IN PARTNERSHIP WITH:
**CNRS**

**Université Denis Diderot (Paris 7)**

Activity Report 2019

# Project-Team PI.R2

Design, study and implementation of languages for proofs and programs

IN COLLABORATION WITH: Institut de Recherche en Informatique Fondamentale

RESEARCH CENTER
**Paris**

THEME
**Proofs and Verification**

# Table of contents

# Project-Team PI.R2

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

**Keywords:**

**Computer Science and Digital Science:**

A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.11. - Proof languages
A2.4.3. - Proofs
A7.2. - Logic in Computer Science
A7.2.3. - Interactive Theorem Proving
A7.2.4. - Mechanized Formalization of Mathematics
A8.1. - Discrete mathematics, combinatorics
A8.4. - Computer Algebra

**Other Research Topics and Application Domains:**

B6.1. - Software industry

# 1. Team, Visitors, External Collaborators

**Research Scientists**

Pierre-Louis Curien [Team leader until October 2019, CNRS, Senior Researcher, Emeritus since October 2019, HDR]
Thierry Coquand [University of Gothenburg, Senior Researcher, until November 2020]
Emilio Jesús Gallego Arias [Inria, Starting Research Position, from November 2019]
Yves Guiraud [Inria, Researcher, HDR]
Hugo Herbelin [Inria, Senior Researcher, HDR]
Jean-Jacques Lévy [Inria, Emeritus, HDR]
Alexis Saurin [Team leader since October 2019, CNRS, Researcher]
Matthieu Sozeau [Inria, Researcher]

**Faculty Members**

Pierre Letouzey [Université Paris Diderot, Associate Professor]
Yann Régis-Gianas [Université Paris Diderot, Associate Professor, HDR]

**Post-Doctoral Fellows**

Amar Hadzihasanovic [Sorbonne Université, Post-Doctoral Fellow, from December 2019]
Kailiang Ji [Inria, Post-Doctoral Fellow, until March 2019]
Exequiel Rivas Gadda [Inria, Post-Doctoral Fellow, until February 2019]

**PhD Students**

Antoine Allioux [Université Paris Diderot, PhD Student, from January 2018]
Félix Castro [Université Paris Diderot, PhD Student, from October 2019]
Kostia Chardonnet [Université Paris-Saclay, PhD Student, from November 2019]
Abhishek De [Université Paris Diderot, PhD Student]
Alen Đurić [Université Paris Diderot, PhD Student, from October 2019]
Colin Gonzalez [Nomadics Lab, PhD Student, from October 2019, granted by CIFRE]
Cédric Ho Thanh [Université Paris Diderot, PhD Student]
Farzad Jafar-Rahmani [Université Paris Diderot, PhD Student, from October 2019]

Hugo Moeneclaey [Université Paris-Saclay, PhD Student, from September 2019]
Théo Zimmermann [Université Paris Diderot, PhD Student, until December 2019]

**Technical staff**

Daniel de Rauglaudre [Inria, Engineer]

**Interns and Apprentices**

Adrien Champougny [Inria, from April 2019 until July 2019]
Vincent Tourneur [Inria, from March 2019 until August 2019]

**External Collaborator**

Jovana Obradović [Institute of Mathematics, Csech Academy of Science]

# 2. Overall Objectives

## 2.1. Overall Objectives

The research conducted in $\pi r^2$ 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 Gentsen [77] 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 [72], then by Howard and de Bruijn at the end of the 60's [89], [109], 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 $\lambda$-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 [69], [70] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [107].

### *3.1.2. Towards the calculus of constructions*

The $\lambda$-calculus, defined by Church [67], 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 $\lambda$-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 $\lambda$-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 [60].

To explain the Curry-Howard correspondence, it is important to distinguish between intuitionistic and classical logic: following Brouwer at the beginning of the $20^{\text{th}}$ 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) $\lambda$-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 [98].

In 1985, Coquand and Huet [69], [70] in the Formel team of Inria-Rocquencourt explored an alternative approach based on Girard-Reynolds' system $F$ [78], [102]. 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 [71] designed an extension of the Calculus of Constructions with a generalisation of algebraic types called inductive types, leading to the Calculus of Inductive Constructions (CIC) that started to serve as a new foundation for the Coq system. This new system, which got its current definitive name Coq, was released in 1991.

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

## 3.2. The development of Coq

During 1984-2012 period, about 40 persons have contributed to the development of Coq, out of which 7 persons have contributed to bring the system to the place it was six years ago. First Thierry Coquand through his foundational theoretical ideas, then Gérard Huet who developed the first prototypes with Thierry Coquand and who headed the Coq group until 1998, then Christine Paulin who was the main actor of the system based on the CIC and who headed the development group from 1998 to 2006. On the programming side, important steps were made by Chet Murthy who raised Coq from the prototypical state to a reasonably scalable system, Jean-Christophe Filliâtre who turned to concrete the concept of a small trustful certification kernel on which an arbitrary large system can be set up, Bruno Barras and Hugo Herbelin who, among other extensions, reorganised Coq on a new smoother and more uniform basis able to support a new round of extensions for the next decade.

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

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

In the last seven 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 [43] and wiki), made its development process public, starting to use public pull requests to track the work of developers, organising yearly hackatons/coding-sprints for the dissemination of expertise and developers & users meetings like the Coq Workshop and CoqPL, and, perhaps more anecdotally, retransmitting Coq working groups on a public YouTube channel.

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

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

On top of the developer community, there is a much wider user community, as Coq is being used in many different fields. The Software Foundations series, authored by academics from the USA, along with the reference Coq'Art book by Bertot and Castéran [61], the more advanced Certified Programming with Dependent Types book by Chlipala [66] and the recent book on the Mathematical Components library by Mahboubi, Tassi et al. provide resources for gradually learning the tool.

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

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

We next briefly describe the main components of Coq.

### *3.2.1. The underlying logic and the verification kernel*

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

### 3.2.2. Programming and specification languages

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

### 3.2.3. Standard library

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

### 3.2.4. Tactics

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

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

### 3.2.5. Extraction

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

### 3.2.6. Documentation

Coq is a feature-rich system and requires extensive training in order to be used proficiently; current documentation includes the reference manual, the reference for the standard library, as well as tutorials, and related tooling [sphinx plugins, coqdoc]. The jsCoq tool allows writing interactive web pages were Coq programs can be embedded and executed.

### 3.2.7. Proof development infrastructure

Coq is used in large-scale proof developments, and provides users miscellaneous tooling to help with them: the coq_makefile and Dune build systems help with incremental proof-checking; the Coq OPAM repository contains a package index for most Coq developments; the CoqIDE, ProofGeneral, and VSCoq user interfaces are environments for proof writing; and the Coq's API does allow users to extend the system in many important ways. Among the current extensions we have QuickChik, a tool for property-based testing; STMCoq and CoqHammer integrating Coq with automated solvers; ParamCoq, providing automatic derivation of parametricity principles; MetaCoq for metaprogramming; Equations for dependently-typed programming; SerAPI, for data-centric applications; etc... This also includes the main open Coq repository living at Github.

## 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 $\mathrm{ML}_F$ 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 [79] 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 [96] and Reynolds [101] and started to be studied in an abstract way in the 80's by Felleisen *et al* [75], leading to Parigot's $\lambda\mu$-calculus [99], 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 $\lambda$-calculus is Landin's SECD machine [95] and Krivine's abstract machine for call-by-name evaluation [92], [91]. 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 [76]: any side-effect expressible in monadic style (and this covers references, exceptions, states, dynamic bindings, ...) can be simulated in $\lambda$-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, $\infty$-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* $\infty$*-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 [59]) 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 [74]).

The categorification process has also reached logic, with the introduction of homotopy type theory. After a preliminary result that had identified categorical structures in type theory [88], it has been observed recently that the so-called "identity types" are naturally equiped with a structure of $\infty$-groupoid: the 1-cells are the proofs of equality, the 2-cells are the proofs of equality between proofs of equality, and so on. The striking resemblance with the fundamental $\infty$-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 [106], and independently by Burroni [64], 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 [93], [97], [80], [81], and beyond: Petri nets [83] and formal proofs of classical and linear logic have been expressed in this framework [82]. 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 [62].

### 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 works of Anick [57] and Squier [105], [104]: Squier has proved that, if a monoid $M$ can be presented by a *finite*, *terminating* and *confluent* rewriting system, then its third integral homology group $H_3(M, \mathbb{Z})$ is finitely generated and the monoid $M$ has *finite derivation type* (a property of homotopical nature). This allowed him to conclude that finite convergent rewriting systems were not a universal solution to decide the word problem of finitely generated monoids. Since then, Yves Guiraud and Philippe Malbos have shown that this connection was part of a deeper unified theory when formulated in the higher-dimensional setting [14], [15], [85], [86], [87].

In particular, the computational content of Squier's proof has led to a constructive methodology to produce, from a convergent presentation, *coherent presentations* and *polygraphic resolutions* of algebraic structures, such as monoids [14] and algebras [32]. 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, corresponding each 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 [94], [58]). 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 [11], and in homological algebra [84][32].

# 4. Highlights of the Year

## 4.1. Highlights of the Year

A one-day scientific meeting in honour of Pierre-Louis Curien's retirement was held at Université Paris Diderot on se 6, 2019 (organisers Antonio Bucciarelli, Bérénice Delcroix-Oger and Thomas Ehrhard) (https://www.irif.fr/plcmeeting).

The paper [35] presents the results of a large collaborative work led by Matthieu Sozeau on the metatheory and implementation of Coq's type theory in Coq itself.

Yves Guiraud defended his habilitation thesis on the 18th of June 2019, entitled "Rewriting methods in higher algebra". Yann Régis-Gianas defended his habilitation thesis on the 22nd of November 2019 entitled "About some metamorphoses of computer programs".

Yves Guiraud was granted an Action Exploratoire, Réécriture Algébrique, to start in January 2020. Emilio Gallego Arias joined the team in November 2019 on a Starting Research Position.

# 5. New Software and Platforms

## 5.1. Coq

*The Coq Proof Assistant*

KEYWORDS: Proof - Certification - Formalisation

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

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

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.10 contains two major new features: support for a native fixed-precision integer type and a new sort SProp of strict propositions. It is also the result of refinements and stabilization of previous features, deprecations or removals of deprecated features, cleanups of the internals of the system and API, and many documentation improvements. This release includes many user-visible changes, including deprecations that are documented in the next subsection, and new features that are documented in the reference manual.

Version 8.10 is the fifth release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.9. Vincent Laporte is the release manager and maintainer of this release. This release is the result of 2500 commits and 650 PRs merged, closing 150+ issues.

See the Zenodo citation for more information on this release: https://zenodo.org/record/3476303#.Xe54f5NKjOQ

NEWS OF THE YEAR: Coq 8.10.0 contains:

- some quality-of-life bug fixes, - a critical bug fix related to template polymorphism, - native 63-bit machine integers, - a new sort of definitionally proof-irrelevant propositions: SProp, - private universes for opaque polymorphic constants, - string notations and numeral notations, - a new simplex-based proof engine for the tactics lia, nia, lra and nra, - new introduction patterns for SSReflect, - a tactic to rewrite under binders: under, - easy input of non-ASCII symbols in CoqIDE, which now uses GTK3.

All details can be found in the user manual.

- Participants: Yves Bertot, Frédéric Besson, Maxime Denes, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Jason Gross, Hugo Herbelin, Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Michael Soegtrop, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Théo Zimmermann, Theo Winterhalter, Vincent Laporte, Arthur Charguéraud, Cyril Cohen, Christian Doczkal and Chantal Keller
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- 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 8.8 to 8.10, with special support for the Coq-HoTT library), 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, and it is able to solve unification problems without resorting to the K rule if not necessary.

Support for well-founded and mutual recursion using measure/well-foundedness annotations, even on indexed inductive types, using an 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.

RELEASE FUNCTIONAL DESCRIPTION: This version of Equations is based on an improved simplification engine for the dependent equalities appearing during dependent eliminations that is also usable as a separate dependent elimination tactic, providing an axiom-free variant of dependent destruction and a more powerful form of inversion. See http://mattam82.github.io/Coq-Equations/equations/2019/01/28/1.2beta.html and the following release notes for more information.

NEWS OF THE YEAR: Equations 1.2 was first released in may this year, after 3 years of development. It provides a refined simplification engine based on the work published at ICFP'19 (see the "Equations Reloaded" paper for details). The system has been improved to also work in the setting of Homotopy Type Theory and provides a more expressive source language and robust dependent elimination tactics.

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

## 5.3. Rewr

*Rewriting methods in algebra*

KEYWORDS: Computer algebra system (CAS) - Rewriting systems - Algebra

FUNCTIONAL DESCRIPTION: Rewr is a prototype of computer algebra system, using rewriting methods to compute resolutions and homotopical invariants of monoids. The library implements various classical constructions of rewriting theory (such as completion), improved by experimental features coming from Garside theory, and allows homotopical algebra computations based on Squier theory. Specific functionalities have been developed for usual classes of monoids, such as Artin monoids and plactic monoids.

NEWS OF THE YEAR: Rewr has been extended with the experimental KGB completion algorithm, based on Knuth-Bendix completion procedure improved by techniques coming from Garside theory.

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

## 5.4. Catex

KEYWORDS: LaTeX - String diagram - Algebra

FUNCTIONAL DESCRIPTION: Catex is a Latex package and an external tool to typeset string diagrams easily from their algebraic expression. Catex works similarly to Bibtex.

NEWS OF THE YEAR: It is now possible to add labels to objects and morphisms

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

## 5.5. Cox

KEYWORDS: Computer algebra system (CAS) - Rewriting systems - Algebra

FUNCTIONAL DESCRIPTION: Cox is a Python library for the computation of coherent presentations of Artin monoids, with experimental features to compute the lower dimensions of the Salvetti complex.

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

## 5.6. jsCoq

KEYWORDS: Coq - Program verification - Interactive - Formal concept analysis - Proof assistant - Ocaml - Education - JavaScript

FUNCTIONAL DESCRIPTION: jsCoq is an Online Integrated Development Environment for the Coq proof assistant and runs in your browser! It aims to enable new UI/interaction possibilities and to improve the accessibility of the Coq platform itself.

RELEASE FUNCTIONAL DESCRIPTION: - Coq 8.10 support - Much improved interaction and general experience - Open / Save dialogs - AST and full serialization of Coq's datatypes - NPM packaging - Timeout support

- Participant: Emilio Jesus Gallego Arias
- Partners: Mines ParisTech - Technion, Israel Institute of Technology
- Contact: Emilio Jesus Gallego Arias
- Publication: jsCoq: Towards Hybrid Theorem Proving Interfaces
- URL: https://github.com/ejgallego/jscoq

## 5.7. coq-serapi

KEYWORDS: Interaction - Coq - Ocaml - Data centric - User Interfaces - GUI (Graphical User Interface) - Toolkit

FUNCTIONAL DESCRIPTION: SerAPI is a library for machine-to-machine interaction with the Coq proof assistant, with particular emphasis on applications in IDEs, code analysis tools, and machine learning. SerAPI provides automatic serialization of Coq's internal OCaml datatypes from/to JSON or S-expressions (sexps).

RELEASE FUNCTIONAL DESCRIPTION: - Support Coq 8.10 - Serialization of extensive AST - Serialization of kernel structures - Support for kernel traces [dumping and replay] - Tokenization of Coq documents - Serialization to JSON - Improved protocol and printing - Bug fixes

- Participant: Karl Palmskog
- Partner: KTH Royal Institute of Technology
- Contact: Emilio Jesus Gallego Arias
- Publication: SerAPI: Machine-Friendly, Data-Centric Serialization for COQ : Technical Report
- URL: https://github.com/ejgallego/coq-serapi

# 6. New Results

## 6.1. Effects in proof theory and programming

**Participants:** Kostia Chardonnet, Emilio Jesús Gallego Arias, Hugo Herbelin, Yann Régis-Gianas, Alexis Saurin, Exequiel Rivas Gadda.

### *6.1.1. A theory of effects and resources*

In collaboration with Thomas Letan (ANSSI), Yann Régis-Gianas developed and proved several properties of a simple web server implemented in Coq using FreeSpec. This work will be presented at CPP 2020.

### *6.1.2. Call-by-need with probabilistic effects*

As a follow up of Chardonnet's Master 1 internship, Kostia Chardonnet and Alexis Saurin continued investigating call-by-need calculi extended with probabilistic choice and started preliminary discussions with Claudia Faggian.

### *6.1.3. Proof-search, algebraically and graphically*

Alexis Saurin worked on proof search in a proof-net scenario, that is proof-net search. A key aspect of proof construction is a management of non-determinism in bottom-up sequent-proof construction, be it when the search succeeds or when facing a failure and the need for backtracking. This is partially dealt with by focussing proof-construction, which reduces drastically the search space while retaining completeness of the resulting proof space (both at the provability level and at the denotational level).

His approach consists in viewing proof-search and sequentialisation as dual aspects of partial proof structures (that is proof nets with open premisses). In particular, he builds on Lafont's parsing criterion to obtain a proof-construction algorithm in which the proof space is not a search tree, as in sequent-calculus, but a dag allowing to share proof-construction paths.

Emilio Jesús Gallego Arias collaborates with Jim Lipton from Wesleyan University on the development of algebraic models for proof search.

## 6.2. Reasoning and programming with infinite data

**Participants:** Kostia Chardonnet, Lucien David, Abhishek De, Farzad Jafar-Rahmani, Luc Pellissier, Yann Régis-Gianas, Alexis Saurin.

This theme is part of the ANR project Rapido (see the National Initiatives section) which ended octobre 1st 2019.

### *6.2.1. Proof theory of non-wellfounded and circular proofs*

#### *6.2.1.1. Validity conditions of infinitary and circular proofs*

In collaboration with David Baelde, Amina Doumane and Denis Kuperberg, Alexis Saurin extended the proof theory of infinite and circular proofs for fixed-point logics in various directions by relaxing the validity condition necessary to distinguish sound proofs from invalid ones. The original validity condition considered by Baelde, Doumane and Saurin in CSL 2016 rules out lots of proofs which are computationally and semantically sound and does not account for the cut-axiom interaction in sequent proofs. In the setting of sequent calculus, Alexis Saurin studied together with David Baelde, Amina Doumane and Denis Kuperberg a relaxed validity condition to allow infinite branches to be supported by threads which may leave the infinite branch, visiting other parts of the proofs and bouncing on axioms and cuts. This allows for a much more flexible criterion, inspired from Girard's geometry of interaction. The most general form of this criterion does not ensure productivity in the sequent calculus due to a discrepancy between the sequential nature of proofs in sequent calculus and the parallel nature of threads. David Baelde, Amina Doumane, Denis Kuperberg and Alexis Saurin provided a slight restriction of the full bouncing validity which grants productivity and validity of the cut-elimination process. This restriction still strictly extends previous notions of validity and is actually expressive enough to be undecidable.

Several directions of research have therefore been investigated from that point:

- Decidability can be recovered by constraining the shapes of bounces (bounding the depth of bounces). They actually exhibited a hierarchy of criteria, all decidable and satisfying the fact that their union corresponds to bouncing validity (which is therefore semi-decidable)
- While the result originaly held only for the multiplicative fragment of linear logic, the result was extended to multiplicative and additive linear logic.

Those results are currently submitted.

#### *6.2.1.2. On the complexity of the validity condition of circular proofs*

Alexis Saurin, together with Rémi Nollet and Christine Tasson, characterised the complexity of deciding the validity of circular proofs. While deciding validity was known to be in PSPACE, they proved that, for $\mu MALL$ proof, it is in fact a PSPACE-complete problem.

The proof is based on a deeper exploration of the connection between thread-validity and the size-change termination principle, a standard tool to prove program termination.

This result has been presented and published at TABLEAUX 2019 [41].

#### *6.2.1.3. Proof nets for non-wellfounded proofs*

Abhishek De and Alexis Saurin set the basis of the theory of non-wellfounded and circular proofs nets (in the multiplicative setting). Non-wellfounded proof nets, aka infinets, were defined extending Curien's presentation of proof nets allowing for a smooth extension to fixed point logics. The aim of this work is to provide a notion of canonical proof obiects for circular proofs free from the irrelevant details of the syntax of the sequent calculus. The first results were published in TABLEAUX 2019 [38] and provide a correctness condition for an infinet to be sequentialisable in a sequent proof.

The results of the TABLEAUX paper are limited in that they only address the case of proofs with finitely many cuts inferences. Abhishek De and Alexis Saurin are currently investigating, with Luc Pellissier, the general case of infinitely many cut in order to then lift the results from straight thread validity to bouncing thread validity.

### *6.2.2. On the denotational semantics of non-wellfounded proofs*

Farzad Jafar-Rahmani started his PhD under the supervision of Thomas Ehrhard and Alexis Saurin in October 2019. His PhD work will focus on the denotational semantics of circular proofs of linear logic with fixed points. After working on the denotational semantics of finitary proofs for linear logic with fixed points (with Kozen rules) during his master, he is currently working at understanding the denotational counterpart of the validity condition of circular proofs.

### *6.2.3. Towards inductive and coinductive types in quantum programming languages*

Kostia Chardonnet started his PhD under the supervision of Alexis Saurin and Benoît Valiron in November 2019. Previously, he did his MPRI Master internship under their joint supervision on designing a calculus of reversible programs with inductive and coinductive types. His research focused on extending a languages of type isomorphisms with inductive and coinductive types and understanding the connections of those reversible programs with $\mu MALL$ type isomorphisms and more specifically with $\mu MALL$ focused circular proof isomorphisms. In his PhD, he shall extend this to the case of a quantum programming language with inductive and coinductive data types.

### *6.2.4. Theory of fixed points in the lambda-calculus*

The results of Alexis Saurin in collaboration with Giulio Manzonetto, Andrew Polonsky and Jacob Grue Simonsen, on two long-standing conjectures on fixed points in the $\lambda$-calculus – the "fixpoint property" and the "double-fixpoint conjecture" – have now appeared in the Journal of Logic and Computation [34]. The former asserts that every $\lambda$-term admits either a unique or an infinite number of $\beta$-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 $\beta$). 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.

## 6.3. Effective higher-dimensional algebra

**Participants:** Antoine Allioux, Pierre-Louis Curien, Alen Durić, Eric Finster, Yves Guiraud, Amar Hadzihasanović, Cédric Ho Thanh, Matthieu Sozeau.

### *6.3.1. Rewriting methods in higher algebra*

Yves Guiraud has completed a four-year collaboration with Eric Hoffbeck (Univ. Paris 13) and Philippe Malbos (Univ. Lyon 1), whose aim was to develop a theory of rewriting in associative algebras, with a view towards applications in homological algebra. They adapted the known notion of polygraph [64] 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 [63] and Poincaré-Birkhoff-Witt bases [100]. Then, they transposed the construction of [14], based on an extension of Squier's theorem [104] 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 was published in Mathematische Zeitschrift [32].

Yves Guiraud has written and defended his “Habilitation à diriger des recherches” manuscript, as a survey on rewriting methods in algebra based on Squier theory [26]. The defense was held in June 2019.

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

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

Building on [9], Yves Guiraud is currently finishing with Matthieu Picantin (Univ. Paris Diderot) a work that generalises already known constructions such as the bar resolution, several resolutions defined by Dehornoy and Lafont [73], and the main results of Gaussent, Guiraud and Malbos on coherent presentations of Artin monoids [11], 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.

Yves Guiraud has started a collaboration with Najib Idrissi (IMJ-PRG, Univ. Paris Diderot) whose aim is to understand the relation between several different methods known to compute small resolutions of algebras and operads: those based on rewriting methods (Anick, Squier) and those that stem from Koszul duality theory.

#### *6.3.2. Normalisation of monoids*

Alen Durić started his Phd thesis (supervised by Yves Guiraud and Pierre-Louis Curien) in October 2019. His work so far has been mostly bibliographical. The goal is to combine methods from rewriting theory (and in particular the method of homotopical completion and reduction developed by Guiraud-Malbos-Mimram) and methods developed by Dehornoy and his coauthors in the study of monoids with Garside families, and by Dehornoy-Guiraud in the study of normalisation for monoids. Alen Durić is currently experimenting with some examples taken from these latter works, with the goal of building coherent presentations for them using the former methods.

#### *6.3.3. Topological aspects of polygraphs*

Amar Hadzihasanović joined the team at the end of November 2019, as a one-year postdoc funded by FSMP. He has been working intensively on the study of shapes appropriate for the description of higher cells as needed in various approaches to higher categories and higher structures. Amar Hadzihasanović’s project is to recast his ideas in the framework of polygraphs, with the aim of bringing topological insights into the study of higher-dimensional rewriting.

### *6.3.4. Opetopes*

The work of Pierre-Louis Curien, Cédric Ho Thanh and Samuel Mimram on syntactic and type-theoretic presentations of opetopes and opetopic sets has been submitted to a journal, and a short version has been presented at the LICS conference in Vancouver this year [45].

Cédric Ho Thanh, in collaboration with Chaitanya Leena Subramaniam, has defined the notion of "opetopic algebras" that leverages the subtle combinatorics of opetopes. This framework encompasses categories, planar operads, and Loday's combinads over planar trees. They have defined an opetopic nerve functor that fully embeds each category of opetopic algebras into the category of opetopic sets. In particular, they obtain fully faithful opetopic nerve functors for categories and for planar coloured operads. These results have been written up in [51]. This work is the first in a series aimed at using opetopic spaces as models for higher algebraic structures. In particular, the aim is to provide new models for infinity-categories and infinity-operads.

### *6.3.5. Foundations and formalisation of higher algebra*

Antoine Allioux (PhD started in February 2018), Eric Finster, Yves Guiraud and Matthieu Sozeau are exploring the development of higher algebra in type theory. To formalise higher algebra, one needs a new source of coherent structure in type theory. During the first year of Allioux's PhD, they studied an internalisation of polynomial monads (of which opetopes and $\infty$-categories are instances) in type theory, which ought to provide such a coherent algebraic structure, inspired by the work of Kock et al [90]. They later realised that this internalisation is however incoherent as presented in pure type theory, essentially because of its reliance on equality types. Since then, they switched to a different view, describing opetopes as an external construction and relying on strict equalities in the metatheory to avoid the coherence problem. Opetopic type theory should then be, similarly to cubical type theory, a type theory indexed over these opetopic structures, where grafting and substitution are computional operations. They are now concentrating on showing that the modified inductive characterisation of opetopes and their algebras, still definable in type theory, gives rise to the standard notion of opetopes in mathematics, an original result in itself.

## 6.4. Incrementality

**Participants:** Thibaut Girka, Yann Régis-Gianas.

In collaboration with Paolo Giarrusso (EPFL, Switzerland), Philipp Shuster (Univ. of Tübingen, Germany), Yann Régis-Gianas developed a new method to incrementalise higher-order programs using formal derivatives and static caching. Yann Régis-Gianas has developed a mechanised proof for this transformation as well as a prototype language featuring efficient derivatives for functional programs. A paper has been presented at ESOP 2019 in Prague. Yann Régis-Gianas also presented this work at several places (Gallium seminar, Galinette seminar, and Chocola seminar).

In collaboration with Olivier Martinot (Paris Diderot), Yann Régis-Gianas studied a new technique to implement incrementalised operations on lists.

In collaboration with Faridath Akinotcho (Paris Diderot), Yann Régis-Gianas studied an incrementalisation of the Earley parsing algorithm.

## 6.5. Metatheory and development of Coq

**Participants:** Félix Castro, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Hugo Herbelin, Pierre Letouzey, Cyprien Mangin, Thierry Martinez, Yann Régis-Gianas, Matthieu Sozeau, Théo Winterhalter, Théo Zimmermann.

### *6.5.1. Meta-programming and Metatheory of Coq*

The MetaCoq project started last year, providing the means to program program transformations and general purpose plugins in Coq, using a quoting/unquoting mechanism. This year, they extended the framework to specify the theory, including the reduction, cumulativity and typing relations of the Polymorphic, Cumulative Calculus of Inductive Constructions at the basis of Coq. Matthieu Sozeau, together with Simon Boulier, Nicolas Tabareau and Théo Winterhalter at Galinette, Cyril Cohen at Marelle, Yannick Forster and Fabian Kunze at the University of Saarbrucken and Abhishek Anand and Gregory Malecha at BedRock Systems, Inc co-authored [54] a full description of the resulting theory (to appear in JAR). This allows for the verification of term manipulations with respect to typing: syntactic translations but also reflexive tactics glue code can hence be verified. The article also develops an alternative extraction mode to OCaml allowing the efficient compilation and execution of meta-programs written in the Template Monad. An example partial extraction of Coq programs to call-by-value pure lambda-calculus is developed this way.

Following up on this work, Matthieu Sozeau led a metatheoretical study of Coq in Coq, proving the basic metatheoretical properties of the typing relation, and developed together with Yannick Forster (Saarbrucken) and Simon Boulier, Nicolas Tabareau and Théo Winterhalter (Gallinette) verified correct versions of type-checking and erasure for a large subset of Coq. This work involved the production of a fully-precise specification for the type theory implemented by Coq, cleaning up the previously untested typing specification, and variants of the algorithms used in its kernel ammenable to proofs of correctness. The corresponding implementations can be extracted and provide an alternative, verified checker for Coq terms, that can run on medium-sized examples. This work will be presented [35] at POPL in New Orleans in January 2020.

### *6.5.2. Homotopy type theory*

Hugo Moeneclaey started in September 2019 a PhD on the syntax of spheres in homotopy type theory, under the supervision of Hugo Herbelin.

Hugo Herbelin and Hugo Moeneclaey worked on the syntax of a variant of Cohen, Coquand, Huber and Mörtberg's Cubical Type Theory justified by an iterated parametricity model where equality on types is defined to be equivalence of types, thus satisfying univalence by construction.

### *6.5.3. Computational contents of the axiom of choice*

Hugo Herbelin developed in collaboration with Nuria Brede (U. Potsdam) a unified logical structure for choice and bar induction principles.

### *6.5.4. Computational contents of Gödel's constructible universe*

Félix Castro started his PhD under the supervision of Hugo Herbelin and Alexandre Miquel in September 2019. His PhD work will focus on the computational contents of Gödel's constructible universe. Previously, he worked on the formalisation of the ramified analytical hierarchy in classical second-order arithmetic.

### *6.5.5. Dependent pattern-matching and recursion*

Together with Cyprien Mangin, Matthieu Sozeau refined the treatment of dependent pattern-matching in the Equations plugin. By carefully studying the type of equalities between indexed inductive types, he devised a new criterion for the elimination of equalities between inductive families based on the notion of forced arguments of constructors, resulting in a simplification of the setup of Cockx and Devriese [68] for simplification of dependent pattern-matching without K. This improved simplifier is part of the latest version of the Equations plugin, which also provides better support for the definition of mutual and well-founded recursive definitions on indexed inductive types. This work was presented at ICFP 2019 in Berlin [36]. A longer journal version is in preparation, along with a dedicated tutorial on Equations slated for inclusion in a new volume of the Software Foundations series dedicated to advanced tools.

Thierry Martinez continued part time 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.

### 6.5.6. Software engineering aspects of the development of Coq

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

Following the migration of the Coq bug tracker from Bugzilla to GitHub which he conducted in 2017, he analysed data (extracted through the GitHub API), in collaboration with Annalí Casanueva Artís from the Paris School of Economics. The results show an increased number of bugs by core developers and an increased diversity of the people commenting bug reports. These quantitative results were completed with qualitative data coming from interviews with main Coq developers, which help interpret them. They validate *a posteriori* the usefulness of such a switch. A paper [43] has been published at ICSME 2019, which is the leading conference on the topic of Software Maintenance and Evolution.

Besides, Théo Zimmermann also studied and influenced the pull-based model that is now used for the development of Coq, he improved the release management process and tools, he studied package distribution and maintenance, in particular with the foundation of the coq-community organisation in 2018, which has taken off by attracting 19 maintainers, and hosting 25 projects. All of these topics are presented in the PhD thesis [28] that he defended in December 2019.

Emilio J. Gallego Arias and Théo Zimmerman took the roles of release managers for the Coq 8.12 and will oversee this release, planned for mid-2020.

Emilio J. Gallego Arias and Théo Zimmerman discussed on future plans for compositional proof checking using the Dune build system, which will include a new library format for Coq. The Dune team was informed, with Emilio J. Gallego Arias participating in the bi-weekly developer meetings. Emilio J. Gallego Arias also started discussion with the Debian OCaml maintainers (who are located at IRIF) as to see how to better integrate Dune with the Debian packaging workflow.

Emilio J. Gallego Arias designed the Coq instrumentation used in the [103] paper, which collects and analyses changes to proof scripts.

Emilio J. Gallego Arias and Karl Palmskog released a new version of the Coq SerAPI tool, which has been used in some recent proof engineering efforts, such as [65], the machine-learning environments CoqGYM and Proverbot9001 [108], [55], offering state of the art proof automation after training with proof data sets, and the educational user interface WaterProof [56]. SerAPI has also been used in some other works undergoing review and thus yet not public.

Emilio J. Gallego Arias and Shachar Itzhaky released a new version of the educational Coq frontend jsCoq [10], and assisted a few users who have been preparing courses using it.

Emilio J. Gallego Arias maintains an ongoing collaboration with the Deducteam group at Inria Saclay on the topic of interactive proof methods and standards; this has resulted in the release of an experimental LSP server for the Lambdapi theorem prover.

Emilio J. Gallego Arias, Hugo Herbelin, and Théo Zimmerman participate in the Logipedia project led by Gilles Dowek, which aims to develop a standard proof interchange format.

### 6.5.7. Software Infrastructure

Emilio J. Gallego Arias did significant work to refactor the Coq codebase in preparation for further work on incremental and multi-core aware type checking.

### 6.5.8. Dissemination activities

Emilio J. Gallego Arias and Théo Zimmerman organised the Coq meetup, an after-work event targeting industry and other communities outside academia.

### 6.5.9. Coordination of the development of Coq

Hugo Herbelin, Matthieu Sozeau, Emilio J. Gallego Arias and Théo Zimmermann, helped by members from Gallinette (Nantes) and Marelle (Sophia-Antipolis), devoted an important part of their time to coordinate the development, to review propositions of extensions of Coq from external and/or young contributors, and to propose themselves extensions.

## 6.6. Formalisation and verification

**Participants:** Pierre-Louis Curien, Lucien David, Emilio Jesús Gallego Arias, Kailiang Ji, Pierre Letouzey, Jean-Jacques Lévy, Cyprien Mangin, Daniel de Rauglaudre, Yann Régis-Gianas, Alexis Saurin, Matthieu Sozeau.

### 6.6.1. Proofs and surfaces

The joint work of Pierre-Louis Curien with Jovana Obradović (former PhD student of the team and now postdoc in Prague), Zoran Petrić and other Serbian colleagues on formalising proofs of incidence theorems (arising by repeated use of Menelaus theorem) by means of a cyclic sequent calculus, has been submitted to a journal, and has been presented at the conference Topology, Algebra, and Categories in Logic (TACL) 2019, Nice, in June 2019 [53].

### 6.6.2. A Coq formalisation of the first-order predicate calculus

In relation with a logic course for master students, Pierre Letouzey made a Coq formalisation of the first-order predicate calculus. The logical rules are expressed in a natural deduction style (with explicit contexts). Pierre Letouzey proposed two low-level representations of formulas : one based on quantifiers with names, the other using “locally nameless” techniques. The equivalence between the two settings has been proved correct. Using this deep embedding, Pierre Letouzey formalised in Coq the whole course notes (prepared some years ago by Alexandre Miquel), including the completeness theorem for this logic. This development is available at https://gitlab.math.univ-paris-diderot.fr/letouzey/natded.

### 6.6.3. A Coq formalisation of circular proofs and their validity condition

During the summer 2019, Alexis Saurin supervised Lucien David’s M1 internship on formalizing in Coq circular proofs and their meta-theory. This work built on Xavier Onfroy’s previous work as well as on Pierre Letouzey’s formalisation of the predicate calculus in natural deduction mentioned above. While the previous work by Xavier Onfroy was both contributing to the proof theory part and the $\omega$-automata part (which is need for the decidability theorem), Lucien David completely focused on the the proof theory side. In particular, he was able to improve significantly on Xavier Onfroy’s formalisation by using ideas from Letouzey’s formalisation of natural deduction and by interacting with Pierre Letouzey and Alexis Saurin. This development is available at https://github.com/LuluDavid/CircularProofsValidity.

### 6.6.4. Lexing and regular expressions in Coq

Pierre Letouzey and Yann Régis-Gianas revisited in Coq classical techniques about lexing and regular expressions. In particular, regular expressions (with complement and conjunction) have been formalised, as well as their Brzozowski derivatives, and the finiteness theorem due to Brzozowski : a given regular expression admits only a finite number of derivatives (up to some equivalence). Both the general equivalence (based on language identity) and practical approximations (similarities) has been considered (and proved decidable). From that, the algorithms building recognizing automata (with derivatives as states) have been formalised and proved, leading to the minimal automata when using the general equivalence (but at a high cost), or to practical approximations of the minimal automata when using various similarities. This work is still ongoing. For instance, the correctness proof of a particular similarity used in an existing implementation (ml-ulex) is quite elusive for the moment. They also plan to extend this development up to a full-scale tool a la ocamllex in Coq.

#### *6.6.5. Real Numbers as sequences of digits in Coq*

Daniel de Rauglaudre has been continuing the formalisation of real numbers defined as sequences of digits in any radix with the LPO axiom/oracle (Limited Principle of Omniscience). Although the operations (additions and multiplications) work with this method, the proof of associativity of addition needs more work to be achieved. This development is available at https://github.com/roglo/coq_real/.

#### *6.6.6. Category theory in Coq*

Daniel de Rauglaudre started an implementation in Coq of Category theory in Coq, using in particular theorems coming from HOTT (HOmotopy Type theory) that he implemeted some years ago. Several notions around Categories have been defined. For example, Yoneda Lemma, among others. This development is available at https://github.com/roglo/mycoqhott/.

#### *6.6.7. Number theory in Coq*

Daniel de Rauglaudre started and almost completed the formalisation in Coq of the proof of Euler's Product Formula, stating that the Riemann zeta function, which is a sum on all the natural numbers, is also a product on all the prime numbers. He also added several theorems about the prime numbers. This development is available at https://github.com/roglo/coq_euler_prod_form.

#### *6.6.8. Proofs of algorithms on graphs*

Jean-Jacques Lévy and Chen Ran (a PhD student at the Institute of Software, Beijing) pursued their work about formal proofs of graph algorithms. Their goal is to provide proofs of algorithms checked by computer and human readable. In 2019, they presented at ITP 2019 a joint paper with Cyril Cohen, Stephan Merz and Laurent Théry on this work [37]. This article compared formal proofs in three different systems (Why3, Coq, Isabelle/HOL) of Tarjan (1972) linear-time algorithm computing the strongly connected components in directed graphs.

The current work is to have a proof of the implementation of this algorithm with imperative programming and memory pointers. They also planed to produce formal proofs of other abstract algorithms such as the Hopcroft-Tarjan (1972) linear-time algorithm for planarity testing in undirected graphs.

#### *6.6.9. Certified compilation and meta-programming*

Matthieu Sozeau participates to the CertiCoq project led by Andrew Appel at Princeton (https://www.cs.princeton.edu/~appel/certicoq) whose aim is to verify a compiler from Coq's Gallina language down to CompCert C-light which provides itself a certified compilation path to assembly language. Together with Yannick Forster at the University of Saarbrucken and the MetaCoq team, Matthieu Sozeau focused the verification of type-checking and erasure which were previously trusted parts of the system. The new verified erasure function fills a gap in the proof of correctness of compilation from Gallina terms down to C-light. The whole compiler can be run on realistic examples (the erasure phase does take most of the compilation time and should be optimised further).

In collaboration with Xavier Denis (Paris Diderot), Yann Régis-Gianas formalised and built a compiler for Mtac2. A paper is in preparation.

# 7. Bilateral Contracts and Grants with Industry

## 7.1. Bilateral Contracts with Industry

Theo Zimmermann will start a research engineer position in January 2020 to continue his research and development work about improving the Software Engineering practices of the development of Coq, especially to continue the improvement of the collaborative development processes and of its ecosystem. This position is funded by the Inria-NomadicLabs grant.

# 8. Partnerships and Cooperations

## 8.1. National Initiatives

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

Pierre-Louis Curien, Yves Guiraud (local coordinator until Sept. 2019) and Matthieu Sozeau are members of the GDR Topologie Algébrique, federating French researchers working on classical topics of algebraic topology and homological algebra, such as homotopy theory, group homology, K-theory, deformation theory, and on more recent interactions of topology with other themes, such as higher categories and theoretical computer science.

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

Yves Guiraud will coordinate the four-year Action Exploratoire Inria Réal (Réécriture Algébrique), starting in January 2020. Its aim is to continue the unification of rewriting-like methods in abtract and higher algebra, with a view toward applications in homological and higher algebra, and group and representation theory. This investigation is pursued in immersion at IMJ-PRG, the fundamental maths common laboratory of Sorbonne Université and Université Paris Diderot.

Emilio J. Gallego Arias is a member of the GDR Génie de la Programation et du Logiciel, in the LTP (Langages, Types et Preuves) group.

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

Yann Régis-Gianas collaborates with ANSSI on the topic of certified ful programming in Coq.

Yann Régis-Gianas collaborates with Nomadic Labs on the topic of certified smart contract compilation.

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

Yann Régis-Gianas and Alexis Saurin (coordinator) are members of the four-year RAPIDO ANR project, started in January 2015 and ended in September 2019. RAPIDO aims at investigating the use of proof-theoretical methods to reason and program on infinite data objects. The goal of the project is to develop logical systems capturing infinite proofs (proof systems with least and greatest fixpoints as well as infinitary proof systems), to design and to study programming languages for manipulating infinite data such as streams both from a syntactical and semantical point of view. Moreover, the ambition of the project is to apply the fundamental results obtained from the proof-theoretical investigations (i) to the development of software tools dedicated to the reasoning about programs computing on infinite data, *e.g.* stream programs (more generally coinductive programs), and (ii) to the study of properties of automata on infinite words and trees from a proof-theoretical perspective with an eye towards model-checking problems. Other permanent members of the project are Christine Tasson from IRIF (PPS team), David Baelde from LSV, ENS-Cachan, and Pierre Clairambault, Damien Pous and Colin Riba from LIP, ENS-Lyon.

Matthieu Sozeau is a member of the CoqHoTT project led by Nicolas Tabareau (Gallinette team, Inria Nantes & École des Mines de Nantes), funded by an ERC Starting Grant, ending in 2020. The PhD grant of Antoine Allioux is funded by the CoqHoTT ERC.

## 8.2. European Initiatives

### *8.2.1. Collaborations in European Programs, Except FP7 & H2020*

- Program: COST

- Project acronym: EUTypes
- Project title: The European research network on types for programming and verification
- Duration: March 2016 - March 2020
- Coordinator: Herman Geuvers
- Other partners: 29 countries
- Abstract: This COST promotes (1) the synergy between theoretical computer scientists, logicians and mathematicians to develop new foundations for 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.

## 8.3. International Initiatives

### *8.3.1. Inria Associate Teams Not Involved in an Inria International Labs*

Pierre-Louis Curien and Claudia Faggian are members of the CRECOGI associate team, coordinated on one side by Ugo dal Lago (research-team FoCUS, Inria Sophia and Bologna), and on the other side by Ichiro Hasuo (NII, Tokyo). The full name of the project is Concurrent, Resourceful and full Computation, by Geometry of Interaction. This project was renewed in 2019 for a duration of two years.

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

### *8.3.2. Inria International Partners*

#### *8.3.2.1. Participation in International Programs*

Pierre-Louis Curien and Alexis Saurin are members of CNRS GDRI-LL a french-italian network on linear logic community in France and Italy.

#### *8.3.2.2. International Initiatives*

Pierre-Louis Curien is principal investigator on the French side for a joint project Inria - Chinese Academy of Sciences. The project's title is "Verification, Interaction, and Proofs" (December 2017 – December 2020). 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).

## 8.4. International Research Visitors

### *8.4.1. Research Stays Abroad*

Matthieu Sozeau visited the Programming Languages group of Benjamin Pierce at the University of Pennsylvania in June and July 2019, along with visits at MIT and Princeton to other members of the NSF DeepSpec project.

Pierre-Louis Curien visited East China Normal University (ECNU), Shanghai, for a month from early October to early December 2019 as invited professor.

# 9. Dissemination

## 9.1. Promoting Scientific Activities

### *9.1.1. Scientific Events: Organisation*

#### *9.1.1.1. Member of the Organizing Committees*

Alexis Saurin, as coordinator of GT-Scalp is member of the organizing committee of the annual meeting of the working group. In 2019, the annual meeting took place in ENS Lyon in October 2019: https://www.irif.fr/gt-scalp/journees-2019.

#### *9.1.1.2. Member of the Steering Committees*

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

### *9.1.2. Scientific Events: Selection*

#### *9.1.2.1. Member of the Conference Program Committees*

Pierre-Louis Curien is member of the program committee of Formal Structures for Computation and Deduction 2020, Paris, June-July 2020.

Yann Régis-Gianas was member of the program committee of Software Language Engineering 2019.

Alexis Saurin was a member of the program committee of Circularity in Syntax and Semantics, held in Gothenbug in November 2019.

### *9.1.3. Journal*

#### *9.1.3.1. 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 a guest editor of the special issue of MSCS on structural proof theory, automated reasoning and computation in celebration of Dale Miller's 60th birthday published in September 2019.

#### *9.1.3.2. Reviewer - Reviewing Activities*

Pierre Letouzey has done several reviews for the Journal of Functional Programming (JFP) and for Science of Computer Programming. Yann Régis-Gianas reviewed a paper for Transactions on Programming Languages and Systems (TOPLAS).

### *9.1.4. Invited Talks*

Hugo Herbelin gave an invited talk on the logical structure of choice and bar induction principles at the Proof, Computation and Complexity workshop (Mittag-Leffler institute). Yann Régis-Gianas gave an invited talk at the DeepSpec workshop of PLDI 2019 about FreeSpec.

### *9.1.5. Conferences and schools attended*

Pierre-Louis Curien presented his joint work on Proofs and surfaces at the conference Topology, Algebra, and Categories in Logic (TACL) 2019, Nice, in June 2019.

Pierre-Louis Curien and Yves Guiraud attended the workshop Homotopy meets homology, Dublin, May 2019.

Pierre-Louis Curien and Yves Guiraud attended the *Higher Structures* conference, Marseille, January 2019.

Pierre-Louis Curien and Cédric Ho Thanh attended the Category Theory 2019 conference, Edinburgh, July 2019.

Pierre-Louis Curien attended Martin Hofmann Memorial Meeting, Munich, July 2019.

Abhishek De attended the Proof Society summer school at Swansea in September 2019.

Cédric Ho Thanh attended the Summer school on Topology, Algebra, and Categories in Logic at Ile de Porquerolles in June 2019.

Hugo Herbelin and Hugo Moeneclaey attended the Types Conference, Oslo, June 2019.

Hugo Moeneclaey attended the International Conference on Homotopy Type Theory (HoTT 2019) and the associated Homotopy Type Theory Summer School, Pittsburgh, August 2019.

Hugo Herbelin attended the Third Workshop on Mathematical Logic and its Applications, Nancy, March 2019.

Hugo Herbelin attended the Workshop Facets of Realizability, Cachan, July 2019.

### *9.1.6. Leadership within the Scientific Community*

#### *9.1.6.1. Scientific Expertise*

Hugo Herbelin was a member of the HCERES committee evaluating the MICS laboratory of Centrale Supelec.

#### *9.1.6.2. Research Administration*

Pierre-Louis Curien has been a member of the Scientific Council of the CIRM (Centre International de Rencontres Mathématiques) until February 2019.

Pierre-Louis Curien was a member of the scientific council of the Computer Science department of University Paris Diderot until May 2019.

Yves Guiraud was the coordinator of the Proofs, Programs and Systems (PPS) pole at IRIF, and a member of the IRIF comité de direction, until August 2019. Since September 2019, Hugo Herbelin coordinates the PPS pole.

Yves Guiraud was a member of the Conseil Scientifique of the CS department of Univ. Paris Diderot until June 2019.

Yves Guiraud is a member of the Conseil Scientifique of the Math department of Univ. Paris Diderot since April 2019.

Alexis Saurin is a coordinator of the Scalp working group of GDR Informatique Mathématique: https://www.irif.fr/gt-scalp/index

Alexis Saurin is member of the “commission recherche” of both Faculté des Sciences and Faculté Société et humanités at Université de Paris.

Alexis Saurin and Hugo Herbelin are members of the IRIF conseil de laboratoire.

## 9.2. Teaching - Supervision - Juries

### *9.2.1. Teaching*

- Master: Matthieu Sozeau, Assistants de Preuve, 18h ETD, MPRI, University Paris Diderot.
- Master: Pierre-Louis Curien, Models of programming languages: domains, categories, games (together with Thomas Ehrhard and Paul-André Melliès), 15h, MPRI, University Paris Diderot.
- Master level: Pierre-Louis Curien taught a course on the Foundations of Programming Languages at East China Normal University (4 hours, November-December 2019).
- Master: Hugo Herbelin taught a class on Homotopy Type Theory (together with Nicolas Tabareau), 24h, LMFI, University Paris Diderot.

- Master: Pierre Letousey teaches two short courses to the LMFI Master 2 students : “Programming in Coq” and “Introduction to computed-aided formal proofs”. These two courses come in addition to Pierre Letousey’s regular duty as teacher in the Computer Science department of University Paris Diderot (including a course on Compilation to M2-Pro students and a course on computed-aided formal proofs to M1 students).
- Master: Yann Régis-Gianas, Functional programming and Type Systems, 12h, MPRI (in addition to four other courses at the master level in University Paris Diderot).
- Master: Alexis Saurin, Outils classiques pour la correspondance de Curry-Howard, 24H, LMFI.
- Summer school: Pierre-Louis Curien gave an introductory mini-course on Homotopy type theory at the Fifteenth International Tbilisi Summer School in Logic and Language, Georgia, in September 2019 (3h).
- Summer school: Hugo Herbelin gave a lecture on type theory at the EUTypes Summer School, Ohrid, in September 2019 (4h).
- Yann Régis-Gianas organised the 4th session of the OCaml MOOC on the FUN platform.
- Yann Régis-Gianas organised the “Journée Francilienne de Programmation”, a programming contest between the first years students of University Paris Sorbonne, University Paris Saclay and University Paris Diderot.

### *9.2.2. Supervision*

#### *9.2.2.1. PhD Supervision*

- PhD started: Félix Castro, Computational contents of the model of constructible sets, cotutelle between Université Paris Diderot and Universidad de la República (Montevideo, Uruguay), started in September 2019, supervised by Hugo Herbelin and Alexandre Miquel.
- PhD Started: Kostia Chardonnet, Inductive and coinductive types in quantum programming languages, Université Paris Saclay, started in November 2019, supervised by Alexis Saurin and Benoît Valiron.
- PhD started: Alen Durić, Normalisation for monoids and higher categories, Université Paris Diderot, started in October 2O19, supervised by Yves Guiraud and Pierre-Louis Curien.
- PhD started: Colin Gonzalez, Certified compilation of spreadsheets as Tezos smart contract, CIFRE, started in September 2019, supervised by Yann Régis-Gianas and Benjamin Canou (Nomadic Labs).
- PhD started: Farzad Jafar-Rahmani, Denotational semantics of circular and non-well-founded proofs, Université Paris Diderot, started in October 2019, supervised by Thomas Ehrhard and Alexis Saurin.
- PhD started: Hugo Moeneclaey, Syntax of spheres in homotopy type theory, Université Paris Diderot, started in September 2019, supervised by Hugo Herbelin.
- PhD in progress: Antoine Allioux, Opetopes in Type Theory, Université Paris Diderot, since March 2018, supervised by Yves Guiraud and Matthieu Soseau.
- PhD in progress: Abhishek De, Proof-nets for fixed-point logics and non-well-founded proofs, Université Paris Diderot, since October 2018, supervised by Alexis Saurin.
- PhD in progress: Cédric Ho Thanh, Opetopes for higher-dimensional rewriting and koszulity, Université Paris Diderot, since September 2017, supervised by Pierre-Louis Curien and Samuel Mimram.
- PhD in progress: Lucas Massoni Sguerra, Formal verification of mechanism design, Université PSL, since January 2018, supervised by Gérard Memmi, Fabien Coehlo, and Emilio J. Gallego Arias.
- PhD in progress: Rémi Nollet, Validity conditions for circular proofs, Université Paris Diderot, since October 2016, supervised by Alexis Saurin and Christine Tasson.
- PhD in progress: Théo Winterhalter, Metatheory and certified implementation of type theories, Université de Nantes, since September 2017, supervised by Nicolas Tabareau and Matthieu Sozeau.

- PhD ended: Gaëtan Gilbert, A type theory with definitional proof-irrelevance, École Nationale Supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire - IMT Atlantique, defended on 20 December 2019, supervised by Nicolas Tabareau and Matthieu Sozeau.
- PhD ended: Théo Zimmermann, Challenges in the collaborative evolution of a proof language and its ecosystem, Université Paris Diderot, defended on 12 December 2019, supervised by Hugo Herbelin and Yann Régis-Gianas.

##### *9.2.2.2. Internships*

- Hugo Herbelin supervised the master internship of Adrien Champougny on a type-theoretic presentation of Primitive Recursive Arithmetic.
- Hugo Herbelin supervised the master internship of Hugo Moeneclaey on a syntax and semantics for cubical type theory with definitional univalence.
- Yann Régis-Gianas supervised the third-year licence internship of Alexandre Moine about the detection of clones in OCaml programs.
- Yann Régis-Gianas supervised the first-year master internship of Astyax Nourel and Sébastien Lecleire about the implementation of an instance server for Learn-OCaml.
- Yann Régis-Gianas cosupervised with Thomas Letan (ANSSI) the master internship of Vincent Tourneur about the certification of a Unix utility using FreeSpec.
- Alexis Saurin supervised the first-year master internship of Lucien David about Coq formalisation of circular proof systems.
- Alexis Saurin cosupervised with Benoît Valiron the MPRI internship of Kostia Chardonnet on (co)inductive types in reversible programming languages.

### *9.2.3. Juries*

- Hugo Herbelin was reviewer of the PhD of Ulysse Gérard.
- Hugo Herbelin was president of the PhD committee of Kenji Maillard.
- Alexis Saurin was member of the jury of the entrance competition to Écoles normales supérieures, in charge of the computer science exam (informatique-mathématiques), for the last three years.

## 9.3. Popularisation

### *9.3.1. Internal or external Inria responsibilities*

- Yves Guiraud is “correspondant communication” for IMJ-PRG at INSMI.
- Yann Régis-Gianas is “correspondant communication” for IRIF at INS2I.
- Jean-Jacques Lévy is member of the Inria-Alumni’s executive committee (4 meetings in 2019).

### *9.3.2. Education*

- Yann Régis-Gianas presented the Learn-OCaml project at the OCaml workshop, at the OCaml Meetup, at the University of Lisbon and at the JFLA.
- Yann Régis-Gianas gave a talk at the conference of CPGE teachers of computer science to present the Learn-OCaml project.

### *9.3.3. Interventions*

- Jean-Jacques Lévy was invited by the French Academy of Sciences to participate to the Nanjing Tech Week (June 22-24) and to visit the new Qingdao International Academician Park (June 28). He also visited the Institute of Software of the Chinese Academy of Sciences (ISCAS) on July 1-3.
- Jean-Jacques Lévy as a member of the AEFC (Association des Experts France Chine) was invited to the 2019 Hangzhou International Human Resources Exchanges and Cooperation Conference (Hangzhou, November 9-11). He also visited the East China Normal University (ECNU) Shanghai (November 12-15) and Jinan University at Guangzhou (November 18-19).
- Jean-Jacques Lévy gave a talk about the Four Ages of Computer Science at the Huashang College in Guangzhou (November 18).

### *9.3.4. Internal action*

Jean-Jacques Lévy talked about “L’informatique en 4 temps” at the Café des Sciences at Inria Rocquencourt (February 12) and to the Journées Européennes du Patrimoine at Inria-Rocquencourt (21 September).

# 10. Bibliography

## Major publications by the team in recent years

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

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

[3] 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, C. BAI, L. GUO, 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] E. J. GALLEGO ARIAS, B. PIN, P. JOUVELOT. *jsCoq: Towards Hybrid Theorem Proving Interfaces*, in "Proceedings of the 12th Workshop on User Interfaces for Theorem Provers, Coimbra, Portugal, 2nd July 2016", S. AUTEXIER, P. QUARESMA (editors), Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, 2017, vol. 239, pp. 15-27, http://dx.doi.org/10.4204/EPTCS.239.2

[11] 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

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

[13] T. Girka, D. Mentré, Y. Régis-Gianas. *Oracle-based Dierential Operational Semantics (long version)*, Université Paris Diderot / Sorbonne Paris Cité, October 2016, https://hal.inria.fr/hal-01419860

[14] Y. Guiraud, P. Malbos. *Higher-dimensional normalisation strategies for acyclicity*, in "Advances in Mathematics", 2012, vol. 231, n$^o$ 3-4, pp. 2294-2351 [*DOI :* 10.1016/j.aim.2012.05.010], https://hal.archives-ouvertes.fr/hal-00531242

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

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

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

[18] H. Herbelin. *A Constructive Proof of Dependent Choice, Compatible with Classical Logic*, in "LICS 2012 - 27th Annual ACM/IEEE Symposium on Logic in Computer Science", Dubrovnik, Croatia, Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012, 25-28 June 2012, Dubrovnik, Croatia, IEEE Computer Society, June 2012, pp. 365-374, https://hal.inria.fr/hal-00697240

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

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

[21] G. Munch-Maccagnoni. *Focalisation and Classical Realisability*, in "Computer Science Logic '09", E. Grädel, R. Kahle (editors), Lecture Notes in Computer Science, Springer-Verlag, 2009, vol. 5771, pp. 409–423

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

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

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

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

## Publications of the year

### Doctoral Dissertations and Habilitation Theses

[26] Y. GUIRAUD. *Rewriting methods in higher algebra*, Université Paris 7, June 2019, Habilitation à diriger des recherches, https://hal.archives-ouvertes.fr/tel-02161197

[27] Y. RÉGIS-GIANAS. *About some Metamorphoses of Computer Programs*, Université Paris Diderot, November 2019, Habilitation à diriger des recherches, https://hal.inria.fr/tel-02405839

[28] T. ZIMMERMANN. *Challenges in the collaborative evolution of a proof language and its ecosystem*, Université de Paris, December 2019, https://hal.inria.fr/tel-02451322

### Articles in International Peer-Reviewed Journals

[29] C. CHENAVIER. *A Lattice Formulation of the F 4 Completion Procedure*, in "International Journal of Algebra and Computation", 2019, https://hal.archives-ouvertes.fr/hal-01489200

[30] C. CHENAVIER. *Syzygies among reduction operators*, in "Journal of Pure and Applied Algebra", 2019, https://arxiv.org/abs/1708.08709 , https://hal.archives-ouvertes.fr/hal-01578555

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

[32] Y. GUIRAUD, E. HOFFBECK, P. MALBOS. *Convergent presentations and polygraphic resolutions of associative algebras*, in "Mathematische Zeitschrift", 2019, vol. 293, n$^{o}$ 1-2, pp. 113-179, https://arxiv.org/abs/1406.0815 [*DOI :* 10.1007/s00209-018-2185-z], https://hal.archives-ouvertes.fr/hal-01006220

[33] Y. JIANG, J. LIU, G. DOWEK, K. JI. *Towards Combining Model Checking and Proof Checking*, in "The Computer Journal", 2019, forthcoming [*DOI :* 10.1093/COMJNL/BXY112], https://hal.inria.fr/hal-01970274

[34] G. MANZONETTO, A. POLONSKY, A. SAURIN, J. G. SIMONSEN. *The fixed point property and a technique to harness double fixed point combinators*, in "Journal of Logic and Computation", September 2019, vol. 29, n$^{o}$ 5, pp. 831-880 [*DOI :* 10.1093/LOGCOM/EXZ013], https://hal.archives-ouvertes.fr/hal-02408192

[35] M. SOZEAU, S. BOULIER, Y. FORSTER, N. TABAREAU, T. WINTERHALTER. *Coq Coq Correct! Verification of Type Checking and Erasure for Coq, in Coq*, in "Proceedings of the ACM on Programming Languages", January 2020, pp. 1-28 [*DOI :* 10.1145/3371076], https://hal.archives-ouvertes.fr/hal-02380196

[36] M. SOZEAU, C. MANGIN. *Equations reloaded*, in "Proceedings of the ACM on Programming Languages", July 2019, vol. 3, n$^{o}$ ICFP, pp. 1-29 [*DOI :* 10.1145/3341690], https://hal.inria.fr/hal-01671777

### International Conferences with Proceedings

[37] R. CHEN, C. COHEN, J.-J. LEVY, S. MERZ, L. THÉRY. *Formal Proofs of Tarjan's Strongly Connected Components Algorithm in Why3, Coq and Isabelle*, in "ITP 2019 - 10th International Conference on Interactive Theorem Proving", Portland, United States, J. HARRISON, J. O'LEARY, A. TOLMACH (editors), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, vol. 141, pp. 13:1 - 13:19 [*DOI :* 10.4230/LIPICS.ITP.2019.13], https://hal.inria.fr/hal-02303987

[38] A. DE, A. SAURIN. *Infinets: The parallel syntax for non-wellfounded proof-theory*, in "TABLEAUX 2019 - 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods", London, United Kingdom, TABLEAUX 2019 - 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, September 2019, https://hal.archives-ouvertes.fr/hal-02337286

[39] P. G. GIARRUSSO, Y. RÉGIS-GIANAS, P. SCHUSTER. *Incremental $\lambda$-Calculus in Cache-Transfer Style Static Memoization by Program Transformation*, in "ESOP 2019 - European Symposium on Programming", Prague, Czech Republic, L. CAIRES (editor), Springer, April 2019, https://hal.inria.fr/hal-02405864

[40] T. LETAN, Y. RÉGIS-GIANAS. *FreeSpec: Specifying, Verifying and Executing Impure Computations in Coq*, in "CPP 2020 - 9th ACM SIGPLAN International Conference on Certified Programs and Proofs", Nouvelle-Orléans, United States, ACM, January 2020, pp. 1-15 [*DOI :* 10.1145/3372885.3373812], https://hal.inria.fr/hal-02422273

[41] R. NOLLET, A. SAURIN, C. TASSON. *PSPACE-Completeness of a Thread Criterion for Cyclic Proofs in Linear Logic with Least and Greatest Fixed Points*, in "TABLEAUX 2019 - 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods", London, United Kingdom, Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings, September 2019, https://hal.archives-ouvertes.fr/hal-02173207

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

[43] T. ZIMMERMANN, A. CASANUEVA ARTÍS. *Impact of switching bug trackers: a case study on a medium-sized open source project*, in "ICSME 2019 - International Conference on Software Maintenance and Evolution", Cleveland, United States, September 2019, https://hal.inria.fr/hal-01951176

### National Conferences with Proceedings

[44] C. BOZMAN, B. CANOU, R. DI COSMO, P. COUDERC, L. GESBERT, G. HENRY, F. LE FESSANT, M. MAUNY, C. MOREL, L. PEYROT, Y. RÉGIS-GIANAS. *Learn-OCaml : un assistant à l'enseignement d'OCaml*, in "JFLA 2019 - Journées Francophones des Langages Applicatifs", Les Rousses, France, January 2019, https://hal.inria.fr/hal-02405876

### Conferences without Proceedings

[45] C. HO THANH, P.-L. CURIEN, S. MIMRAM. *A Sequent Calculus for Opetopes*, in "LICS 2019 - Logic in computer science 2019", Vancouver, Canada, June 2019, https://hal.archives-ouvertes.fr/hal-02406569

### Scientific Books (or Scientific Book chapters)

[46] D. BAELDE, A. FELTY, G. NADATHUR, A. SAURIN. *A special issue on structural proof theory, automated reasoning and computation in celebration of Dale Miller's 60th birthday*, Cambridge University Press, September 2019, vol. 29, n$^{o}$ 8, pp. 1007-1008 [*DOI :* 10.1017/S0960129519000136], https://hal.archives-ouvertes.fr/hal-02408211

[47] Z. L. DARGAYE, Y. RÉGIS-GIANAS. *31ème Journées Francophones des Langages Applicatifs*, IRIF, January 2020, https://hal.inria.fr/hal-02427360

### Research Reports

[48] N. JEANNEROD, C. MARCHÉ, Y. RÉGIS-GIANAS, M. SIGHIREANU, R. TREINEN. *Specification of UNIX Utilities*, ANR, October 2019, https://hal.inria.fr/hal-02321691

### Other Publications

[49] P.-L. CURIEN, C. HO THANH, S. MIMRAM. *Syntactical approaches to opetopes*, March 2019, working paper or preprint, https://hal.archives-ouvertes.fr/hal-02064784

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

[51] C. HO THANH, C. LEENA SUBRAMANIAM. *Opetopic algebras I: Algebraic structures on opetopic sets*, November 2019, 38 pages, https://hal.archives-ouvertes.fr/hal-02343861

[52] C. HO THANH, C. LEENA SUBRAMANIAM. *Opetopic algebras III: Presheaf models of homotopy-coherent opetopic algebras*, January 2020, working paper or preprint, https://hal.archives-ouvertes.fr/hal-02448208

[53] M. MILICEVIC, D. BARALIC, J. OBRADOVIC, Z. PETRIC, M. ZEKIC, R. ZIVALJEVIC, P.-L. CURIEN. *Proofs and surfaces*, December 2019, working paper or preprint, https://hal.archives-ouvertes.fr/hal-02410910

[54] M. SOZEAU, A. ANAND, S. BOULIER, C. COHEN, Y. FORSTER, F. KUNZE, G. MALECHA, N. TABAREAU, T. WINTERHALTER. *The MetaCoq Project*, June 2019, working paper or preprint, https://hal.inria.fr/hal-02167423

## References in notes

[55] *Proverbot, a bot for proving*, 2019, Accessed: 2019-12-09, https://github.com/UCSD-PL/proverbot9001

[56] *Waterproof: an educational environment for writing mathematical proofs in interactive notebooks*, 2019, Accessed: 2019-12-09, https://github.com/impermeable/waterproof

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

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

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

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

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

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

[63] 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

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

[65] A. CELIK, K. PALMSKOG, M. PAROVIC, E. J. GALLEGO ARIAS, M. GLIGORIC. MCOQ*: Mutation Proving for Analysis of Verification Projects*, in "Proceedings of the 34rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2019", 2019

[66] 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

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

[68] J. COCKX, D. DEVRIESE. *Proof-relevant unification: Dependent pattern matching with only the axioms of your type theory*, in "J. Funct. Program.", 2018, vol. 28, e12 p. , https://doi.org/10.1017/S095679681800014X

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

[70] 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

[71] 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

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

[73] 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

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

[75] 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

[76] 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

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

[78] 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

[79] 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

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

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

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

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

[84] 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

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

[86] 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

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

[88] 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

[89] 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

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

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

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

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

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

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

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

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

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

[99] 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

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

[101] 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

[102] 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

[103] T. RINGER, A. SANCHEZ-STERN, D. GROSSMAN, S. LERNER. *REPLICA: REPL Analysis for Coq Instrumentation*, in "Certified Programs and Proofs (CPP 2019)", 2019

[104] 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

[105] 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

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

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

[108] K. YANG, J. DENG. *Learning to Prove Theorems via Interacting with Proof Assistants*, in "International Conference on Machine Learning", 2019

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