/10.5281/zenodo.1133970

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