IN PARTNERSHIP WITH:
CNRS

Université Denis Diderot (Paris 7)

Activity Report 2017

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
A2.1.3. - 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. Personnel

Research Scientists

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

Faculty Members

Thierry Coquand [University of Gothenburg, Professor, Inria International Chair]
Pierre Letouzey [Univ Paris Diderot (Paris 7), Associate Professor]
Yann Régis-Gianas [Univ Paris Diderot, Associate Professor]

Post-Doctoral Fellows

Cyrille Chenavier [ATER Univ Paris Diderot, until Aug 2017]
Eric Finster [ERC CoqHoTT, from Jan 2017]
Kailiang Ji [Inria, from Dec 2017]
Exequiel Rivas Gadda [Inria, from Dec 2017]

PhD Students

Amina Doumane [Univ Paris Diderot, until June 2017]
Étienne Miquey [Univ Paris Diderot, until Nov 2017]
Jovana Obradović [Univ Paris Diderot, until Aug 2017]
Maxime Lucas [Univ Paris Diderot, until Dec 2017]
Thibaut Girka [Mitsubishi]
Guillaume Claret [Univ Paris Diderot]
Cyprien Mangin [Univ Paris Diderot]
Théo Zimmermann [Univ Paris Diderot]
Cédric Ho Thanh [Univ Paris Diderot, since Sep 2017]

Technical staff

Daniel de Rauglaudre [Inria]
Thierry Martinez [Inria]

Interns

Kostia Chardonnet [Inria, from Jun 2017 until Sep 2017]
Charlotte Barot [Intern at IRIF, from May 2017 until Sep 2017]
Théo Winterhalter [Inria, from March 2017 until Sep 2017]

Administrative Assistants

Lindsay Polienor [Inria]
Sandrine Vergès [Inria]

External Collaborators

Arnaud Spiwack [Tweag I/0, engineer]
Samuel Mimram [Ecole Polytechnique, associate professor]
Claudia Faggian [IRIF, CNRS, Researcher]
Philippe Malbos [Université Lyon I, associate professor]
Nicolas Tabareau [Inria Nantes, Researcher]

2. Overall Objectives

2.1. Overall Objectives

The research conducted in πr^2 is devoted both to the study of foundational aspects of formal proofs and programs and to the development of the Coq proof assistant software, with a focus on the dependently typed programming language aspects of Coq. The team acts as one of the strongest teams involved in the development of Coq as it hosts in particular the current coordinator of the Coq development team.

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

3. Research Program

3.1. Proof theory and the Curry-Howard correspondence

3.1.1. Proofs as programs

Proof theory is the branch of logic devoted to the study of the structure of proofs. An essential contributor to this field is Gentzen [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 [77], 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 [74], [75] 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 [72], 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 [62].

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 [104].

In 1985, Coquand and Huet [74], [75] in the Formel team of Inria-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [84], [108]. 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 [76] 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 five 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 by employees of Inria, the CNAM and Paris 7.

In the last five 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 and wiki), made its development process public, starting to use public pull-requests to track the work of developers, organizing yearly hackatons/coding-sprints for the dissemination of expertise and developers & users meetings like the Coq Workshop and CoqPL, and, perhaps more annectodicaly, retransmitting Coq working groups on a public youtube channel.

This move was made possible thanks to the hiring of Maxime Dénès in 2016 as an Inria research engineer (in Sophia-Antipolis), and the work of Matej Košík (1-year research engineer) whose work involved making the development process more predictable, streamlined and to provide a higher level of quality to the whole system, while relieving researchers from some time-consuming software development tasks. Maxime Dénès is also working in collaboration with Yves Bertot to develop the Coq consortium, which aims to become the incarnation of the global Coq community and 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), Deducteam (Saclay) 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 hosts 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 version Coq 8.7.1 had 46 contributors (counted from the start of 8.7 development), while 8.6 had 38 contributors.

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 [64], the more advanced Certified Programming with Dependent Types book by Chlipala [71] 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 now 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 [101] and Reynolds [107] 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 [105], 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 [100] and Krivine's abstract machine for call-by-name evaluation [97], [96]. 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 [61]) 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 [79]).

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 ressemblance 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 [70], 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 [98], [103], [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 [65].

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 [59] and Squier [110], [109] 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 [12], [13], [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 [12] and algebras [48]. 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 [99], [60]). 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][48].

4. Highlights of the Year

4.1. Highlights of the Year

4.1.1. Awards

Amina Doumane was awarded the The Kleene Award for Best Student Paper at the LICS 2017 conference, for her work on "Constructive Completeness for the Linear-Time mu-Calculus". She also received in January 2018 the prize of the Journal La Recherche for the same paper.

Amina Doumane was awarded the Gilles Kahn 2017 prize for her PhD thesis entitled "On the infinitary proof theory of logics with fixed points" supervised by Alexis Saurin, David Baelde and Pierre-Louis Curien.

Ludovic Patey was awarded the Prix Thiessé de Rosemont / Demassieux 2017 for his PhD thesis "Les mathématiques à rebours de théorèmes de type Ramsey", supervised by Laurent Bienvenu and Hugo Herbelin.

BEST PAPERS AWARDS:

[38]
A. DOUMANE. *Constructive completeness for the linear-time μ-calculus*, in "Conference on Logic in Computer Science 2017", Reykjavik, Iceland, June 2017, https://hal.archives-ouvertes.fr/hal-01430737

[23]

A. DOUMANE. *On the infinitary proof theory of logics with fixed points*, Université Paris 7 - Diderot, June 2017, https://hal.archives-ouvertes.fr/tel-01676953

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: Version 8.7 features a large amount of work on cleaning and speeding up the code base, notably the work of Pierre-Marie Pédrot on making the tactic-level system insensitive to existential variable expansion, providing a safer API to plugin writers and making the code more robust.

New tactics: Variants of tactics supporting existential variables "eassert", "eenough", etc. by Hugo Herbelin. Tactics "extensionality in H" and "inversion_sigma" by Jason Gross, "specialize with" accepting partial bindings by Pierre Courtieu.

Cumulative Polymorphic Inductive Types, allowing cumulativity of universes to go through applied inductive types, by Amin Timany and Matthieu Sozeau.

The SSReflect plugin by Georges Gonthier, Assia Mahboubi and Enrico Tassi was integrated (with its documentation in the reference manual) by Maxime Dénès, Assia Mahboubi and Enrico Tassi.

The "coq_makefile" tool was completely redesigned to improve its maintainability and the extensibility of generated Makefiles, and to make "_CoqProject" files more palatable to IDEs by Enrico Tassi.

A lot of other changes are described in the CHANGES file.
NEWS OF THE YEAR: Version 8.7 was released in October 2017 and version 8.7.1 in December 2017, development started in January 2017. This is the second release of Coq developed on a time-based development cycle. Its development spanned 9 months from the release of Coq 8.6 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 scientific advance in this version is the integration of cumulative inductive types in the system. More practical advances in stability, performance, usability and expressivity of tactics were also implemented, resulting in a mostly backwards-compatible but appreciably faster and more robust release. Much work on plugin extensions to Coq by the same development team has also been going on in parallel, including work on JSCoq by Emilio JG Arias, Ltac 2 by P.M-Pédrot, which required synchronised changes of the main codebase. In 2017, the construction of the Coq Consortium by Yves Bertot and Maxime Dénès has greatly advanced and is now nearing its completion.

- 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 8.7.1
- 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/

6. New Results

6.1. Effects in proof theory and programming

Participants: Hugo Herbelin, Étienne Miquey, Yann Régis-Gianas, Alexis Saurin.

6.1.1. A classical sequent calculus with dependent types

Dependent types are a key feature of type systems, typically used in the context of both richly-typed programming languages and proof assistants. Control operators, which are connected with classical logic along the proof-as-program correspondence, are known to misbehave in the presence of dependent types [14], unless dependencies are restricted to values. As a step in his work to develop a sequent-calculus version of Hugo Herbelin's dPA_ω system [16], Étienne Miquey proposed a sequent calculus with classical logic and dependent types. His calculus—named dL—is an extension of the $\mu\widetilde{\mu}$-calculus with a syntactical restriction of dependent types to the fragment of *negative-elimination free* proofs. The corresponding type system includes a list of explicit dependencies, which maintains type safety. He showed that a continuation-passing style translation can be derived by adding delimited continuations, and how a chain of dependencies can be related to a manipulation of the return type of these continuations. This work has been presented at ESOP 2017 [40].

6.1.2. Normalisation and realisability interpretation of call-by-need with control

The call-by-need evaluation strategy is an evaluation strategy of the λ-calculus which evaluates arguments of functions only when needed, and, when needed, shares their evaluations across all places where the argument is needed. The call- by-need evaluation is for instance at the heart of a functional programming language such as Haskell. A continuation-passing-style semantics for call-by-need, de facto giving a semantics to control operators, was proposed in the 90s by Okasaki, Lee and Tarditi. However, this semantics does not ensure normalisation of simply-typed call-by-need evaluation, thus failing to ensure a property which holds in the simply-typed call-by-name and call-by-value cases. Étienne Miquey and Hugo Herbelin have been considering a call-by-need λ-calculus due to Ariola et al. for which they proved the normalisation by means of a realisability interpretation. Incidentally, the variant of realisability they proposed allows to define realisers as pairs of a term and a substitution. This paves the way to give interpretation of calculus with global and mutable memory. This work has been accepted for publication at the FOSSACS 2018 conference.

6.1.3. A sequent calculus with dependent types for classical arithmetic

In 2012, Hugo Herbelin showed that classical arithmetic in finite types extended with strong elimination of existential quantification proves the axiom of dependent choice. Getting classical logic and choice together without being inconsistent is made possible by: (1) constraining strong elimination of existential quantification to proofs that are essentially intuitionistic; (2) turning countable universal quantification into an infinite conjunction of classical proofs, which are evaluated along a call-by-need evaluation strategy, so as to extract from them intuitionistic contents that complies to the intuitionistic constraint put on strong elimination of existential quantification.

Relying on its sequent calculus with dependent types and its realisability interpretation for call-by-need with control, Étienne Miquey proposed in his thesis a sequent calculus with the same computational features [25]. His calculus therefore also allows for the direct definition of proof terms for the axioms of countable and dependent choices. The proofs of normalisation and soundness are made through a realisability interpretation of the calculus, which is obtained by using Danvy's methodology of semantics artifacts.

6.1.4. Reverse mathematics of Gödel's completeness theorem

Charlotte Barot, under the supervision of Hugo Herbelin, studied the relative intuitionistic strength of Gödel's completeness theorem, the ultrafilter lemma, and different forms of the Fan Theorem, as a way to transfer computational contents of proofs from one to the other theorems.

6.1.5. A theory of effects and resources

Arnaud Spiwack, in collaboration with Jean-Philippe Bernardy, Mathieu Boespflug, Ryan R. Newton and Simon Peyton-Jones, developed an extension of the type system of Haskell with linear types The work is to be presented at POPL'18.

In collaboration with Thomas Letan (Agence Nationale pour la Sécurité des Systèmes Informatiques), Yann Régis-Gianas studied how free monads can be used to develop modular implementations and proofs of effectful systems. This proof technique is applied to the formal study of architectural attacks on IBM PC like architectures.

6.1.6. Classical realisability and implicative algebras

Étienne Miquey has been working with Alexandre Miquel in Montevideo on the topic of implicative algebras. Implicative algebras are an algebraisation of the structure needed to develop a realisability model. In particular, they give rise to the usual ordered combinatory algebras and thus to the triposes used to model classical realisability. An implicative algebra is given by an implicative structure (which consists of a complete semi-lattice with a binary operation $\rightarrow$) together with a separator containing the element interpreted as true in the structure. Following the work of Guillaume Munch-Maccagnoni on focalisation and classical realisability, Étienne Miquey gave alternative presentations within structures based on other connectives rather than $\rightarrow$, namely disjunctive algebras (based on negation, "par") and conjunctive algebras (negation, tensor). Such connectives correspond to the decomposition of the arrow according to the strategy of evaluation (call-by-name/call-by-value). In particular, he showed that disjunctive algebras were particular cases of implicative algebras; and that conjunctive algebras can be obtained by duality from disjunctive algebras. Besides, Étienne Miquey has formalised the theory of implicative algebras (resp. disjunctive, conjunctive) in Coq.

6.2. Reasoning and programming with infinite data

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

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

6.2.1. Proof theory of infinitary and circular proofs

In collaboration with David Baelde and Guilhem Jaber, Amina Doumane and Alexis Saurin extended the proof theory of infinite proofs for fixpoint logics by relaxing the validity condition necessary to distinguish sound proofs from invalid ones. In CSL 2016, Baelde, Doumane and Saurin proved cut-elimination and focalisation for infinite proofs for $\mu MALL$ with a validity condition inspired from the acceptance condition of parity automata (or the winning condition of parity games). However, this validity condition rules out lots of proofs which are computational sound and does not account for the cut-axiom interaction in sequent proofs.

With Jaber, they relaxed the 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, approximating productivity. If the decidability of the validity condition in the most general case is still open, it allows for decidable restrictions which are still useful in the sense they allow for a much more flexible writing of circular proofs (or, through the proofs-as-programs bridge, circular progams). Cut-elimination is obtained in two steps, combining CSL 2016 result with a technique for "straightening" bouncing threads, that is perfoming just the necessary amount of cut-elimination to recover straight threads, the two results are combined thanks to a compression lemma, a standard result from infinitary rewriting ensuring that a transfinite strongly converging sequence can be turned into an ω-indexed strongly converging sequence. Preliminary results were presented at the Types 2017 conference.

6.2.2. Automata theory meets proof theory: completeness of the linear time mu-calculus.

Amina Doumane extended her previous results with David Baelde, Lucca Hirschi and Alexis Saurin proving a constructive completeness theorem for the full linear-time μ-calculus, while the previous results only captured a fragment of the linear-time mu-calculus expressing all inclusions of Büchi automata suitably encoded as formulas.

In order to achieve this tour de force (for which her publication at LICS 2017 received the Kleene award of the best student paper [38], see Highlights of the year), she identified several fragments of the linear-time mu-calculus corresponding to various classes of ω-automata and proved completeness of those classes by using circular proof systems and finitisation of the infinite proofs in the Kozen's usual axiomatisation (see paragraph on finitising circular proofs for more details).

6.2.3. 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.

In the setting of fixpoint logics with circular proofs, several investigations were carried on in the team:

- firstly, in the setting of the usual validity condition for circular proofs of $\mu MALL$, Doumane extended in her PhD thesis a translatibility criterion for finitising circular proofs which was first used in joint work with Baelde, Saurin and Hirschi and later applied to the full linear-time mu-calculus in her LICS 2017 paper. Her translatibility criterion abstracts the proof scheme for finitising circular proofs and is not formulated with respect to a specific fragment of the logic, but with respect to conditions allowing finitisation of the cycles.
- Secondly, Nollet, working with Saurin and Tasson, recently proposed 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. 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.

6.2.4. Co-patterns

In collaboration with Paul Laforgue (Master 2, University Paris 7), Yann Régis-Gianas developed an extension of OCaml with copatterns. Copatterns generalize standard ML patterns for algebraic datatypes: While a pattern-matching destructs a finite value defined using a constructor, a copattern-matching creates an infinite computation defined in terms of its answers to observations performed by the evaluation context. They exploits the duality between functions defined by pattern matching and functions that define codata by copattern-matching, going from the second to the first by introducing a well-typed inversion of control which is a purely local syntactic transformation. This result shows that copattern-matching can be added with no effort to any programming language equipped with second-order polymorphism and generalized algebraic datatypes. This work has been published in the proceeding of PPDP'17. A short paper has also been accepted at JFLA'18.

6.2.5. Streams, classical logic and the ordinal λ-calculus

Polonsky and Saurin defined an extension of infinitary λ-calculi allowing transfinite iteration of abstraction and ordinal sequences of applications, Λ^o, and established a standardisation theorem for this calculus. The $\Lambda\mu$-calculus can be embedded in this calculus, as well as Saurin's full Stream hierarchy: as a consequence, they obtain a uniform framework to investigate this family of calculi and provide uniform proofs of important results such a standardisation.

6.2.6. Theory of fixpoints in the lambda-calculus

In collaboration with Manzonetto, Polonsky and Simonsen, Saurin studied two long-standing conjectures on fixpoints in the λ-calculus: the "fixpoint property" and the "double-fixpoint conjecture". The former asserts that every λ-term admits either a unique or an infinite number of β-distinct fixpoints while the second, formulated by Statman, says that there is no fixpoint satisfying $Y\delta = Y$ for $\delta = \lambda y, x.x(yx)$. They proved the first conjecture in the case of open terms and refute it in the case of sensible theories (instead of β). Moreover, they provide sufficient conditions for both conjectures in the general case. Concerning the double-fixpoint conjecture, they propose a proof technique identifying two key properties from which the results would follow, while they leave as conjecture to prove that those actually hold. Those results are currently submitted to a journal [54].

6.3. Effective higher-dimensional algebra

Participants: Cyrille Chenavier, Pierre-Louis Curien, Yves Guiraud, Cédric Ho Thanh, Maxime Lucas, Philippe Malbos, Samuel Mimram, Jovana Obradović, Matthieu Sozeau.

6.3.1. Higher linear rewriting

Yves Guiraud and Philippe Malbos have completed a four-year long collaboration with Eric Hoffbeck (LAGA, Univ. Paris 13), 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 [70] 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 [69] and Poincaré-Birkhoff-Witth bases [106]. Then, they transposed the construction of [12], based on an extension of Squier's theorem [109] 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 has just been submitted for publication [48].

Cyrille Chenavier has continued his work on reduction operators, a functional point of view on rewriting in associative algebras initiated by Berger [63], on which his PhD thesis was focused [4]. First, using the lattice structure of the reduction operators, he gave a new algebraic characterisation of confluence, and developed a new algorithm for completion, based on an iterated use of the meet-operation of the lattice [29]. Then he related this completion procedure to Faugère's F4 completion procedure for noncommutative Gröbner bases [80]. Finally, he gave a construction of a linear basis of the space of syzygies of a set of reduction operations, and used this work to optimise his completion procedure [46].

6.3.2. Cubical higher algebra

Maxime Lucas, supervised by Yves Guiraud and Pierre-Louis Curien, has applied the rewriting techniques of Guiraud and Malbos [93] to prove coherence theorems for bicategories and pseudofunctors. He obtained a coherence theorem for pseudonatural transformations thanks to a new theoretical result, improving on the former techniques, that relates the properties of rewriting in 1- and 2-categories [32]. Then he has transposed to a cubical setting, and improved, the results of [12]. This first involved a deep foundational work on the connections between globular and cubical higher categories [52], generalising several already known links in a unique theoretical setting [67], [68], [58], [111]. Then, he could prove Squier's theorem, giving a construction of a polygraphic resolution of monoids in the category of cubical Gray monoids [51]. All these results are contained in his PhD thesis, that was successfully defended in December 2017 [24].

6.3.3. Coherent Presentations of Monoidal categories

Presentations of categories are a well-known algebraic tool to provide descriptions of categories by means of generators, for objects and morphisms, and relations on morphisms. Pierre-Louis Curien and Samuel Mimram have generalised this notion, in order to consider situations where the objects are considered modulo an equivalence relation, which is described by equational generators. When those form a convergent (abstract) rewriting system on objects, there are three very natural constructions that can be used to define the category which is described by the presentation: one consists in turning equational generators into identities (i.e. considering a quotient category), one consists in formally adding inverses to equational generators (i.e. localising the category), and one consists in restricting to objects which are normal forms. Under suitable coherence conditions on the presentation, the three constructions coincide, thus generalising celebrated results on presentations of groups. Those conditions are then extended to presentations of monoidal categories [30].

6.3.4. Categorified cyclic operads

The work of Pierre-Louis Curien and Jovana Obradović on categorified cyclic operads has been condtionally accepted in the Journal Applied Categorical Structures [47]. The revision will include a careful treatment of weakened identity laws, as well of weakened equivariance laws. It will also include the details of an example and an illustration of the work. The example involves a generalisation of profunctors, and the application is to the notion of anti-cyclic operad, which they prove to be "sign-coherent".

6.3.5. Syntactic aspects of hypergraph polytopes

In collaboration with Jelena Ivanović, Pierre-Louis Curien and Jovana Obradović have introduced an inductively defined tree notation for all the faces of polytopes arising from a simplex by truncations, that allows them to view inclusion of faces as the process of contracting tree edges. This notation instantiates to the well-known notations for the faces of associahedra and permutohedra. Various authors have independently introduced combinatorial tools for describing such polytopes. In this work, the authors build on the particular approach developed by Došen and Petrić, who used the formalism of hypergraphs to describe the interval of polytopes from the simplex to the permutohedron. This interval was further stretched by Petrić to allow truncations of faces that are themselves obtained by truncations, and iteratively so. The notation applies to all these polytopes, and this fact is illustrated by showing that it instantiates to a notation for the faces of the permutohedron-based associahedra, that consists of parenthesised words with holes. In their work, Pierre-Louis Curien, Jovana Obradović and Jelena Ivanović also explore links between polytopes and categorified operads, as a follow-up of another work of Došen and Petrić, who had exhibited some families of hypergraph polytopes (associahedra, permutohedra, and hemiassociahedra) describing the coherences, and the coherences between coherences etc., arising by weakening sequential and parallel associativity of operadic composition. Their work is complemented with a criterion allowing to recover the information whether edges of these "operadic polytopes" come from sequential, or from parallel associativity. Alternative proofs for some of the original results of Došen and Petrić are also given. A paper containing this material has been accepted in the Journal Homotopy and Related Structure [33].

6.3.6. Opetopes

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 has developped a corresponding syntax, which he presented at the workshop "Categories for homotopy and rewriting" (CIRM, September 2017).

Cédric Ho Thanh started his PhD work around opetopes in September 2017. His first contributions include a careful embedding of opetopic sets into polygraphs, and a (finite) critical pair lemma for opetopic sets. Indeed, opetopic sets seem to delimit a subset of polygraphs in which the basics of rewriting theory can be developped, without the anomalies already observed by Lafont and others happening, like the existence of a possibly infinite set of critical pairs in a rewriting system specified by finitely many rules. Opetopes are tree-like and hence first-order-term-like and that is the intuitive reason why these anomalies are avoided.

6.3.7. Higher Garside theory

Building on [9], Yves Guiraud is currently finishing with Matthieu Picantin (IRIF, Univ. Paris 7) a work that generalises already known constructions such as the bar resolution, several resolutions defined by Dehornoy and Lafont [78], 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.

6.3.8. Foundations and formalisation of higher algebra

Yves Guiraud has started a collaboration with Marcelo Fiore (Univ. Cambridge) on the foundations of higher-dimensional categories, with the aim to define a general notion of polygraphs for various notions of algebraic structures. This is based on seeing higher categories as n-oids in a specific n-oidal category (a category with n monoidal structures with exchange morphisms between them). With that point of view, a good notion of polygraph can be iteratively defined for monoids in any monoidal category with pullbacks, which is a sufficiently general setting for most purposes.

Eric Finster, Yves Guiraud and Matthieu Sozeau have started to explore the links between combinatorial higher algebra and homotopy type theory, two domains that describe computations with a homotopical point of view. Their first goal is to formalise the rewriting methods of [12] and [10] in homotopy type theory, establishing a first deep connection between the two fields. This direction will be explored further by Antoine Allioux, a PhD student co-directed by Guiraud and Sozeau, starting in February 2018.

6.4. Incrementality

Participants: Thibaut Girka, Yann Régis-Gianas, Kostia Chardonnet.

In collaboration with Colin Gonzalez, Yann Régis-Gianas developed BLACS, a programming framework that applies differential functional programming techniques to the implementation of asynchronous spreadsheets for big data.

In collaboration with Lelio Brun (ENS), Yann Régis-Gianas developed DeltaCoq, a library for certified incremental functional programming. A paper is in preparation.

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 will be submitted to ICFP 2018.

6.4.2. Difference languages

In collaboration with David Mentré (Mitsubishi), Thibaut Girka and Yann Régis-Gianas developed a metatheoretical framework to develop verifiable difference languages in Coq. Such formal differences capture semantic differences between close programs. This work appeared in PPDP'17 [39].

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, Cyprien Mangin, Théo Zimmermann.

6.5.1. Homotopy type theory

Hugo Herbelin worked on the computational contents of extensional equality in type theory. Exploiting the idea introduced in Cohen, Coquand, Huber and Mörtberg's Cubical Type Theory of equality as abstraction over a geometrical dimension, he developed a direct-style system of notations for a scoped iterated parametricity semantics. The resulting logic respects equivalence of types by construction, thus providing a simple computational content to the key axiom of Homotopy Type Theory, namely the axiom of univalence.

6.5.2. Proof irrelevance and Homotopy Type Theory

Gaëtan Gilbert (PhD student of N. Tabareau, Gallinette and M. Sozeau, started in 2016) is studying the integration of a new notion of propositions, called *strict* propositions, in the calculus of inductive constructions. This new sort dubbed sProp supports definitional proof-irrelevance (two proofs of a strict proposition are always convertible), while maintaining compatibility with Univalence or Uniqueness of Identity Proofs. The goal of this work is to provide a more comfortable programming experience in the system by allowing more proofs to be identified "for free" during conversion. This should have an impact both on programming with dependent types (avoiding issues with coercions during typechecking) and for the development of homotopy type theory (avoiding "trivial" transports of equality proofs on natural numbers for example). Gaëtan Gilbert has developed a prototype version integrating this extension in Coq.

6.5.3. Extensionality and Intensionality in Type Theory

Théo Winterhalter (internship co-advised by Matthieu Sozeau and Andrej Bauer in 2017, now PhD student at Inria Nantes, co-advised by Nicolas Tabareau and Matthieu Sozeau) studied a translation from extensional to intensional type theory during his internship with Matthieu Sozeau and a general framework for formalising variants of type theory previously with Andrej Bauer at the University of Ljubljana in Slovenia. They developed a revised version of the translation by Nicolas Oury which doesn't require the use of John Major equality nor suspicious axioms associated to it. It results in a mixed translation that can transport derivations of extensional type theory into intensional 2-level type theory (with an original, syntactic presentation of the latter). This allows in principle to use the convenience of the reflection rule of equality in proofs while being able to derive decorated terms checkable by the kernel of a 2-level variant of Coq: one where both a univalent equality and a strict equality with uniqueness of identity proofs can cohabit. They are working on a Coq formalisation of this result using the Template-Coq framework, which will be extracted to a translation plugin to provide this facility in Coq itself.

6.5.4. Dependent pattern-matching

Cyprien Mangin developed a new simplification engine on top of the Equations plugin. This simplification engine is similar to the one of Cockx [73], allowing an interpretation of dependent pattern-matching that is independent of axioms like UIP or Univalence. While refining the implementation, he also designed a few optimisations allowing for a smarter compilation scheme, in terms of the required properties of the objects and the size of the generated proofs. Matthieu Sozeau concentrated on making the treatment of recursive functions more robust and complete, leading to the first tool of this kind for Coq that can handle both mutual and nested structurally recursive functions along with nested well-founded definitions. The elimination principle generation part of the system was adapted accordingly, putting the tool in good position to replace the previous Function tool of Coq that supports neither dependent pattern-matching nor nested fixpoints. Matthieu Sozeau developed a number of examples showcasing the tool, the largest one having actually been first developed by a student of the MPRI 2.7.2 course. An article presenting this tool and the smart case analysis method is in revision [53]. Version 1.0 of the system was released in December 2017. Cyprien Mangin gave a demo / presentation of the tool at the EUTypes Type Theory Tools workshop in January 2017 and will present a poster and demonstration of the new version at PEPM 2018 in Los Angeles.

Thierry Martinez started 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. Transferring theorems along isomorphisms

Following his work on theorem transfer along (iso)morphisms, Théo Zimmermann has started to explore more fundamental aspects that are connected to it: the concept of logical relation, which was originally invented to prove behavioral equivalence of programs and served to formalise parametricity, seems, following Hermida, Reddy and Robinson, to correspond to a very generic relational notion of morphism that was precisely the one needed for transfer lemmas.

6.5.6. Unification

Matthieu Sozeau has developped a complete reimplementation of the basic tactics of Coq in terms of the type-inference unification algorithm of Coq. This work is scheduled to be integrated in part in the 8.8 version of Coq due next year. It should provide a clean slate for development of the 9 series of Coq relying solely on an algorithm close to the one studied with Beta Ziliani in [22].

6.5.7. Cumulativity for Inductive Types

Together with Amin Timany (PhD student of Bart Jacobs at KU Leuven), Matthieu Sozeau developed an extension of the Calculus of Inductive Constructions featuring cumulativity for inductive types [44]. This extension is useful for developments using universe polymorphism like Category Theory formalisations and the HoTT library [36] but also crucial to develop syntactic program translations that add structures to types, as advocated by Boulier et al [66], requiring to validate the cumulativity rule on sigma types. They showed the relative consistency of this extension of the calculus using a set-theoretic model, inspired by the one of Lee and Werner [102] for proof-irrelevance. This extension is integrated in the 8.7 release of Coq and involved a large amount of design and implementation work in particular in relation with the unification strategy used in presence of subtyping and delta reduction, extending the framework studied in [34]. An article describing this work is in revision.

6.6. Formalisation work

Participants: Jean-Jacques Lévy, Daniel de Rauglaudre.

6.6.1. Proofs of algorithms on graphs

Jean-Jacques Lévy and Chen Ran (a PhD student of the Institute of Software, Beijing, visiting the Toccata team 9 months until April 2017) pursue their work about formal proofs of algorithms. Their goal is to provide proofs of algorithms which ought to be both checked by computer and easily 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. There are mainly two algorithms: one by Kosaraju (1978) working in two phases (some formal proofs of it have already been achieved by Pottier with Coq and by Théry and Gonthier with Coq-SSReflect), one by Tarjan (1972) working in a single pass.

Their proofs use a first-order logic with definitions of inductive predicates. This logic is the one defined in the Why3 system (research-team Toccata, Saclay). They widely use automatic provers interfaced with Why3. A minor part of these proofs is also achieved in Coq. The difficulty of this approach is to combine automatic provers and intuitive design.

In 2017, the same proofs were fully completed in Coq-ssreflect by Cohen and Théry, and in Isabelle-HOL by Merz, both proofs with the assistance of J.-J. Lévy. A Fstar proof is also under development. These proofs are between a factor 4 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.

Part of this work (Tarjan 1972) was presented at JFLA 2017 in Gourette [41]. A more comprehensive version was presented at the VSTTE 2017 conference in Heidelberg [37]. Scripts of proofs can be found at http://jeanjacqueslevy.net/why3.

6.6.2. Banach-Tarski Paradox

Banach-Tarski Paradox states that, if we admit the axiom of choice, a sphere is equidecomposable into two spheres identical to the initial one. The equidecomposability is a property of geometric objects: two objects (sets) are equidecomposable if they can be partitioned into a same finite number of sets, and each set of the first object is mapped to a set of the second object by only rotations and translations. In other words, one breaks the first object into a finite number of pieces, and with them, one reconstructs the second object. Its pen and paper proof was done in 1924 by Banach and Tarski.

The formal proof was completed this year by Daniel de Rauglaudre, after 9 months, with a result of about 10000 lines of Coq. A paper about it was published in JFR (Journal of Formalized Reasoning) [35].

6.6.3. Univalence for Free

Together with E. Tanter at Inria Chile and N. Tabareau at Inria Nantes, Matthieu Sozeau developed the theory and implementation of an ad-hoc version of univalence. This axiom at the basis of Homotopy Type Theory morally says that all constructions of type theory are invariant under equivalence, which for programming purposes means invariance by isomorphism. Using a carefuly designed variant of the parametriciy translation for type theory, they can show that indeed all type constructors of type theory, except indexed inductive types with non-hset indices respect univalence. In practice, this leads to a type-class based framework for constructing the proofs that values of a given type do indeed transport equivalences/isomorphisms correctly, relying on univalence itself only for universes and in well-delimited places. An article about this work is in revision [57].

6.6.4. 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 for the Coq programming language down to CompCert C-light which provides itself a certified compilation path to assembly language. The compiler can already be run and most phases are proven correct. As part of this work, Matthieu Sozeau took the lead of the Template-Coq library development originally developed by Gregory Malecha and extended it. Template-Coq provides quoting and unquoting facilities for Coq's kernel syntax and environment to Coq, allowing to reason on the actual definitions checked

by the Coq system in Coq itself. For CertiCoq, the quoted type of Coq terms corresponds to its frontend language. The plugin can however be used in many other ways, notably to implement certified syntactic translations from Coq (or extended theories) to Coq, and to develop plugins to the Coq system in Coq itself. Together with Nicolas Tabareau and Simon Boulier in Nantes and Abhishek Anand at Cornell University, they are developing a general plugin for certified meta-programming in the system. It will be presented at CoqPL'18 [42]. Matthieu Sozeau worked in particular on reimplementing the basic typing and conversion algorithms of Coq inside Coq itself, providing a mechanised specification of the implementation of the system that can be used to verify arbitrarily large parts of it. The type inference algorithm developed there is also useful to help writting program translations on the "forgetful" kernel syntax.

7. Partnerships and Cooperations

7.1. National Initiatives

Alexis Saurin (coordinator) and Yann Régis-Gianas 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.

Pierre-Louis Curien (coordinator), Yves Guiraud (local coordinator), Philippe Malbos and Samuel Mimram are members of the four-year Cathre ANR project (January 2014 to December 2017). This project investigates the general theory of higher-dimensional rewriting, the development of a general-purpose library for higher-dimensional rewriting, and applications in the fields of combinatorial linear algebra, combinatorial group theory and theoretical computer science. This project is joint with mathematicians and computer scientists from LAGA (Univ. Paris 13), LIX (École Polytechnique), ICJ (Univ. Lyon 1 and Univ. Saint-Étienne), I2M (Univ. Aix-Marseille) and IMT (Univ. Toulouse 3). The project Cathre provided the funding for the PhD of Maxime Lucas.

Pierre-Louis Curien, Yves Guiraud, Hugo Herbelin, Philippe Malbos, Samuel Mimram and Alexis Saurin are members of the GDR Informatique Mathématique, in the Géocal (Geometry of computation) and LAC (Logic, algebra and computation) working groups.

Pierre-Louis Curien, Yves Guiraud (local coordinator), Philippe Malbos, Samuel Mimram 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.

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

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

Matthieu Sozeau is a member of the CoqHoTT project led by Nicolas Tabareau (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. Inria International Labs

7.3.1.1. Other 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 will officially start 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 Hasuoi (NII, Tokyo). The full name of the project is Concurrent, Resourceful and Effectful 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 a 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 Inria-CAS project (a new programme of Inria with the 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, starting in December 2017.

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, and Universidad Nacional de Córdoba.

7.3.4. Participation in Other International Programs

Pierre-Louis Curien participates to the ANR International French-Chinese project LOCALI (Logical Approach to Novel Computational Paradigms), coordinated by Gilles Dowek (Deducteam). This project ended in July 2017.

7.4. International Research Visitors

7.4.1. Visits of International Scientists

John Baez (University of California River Side) visited the team for a week in November 2017.

Marcelo Fiore (University of Cambridge) visited the team for two weeks in February 2017.

Jovana Obradović (now a postdoc at Charles University, Prague) visited the team from December 1 to December 10 2017.

Amin Timany (KU Leuven, Belgium) visited the team for two months in March-April 2017 and collaborated with Matthieu Sozeau on the design and implementation of cumulative inductive types in Coq.

7.4.2. Visits to International Teams

7.4.2.1. Research Stays Abroad

Pierre-Louis Curien visited East China Normal University for a month in June 2017 (collaborations with Yuxin Deng and Min Zhang). Pierre-Louis Curien and Jovana Obradović visited the Institute of Mathematics of the Serbian Academy of Sciences in Belgrade in July 2017 (collaboration with Zoran Petrić).

Jean-Jacques Lévy visited the Institute of Software of Chinese Academy of Sciences (ISCAS) in December 2017 (project VIP and on-going work with Ran Chen) during 2 weeks. He gave talks at ISCAS hosted by Ying Jiang, and during a third week at ECNU Shanghai hosted by Min Zhang, USTC Suzhou (University of Science and Technology of China) hosted by Xinyu Feng, Nankai University in Tianjin hosted by Chunfu Jia.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Organisation

8.1.1.1. General Chair, Scientific Chair

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

8.1.1.2. Member of the Organising Committees

Yann Régis-Gianas was multimedia chair of the organising committee of POPL 2017 that took place in Paris in January 2017.

Yves Guiraud, Philippe Malbos and Samuel Mimram have organised the third edition of the Higher-Dimensional Rewriting and Algebra (HDRA) workshop of the Formal Structures for Computation and Deduction conference (FSCD), held in Oxford in September 2017.

Yves Guiraud and Samuel Mimram, with Dimitri Ara (Univ. Aix-Marseille) have organised the "Categories in Homotopy and Rewriting" one-week international conference, at the CIRM, in Marseille, in September 2017. As the closing conference of the Cathre ANR project, focused on higher-dimensional algebra, this conference attracted 80 participants, working in category theory, algebraic topology, logic and theoretical computer science.

Matthieu Sozeau co-organised with Nicolas Tabareau the 3rd Coq Implementors Workshop in Le Croisic, France, June 12-16 2017. It included presentations from developers, both from France and abroad and a large amount of hacking.

8.1.2. Scientific Events Selection

8.1.2.1. Member of the Conference Program Committees

Hugo Herbelin was a member of the program committees of the conference FSCD'17, of the TYPES'17 venue, as well as of the PxTP'17 and CoqPL'18 workshops.

Matthieu Sozeau was member of the program committee of CoqPL'17.

Yann Régis-Gianas was member of the program committee of JFLA'18.

Alexis Saurin was a member of the program committee of the workshop Coinduction in Type Theory which took place in Chambéry from the 3rd to the 6th of July, 2017

Alexis Saurin was a member of the program committee of the workshop on Trends in Linear Logic and Applications which took place in Oxford on the 3rd of September 2017 as a satellite event of FSCD conference.

8.1.2.2. Member of the Conference Steering Committees

Hugo Herbelin was a member of the steering committee of the conference *Formal Structures for Computation and Deduction* (FSCD) until September 2017.

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

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

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

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 an invited talk at the Conference Categories for Homotopy Theory and Rewriting on "A syntactic approach to polynomial functors, polynomial monads and opetopes" (September 2017).

Yann Régis-Gianas gave an invited talk at the Conference for Trends in Functional Programming In Education on the OCaml MOOC (April 2017).

8.1.5. Scientific Expertise

Pierre-Louis Curien and Yves Guiraud have been members of the "Comité de sélection" for an assistant professor position in mathematical foundations of computer science at the University Paris 7 (spring 2017).

8.1.6. Research Administration

Pierre-Louis Curien, Hugo Herbelin and Yves Guiraud are members of the scientific council of the Computer Science deparment of University Paris 7.

Yves Guiraud is the head of the "Preuves, Programmes and Systèmes (PPS)" pole of the IRIF laboratory (since April 2016), a member of the IRIF council (January 2016 - December 2017), and of the IRIF direction council (since September 2017).

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

8.1.7. Presentation of papers

Hugo Herbelin gave a talk at the GT Geocal-Lac on a proof of Gödel's completeness theorem using side-effects.

Hugo Herbelin gave a talk at the PPS pole days on the intuitionistic reverse mathematics properties of Gödel's completeness theorem.

Étienne Miquey gave a talk at the ESOP conference in Uppsala (Sweden) on "A classical sequent calculus with dependent types" (April 2017).

Jovana Obradović gave a talk at the Conference Topology in Ecuador (Galapagos Islands) on "Categorified cyclic operads" (August 2017).

Théo Zimmermann gave a talk on Coq's Prolog and application to defining semi-automatic tactics at the TTT'17 workshop (Type-Theory Based Tools).

Cyprien Mangin gave a talk on Equations at the TTT'17 workshop.

Matthieu Sozeau gave a talk on Coq 8.6 at the CoqPL'17 workshop in January 2017.

8.1.8. Talks in seminars

Pierre-Louis Curien gave talks at NII (Hasuo's Lab, Tokyo) and at Shanghai Jiaotong University on "Trialgebraic structures on faces of some families of polytopes" (October-November 2017).

Théo Zimmermann gave a talk on transfer of isomorphisms at the Deducteam seminar (April 2017).

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

Hugo Herbelin attended TTT'17, TYPES'17 and FSCD'17.

Cyrille Chenavier, Maxime Lucas, Philippe Malbos and Samuel Mimram attended the HDRA workshop in Oxford (September 2017).

Cyrille Chenavier, Pierre-Louis Curien, Yves Guiraud, Maxime Lucas, Philippe Malbos and Samuel Mimram attended the "Categories in Homotopy and Rewriting" conference in Marseille (September 2017).

Hugo Herbelin attended the conferences Types 2017 in Budapest (Hungaria, May), the Coq implementors workshop in Le Croisic (June), the FSCD conference in Oxford (UK, September). He attended various workshops of the POPL'17 and FSCD'17 conference, including TTT'17 and CoqPL'17. He participated to the Big Proofs seminar in Cambridge (July).

Jean-Jacques Lévy participated to CAV and VSTTE 2017 conferences, Heidelberg, Germany, July 21-28 where his co-author Ran Chen presented the article [37].

Matthieu Sozeau attended POPL'17, CPP'17 and CoqPL'17 in Paris (January), TYPES'17 in Budapest (Hungaria, May) and the Coq implementors workshop in Le Croisic (June).

Étienne Miquey attended the conference ESOP in Uppsala (Sweden) in April 2017.

Pierre-Louis Curien and Jovana Obradović attended the conference Topology in Ecuador (Galapagos Islands) in August 2017, and the conference Geometry and combinatorics of associativity in Dublin in October 2017.

Pierre-Louis Curien and Claudia Faggian attended the CRECOGI meeting in Ito (Japan), and Claudia Faggian gave a talk on "Proof techniques in Probabilistic Reduction Systems" (November 2017).

Théo Zimmermann attended the TTT'17 workshop (January 2017) and the Coq implementors workshop in Le Croisic (June 2017).

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 2017 were Pierre-Marie Pédrot, Guilhem Jaber, Francesco A. Genco, Ludovic Patey.

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 (IRIF) 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).

Master: Hugo Herbelin teaches the course on the proof-as-program correspondence for classical logic and beyond at the LMFI.

Master: Pierre Letouzey teaches two short courses to the LMFI Master 2 students : "Models of programming" 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).

Master: Yves Guiraud gave a course on the applications of rewriting methods in algebra in the M2 Mathématiques Fondamentales of Lyon (Univ. Lyon 1 and ENS Lyon).

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 taught a course at the EJCP'17 summer school in Toulous in June 17, on an introduction to interactive theorem proving.

Master: Alexis aurin 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.

8.2.2. Supervision

Internship: Hugo Herbelin has supervised the M2 internship of Charlotte Barot.

Internship: Yann Régis-Gianas has supervised the L3 internship of Kostia Chardonnet.

Internship: Yann Régis-Gianas has supervised the M2 internship of Colin Gonzalez.

PhD (completed) Amina Doumane, supervised by Alexis Saurin, David Baelde and Pierre-Louis Curien, successfully defended in June 2017 (title: On the infinitary proof theory of logics with fixed points).

PhD (completed): Étienne Miquey, co-supervised by Hugo Herbelin and Alexandre Miquel, Réalisabilité classique et effets de bords, September 2014, successfully defended in November 2017.

PhD (completed): Jovana Obradović, supervised by Pierre-Louis Curien, successfully defended in Septemver 2017 (title: Cyclic operads: syntactic, algebraic and categorified aspects).

PhD (completed): Maxime Lucas, supervised by Yves Guiraud and Pierre-Louis Curien, successfully defended in December 2017 (title: Cubical categories for homotopy and rewriting).

PhD in progress: Guillaume Claret, Programmation avec effets en Coq, (started in September 2012), supervised by Hugo Herbelin and Yann Régis-Gianas, the dissertation was completed in 2015 but Guillaume Claret moved in the meantime to a private company and the defense has been delayed to 2018.

PhD in progress: Thibaut Girka, Differential semantics (started in January 2014), supervised by Roberto Di Cosmo and Yann Régis-Gianas.

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

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

PhD starting: Cédric Ho Thanh, on Opetopes for higher-dimensional rewriting and koszulity, supervised by Pierre-Louis Curien and Samuel Mimram.

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 Samuel Mimram and Yves Guiraud.

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 referee for the Habilitation of Russ Harmer (ENS Lyon, May 2017).

Pierre-Louis Curien and Alexis Saurin were members of the jury of the thesis of Amina Doumane (Paris 7, June 2017).

Pierre-Louis Curien was president of the jury of the Habilitation of Matthieu Picantin (Paris 7, July 2017).

Pierre-Louis Curien was president of the jury of the thesis of Simon Castellan (ENS Lyon, July 2017).

Pierre-Louis Curien was member of the jury of the thesis of Jovana Obradović (Paris 7, September 2017).

Pierre-Louis Curien was referee for the Habilitation of Paul-André Melliès (Paris 7, November 2017).

Pierre-Louis Curien was president of the jury of Habilitation of Damiano Mazza (Paris 13, December 2017).

Pierre-Louis Curien was referee for the Habilitation of Samuele Giraudo (Marne la Vallée, December 2017).

Pierre-Louis Curien was member of the jury of the thesis of Nicolas Ninin (Paris Saclay, December 2017).

Pierre-Louis Curien was president of the jury of the thesis of Luc Pellissier (Paris 13, December 2017).

Pierre-Louis Curien was referee for the thesis of Christopher Nguyen (Macquarie University, Sydney, December 2017).

Pierre-Louis Curien and Yves Guiraud were members of the jury of the thesis of Maxime Lucas (Paris 7, December 2017).

8.3. Popularization

Hugo Herbelin wrote with Sandrine Blazy and Pierre Castéran an introduction to Coq for engineers edited by Techniques de l'Ingénieur.

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] 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

[12] 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

[13] 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

[14] 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

[15] 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/

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

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

[22] 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

[23] *Best Paper*
A. DOUMANE. *On the infinitary proof theory of logics with fixed points*, Université Paris 7 - Diderot, June 2017, https://hal.archives-ouvertes.fr/tel-01676953.

[24] M. LUCAS. *Cubical categories for homotopy and rewriting*, Université Paris 7, Sorbonne Paris Cité, December 2017, https://hal.archives-ouvertes.fr/tel-01668359

[25] É. MIQUEY. *Classical realizability and side-effects*, Université Sorbonne Paris Cité - Université Paris Diderot (Paris 7) ; Universidad de la República - Montevideo, Uruguay, November 2017, https://hal.inria.fr/tel-01653733

[26] J. OBRADOVIC. *Cyclic operads: syntactic, algebraic and categorified aspects*, Université Paris Diderot - Paris 7 - Sorbonne Paris Cité, September 2017, https://hal.archives-ouvertes.fr/tel-01676983

Articles in International Peer-Reviewed Journals

[27] J.-P. BERNARDY, M. BOESPFLUG, R. R. NEWTON, S. PEYTON JONES, A. SPIWACK. *Linear Haskell: practical linearity in a higher-order polymorphic language*, in "Proceedings of the ACM on Programming Languages", December 2017, vol. 2, n° POPL, pp. 1-29, https://arxiv.org/abs/1710.09756 [*DOI :* 10.1145/3158093], https://hal.archives-ouvertes.fr/hal-01673536

[28] S. BLAZY, P. CASTÉRAN, H. HERBELIN. *L'Assistant de Preuve Coq Table des matières*, in "Techniques de l'Ingenieur", August 2017, https://hal.inria.fr/hal-01645486

[29] C. CHENAVIER. *Reduction Operators and Completion of Rewriting Systems*, in "Journal of Symbolic Computation", 2017, https://arxiv.org/abs/1605.00174 , forthcoming, https://hal.archives-ouvertes.fr/hal-01325907

[30] P.-L. CURIEN, S. MIMRAM. *Coherent Presentations of Monoidal Categories*, in "Logical Methods in Computer Science", September 2017, vol. 13, n° 3, pp. 1-38, https://arxiv.org/abs/1705.03553 [*DOI :* 10.23638/LMCS-13(3:31)2017], https://hal.inria.fr/hal-01662524

[31] F. LOULERGUE, W. BOUSDIRA, J. TESSON. *Calculating Parallel Programs in Coq using List Homomorphisms*, in "International Journal of Parallel Programming", 2017, vol. 45, n° 2, pp. 300-319 [*DOI :* 10.1007/s10766-016-0415-8], https://hal.inria.fr/hal-01159182

[32] M. LUCAS. *A coherence theorem for pseudonatural transformations*, in "Journal of Pure and Applied Algebra", 2017, vol. 221, n° 5, pp. 1146-1217, https://arxiv.org/abs/1508.07807 [*DOI :* 10.1016/J.JPAA.2016.09.005], https://hal.archives-ouvertes.fr/hal-01191867

[33] J. OBRADOVIC, P.-L. CURIEN, J. IVANOVIC. *Syntactic aspects of hypergraph polytopes*, in "Journal of Homotopy and Related Structures", 2017, forthcoming, https://hal.archives-ouvertes.fr/hal-01669490

[34] 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

[35] D. DE RAUGLAUDRE. *Formal Proof of Banach-Tarski Paradox*, in "Journal of Formalized Reasoning", October 2017, vol. 10, n° 1, pp. 37-49 [*DOI :* 10.6092/ISSN.1972-5787/6927], https://hal.archives-ouvertes.fr/hal-01673378

International Conferences with Proceedings

[36] A. BAUER, G. JASON, P. LUMSDAINE, M. SHULMAN, M. SOZEAU, B. SPITTERS. *The HoTT Library: A Formalization of Homotopy Type Theory in Coq*, in "CPP'17", Paris, France, CPP'17, ACM, January 2017, 9 p. [*DOI :* 10.1145/3018610.3018615], https://hal.inria.fr/hal-01421212

[37] R. CHEN, J.-J. LÉVY. *A Semi-automatic Proof of Strong connectivity*, in "9th Working Conference on Verified Software: Theories, Tools and Experiments (VSTTE)", Heidelberg, Germany, July 2017, https://hal.inria.fr/hal-01632947

[38] *Best Paper*
A. DOUMANE. *Constructive completeness for the linear-time μ-calculus*, in "Conference on Logic in Computer Science 2017", Reykjavik, Iceland, June 2017, https://hal.archives-ouvertes.fr/hal-01430737.

[39] T. GIRKA, D. MENTRÉ, Y. RÉGIS-GIANAS. *Verifiable Semantic Difference Languages*, in "International Symposium on Principles and Practice of Declarative Programming", Namur, Belgium, October 2017 [*DOI :* 10.1145/3131851.3131870], https://hal.inria.fr/hal-01653283

[40] É. MIQUEY. *A Classical Sequent Calculus with Dependent Types* , in "26th European Symposium on Programming", Uppsala, Sweden, April 2017, https://hal.inria.fr/hal-01375977

National Conferences with Proceedings

[41] R. CHEN, J.-J. LÉVY. *Une preuve formelle de l'algorithme de Tarjan-1972 pour trouver les composantes fortement connexes dans un graphe*, in "JFLA 2017 - Vingt-huitièmes Journées Francophones des Langages Applicatifs", Gourette, France, Vingt-huitièmes Journées Francophones des Langages Applicatifs, January 2017, https://hal.inria.fr/hal-01422215

Conferences without Proceedings

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

[43] T. ZIMMERMANN, H. HERBELIN. *Coq's Prolog and application to defining semi-automatic tactics*, in "Type Theory Based Tools", Paris, France, January 2017, https://hal.archives-ouvertes.fr/hal-01671994

Research Reports

[44] A. TIMANY, M. SOZEAU. *Consistency of the Predicative Calculus of Cumulative Inductive Constructions (pCuIC)*, KU Leuven, Belgium ; Inria Paris, October 2017, n^{o} RR-9105, 30 p. , Version 2 fixes some typos from version 1, https://hal.inria.fr/hal-01615123

Other Publications

[45] C. CHENAVIER. *A Lattice Formulation of the F 4 Completion Procedure*, March 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01489200

[46] C. CHENAVIER. *Syzygies among reduction operators*, August 2017, https://arxiv.org/abs/1708.08709 - working paper or preprint, https://hal.archives-ouvertes.fr/hal-01578555

[47] 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

[48] Y. GUIRAUD, E. HOFFBECK, P. MALBOS. *Convergent presentations and polygraphic resolutions of associative algebras*, December 2017, 65 pages, https://hal.archives-ouvertes.fr/hal-01006220

[49] H. HERBELIN, É. MIQUEY. *Normalization and continuation-passing-style interpretation of simply-typed call-by-need λ-calculus with control*, July 2017, working paper or preprint, https://hal.inria.fr/hal-01570987

[50] N. JEANNEROD, Y. RÉGIS-GIANAS, R. TREINEN. *Having Fun With 31.521 Shell Scripts*, April 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01513750

[51] M. LUCAS. *A cubical Squier's theorem*, December 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01662132

[52] M. LUCAS. *Cubical (ω,p)-categories*, December 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01662127

[53] C. MANGIN, M. SOZEAU. *Equations reloaded*, December 2017, working paper or preprint, https://hal.inria.fr/hal-01671777

[54] G. MANZONETTO, A. POLONSKY, A. SAURIN, J. G. SIMONSEN. *The Fixed Point Property and a Technique to Harness Double Fixed Point Combinators* , December 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01672846

[55] É. MIQUEY, H. HERBELIN. *Realizability interpretation and normalization of typed call-by-need λ-calculus with control*, October 2017, working paper or preprint, https://hal.inria.fr/hal-01624839

[56] É. MIQUEY. *A Classical Sequent Calculus with Dependent Types (Extended Version)*, December 2017, working paper or preprint, https://hal.inria.fr/hal-01519929

[57] N. TABAREAU, É. TANTER, M. SOZEAU. *Equivalences for Free!: Univalent Parametricity for Effective Transport*, July 2017, working paper or preprint, https://hal.inria.fr/hal-01559073

References in notes

[58] F. A. AL-AGL, R. BROWN, R. STEINER. *Multiple categories: the equivalence of a globular and a cubical approach*, in "Adv. Math.", 2002, vol. 170, n° 1, pp. 71–118, https://doi.org/10.1006/aima.2001.2069

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

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

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

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

[63] R. BERGER. *Confluence and Koszulity*, in "J. Algebra", 1998, vol. 201, n° 1, pp. 243–283

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

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

[66] 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

[67] R. BROWN, P. J. HIGGINS. *The equivalence of ∞-groupoids and crossed complexes*, in "Cahiers Topologie Géom. Différentielle", 1981, vol. 22, n° 4, pp. 371–386

[68] R. BROWN, P. J. HIGGINS. *The equivalence of ω-groupoids and cubical T-complexes*, in "Cahiers Topologie Géom. Différentielle", 1981, vol. 22, n° 4, pp. 349–370

[69] 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

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

[71] 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

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

[73] J. COCKX, D. DEVRIESE, F. PIESSENS. *Pattern matching without K*, in "Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014", 2014, pp. 257–268, http://doi.acm.org/10.1145/2628136.2628139

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

[75] 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

[76] 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

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

[78] 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

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

[80] J.-C. FAUGÉRE. *A new efficient algorithm for computing Gröbner bases* (F_4), in "J. Pure Appl. Algebra", 1999, vol. 139, n^{o} 1-3, pp. 61–88, Effective methods in algebraic geometry (Saint-Malo, 1998), https://doi.org/10.1016/S0022-4049(99)00005-5

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

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

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

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

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

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

[102] G. LEE, B. WERNER. *Proof-irrelevant model of CC with predicative induction and judgmental equality*, in "Logical Methods in Computer Science", 2011, vol. 7, n° 4

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

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

[105] 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

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

[107] 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

[108] 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

[109] 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

[110] 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

[111] R. STEINER. *Omega-categories and chain complexes*, in "Homology Homotopy Appl.", 2004, vol. 6, n^{o} 1, pp. 175–200, http://projecteuclid.org/euclid.hha/1139839551

[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