RESEARCH CENTRE

Inria Paris Centre

IN PARTNERSHIP WITH:

CNRS, Université Paris Cité

2023

ACTIVITY REPORT

Project-Team

PICUBE

Proof assistants at the heart of mathematical reasoning

IN COLLABORATION WITH: Institut de Recherche en Informatique Fondamentale

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Inria

Contents

Project-Team PICUBE

Creation of the Project-Team: 2022 December 01

Keywords

Computer sciences and digital sciences

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

A9.2. – Machine learning

A9.4. – Natural language processing

A9.8. – Reasoning

Other research topics and application domains

B6.1. – Software industry

1 Team members, visitors, external collaborators

Research Scientists

- Paul-Andre Mellies [Team leader, CNRS, Senior Researcher]
- Guillaume Baudart [Inria, Researcher, from Oct 2023]
- Pierre-Louis Curien [CNRS, Emeritus, HDR]
- Thomas Ehrhard [CNRS, Senior Researcher]
- Emilio Jesus Gallego Arias [INRIA, Starting Research Position]
- Hugo Herbelin [INRIA, Senior Researcher, HDR]
- Jean-Jacques Lévy [IRIF, Emeritus, HDR]
- Daniela Petrisan [UNIV PARIS, Researcher]
- Alexis Saurin [CNRS, Researcher]
- Gabriel Scherer [Inria, Researcher, from Oct 2023]

Faculty Members

- Pierre Letouzey [UNIV PARIS - CITE, Associate Professor Delegation, from Sep 2023]
- Pierre Letouzey [UNIV PARIS, Associate Professor, until Aug 2023]

Post-Doctoral Fellow

- Alen Duric [FSMP, Post-Doctoral Fellow, from Nov 2023]

PhD Students

- Esaïe Bauer [UNIV PARIS]
- Thomas Binetruy [EQUISAFE]
- Vincent Blazy [UNIV PARIS, from Oct 2023]
- Felix Castro [UNIV PARIS, until Sep 2023]
- El Mehdi Cherradi [CGEIET, until Aug 2023]
- Moana Jubert [INRIA]
- Vincent Moreau [UNIV PARIS]
- David Reboullet [UNIV PARIS - CITE, from Feb 2023 until Feb 2023]

Technical Staff

- Ramkumar Ramachandra [INRIA, Engineer, until Mar 2023]

Administrative Assistants

- Meriem Guemair [INRIA]
- Diana Marino Duarte [INRIA, from Mar 2023]

External Collaborator

- Theo Zimmermann [INRIA, until Feb 2023]

2 Overall objectives

The Picube team is a joint project-team of INRIA, Université Paris-Cité and CNRS, within the IRIF's Proofs, Programs and Systems pole. It covers five main research themes:

1. the fundamental structures of logic and mathematical reasoning ;
2. differential and probabilistic tools for programming, reasoning and learning;
3. architecture and design of a proof assistant for mathematicians;
4. formalization and linguistics of mathematics;
5. higher-dimensional algebra and homotopic type theory.

The Picube team wishes to take advantage of recent advances in the fields of:

- type theory and the foundations of mathematics: homotopic type theory, realizability and forcing, differential linear logic,
- semantics of programming languages: computational effects, differential and probabilistic programming,
- architecture and design of proof assistants: mathematical formalization, unification and symbolic elaboration techniques.

in order to reduce the gap between the vernacular language currently used by mathematicians in their daily practice and the formal language used today in proof assistants such as Coq, Agda or Lean.

The research project builds on the knowledge and expertise accumulated in the Pi.R2 team and integrates new ingredients in the direction of certified mathematics, differential and probabilistic programming and learning, with a view to tackling the above themes.

3 Research program

Participants: Guillaume Baudart, Pierre-Louis Curien, Abhishek De, Alen Duric, Thomas Ehrhard, Emilio Jesus Gallego Arias, Hugo Herbelin, Farzad Jafar-Rahmani, Pierre Letouzey, Paul-Andre Mellies, Vincent Moreau, Alexis Saurin, Gabriel Scherer.

We describe the contributions in each of the five research directions of the team.

3.1 The fundamental structures of logic and mathematical reasoning

Proof theory is the branch of logic devoted to the study of the structure of proofs. An essential contributor to this field is Gentzen [50] 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 [44], then by Howard and de Bruijn at the end of the 60's [61, 32], that proofs had the very same structure as programs: for instance, natural deduction proofs can be identified as typed programs of the ideal programming language known as λ-calculus. This proofs-as-programs

correspondence has been the starting point to a large spectrum of researches and results contributing to deeply connect logic and computer science. In particular, it is from this line of work that Coquand and Huet's Calculus of Constructions [41, 42] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [74].

3.1.1 Towards the calculus of constructions

The λ-calculus, defined by Church [40], is a remarkably succinct model of computation that is defined via only three constructions (abstraction of a program with respect to one of its parameters, reference to such a parameter, application of a program to an argument) and one reduction rule (substitution of the formal parameter of a program by its effective argument). The λ-calculus, which is Turing-complete, i.e. which has the same expressiveness as a Turing machine (there is for instance an encoding of numbers as functions in λ-calculus), comes with two possible semantics referred to as call-by-name and call-by-value evaluations. Of these two semantics, the first one, which is the simplest to characterise, has been deeply studied in the last decades [25].

To explain the Curry-Howard correspondence, it is important to distinguish between intuitionistic and classical logic: following Brouwer at the beginning of the 20^{th} century, classical logic is a logic that accepts the use of reasoning by contradiction while intuitionistic logic proscribes it. Then, Howard's observation is that the proofs of the intuitionistic natural deduction formalism exactly coincide with programs in the (simply typed) λ-calculus.

A major achievement has been accomplished by Martin-Löf who designed in 1971 a formalism, referred to as modern type theory, that was both a logical system and a (typed) programming language [64].

In 1985, Coquand and Huet [41, 42] in the Formel team of INRIA-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [51, 68]. This formalism, called the Calculus of Constructions, served as logical foundation of the first implementation of Coq in 1984. Coq was called CoC at this time.

3.1.2 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 [43] 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 Differential and probabilistic tools for programming, reasoning and learning

In this research topic, we want to design and implement an incremental and probabilistic notion of *mathematical document* amenable to statistical learning methods. We will rely on differential, probabilistic and metric extensions of Martin-Löf dependent type theory, the formal system on which the Coq proof assistant is implemented.

3.3 A probabilistic and differential type theory.

Our first objective will be to develop a type theoretic and compositional framework for data and probabilistic programs, taking into account independence, distance and expectation for probabilistic distributions of data and/or programs. We will build on two recent advances in the field. First, the introduction of probabilistic programming languages for differential privacy based on ideas of linear logic, and equipped with

a compositional and typed metrics measuring distance between programs as well as between large-scale data [48, 33, 21]. Second, the development of differential extensions of functional programming languages allowing to implement naturally optimisation algorithms based on gradient retropropagation [19, 34] of which we wish to explore the possible connections with differential linear logic [47].

3.4 A metrics on proofs and on types.

We shall develop a metric point of view on the homotopical framework of Martin-Löf Type Theory [67] so as to be able to define behavioural and observational distances between proofs in types, and between types in universes. We shall use the recent fibrational characterisation of the Kantorovich-Wasserstein distance on probability distributions [30] in order to lift these metrics to distributions within Type Theory. Our goal is to obtain in that way distances that we shall be able to evaluate and optimise between distributions of proofs/elements in a type, or distributions of types in a Martin-Löf universe.

One of our objectives will then be to understand how to calibrate so defined distances within Type Theory using mathematical information retrieval (MIR) algorithms based on distances obtained by automatic learning methods [20]. We will also try to articulate concurrent separation logic with Type Theory in order to give a compositional account of the dependence and independence of the various components of the probability distributions on data, proofs and concepts. We shall work in the spirit of [27, 24] building on the formal correspondence observed by Alex Simpson [71] between separation between memory states of a machine and independence between random variables.

We consider possible interactions with François Pottier (Cambium) and Arthur Charguéraud (Camus) on these questions of concurrent separation logic and its correct integration within Martin-Löf Type Theory.

3.5 Structure of mathematical documents

One of the most innovating and federating aspects of the project will be to conceive and implement a formal notion of *mathematical document* and its connections with underlying logical theories, in the line of the recent advances by Makarius Wenzel in Isabelle/PIDE [75]. Specifically this framework will include *projection* based mathematical content extraction tools allowing to build, out of a mathematical document, the libraries relevant to a given theorem or proof. In that way the users will be able to know in which logical fragment they are working at each step of their mathematical activities, and the relations between the various components of a library. We will devote a special care to the notion of *transformation paths* which connect mathematical documents by successive applications of "patches", in the spirit of git, darcs or Pijul [65, 36, 22]. We shall formalise and implement this notion of transformation path in such a way that we shall be able to compose them efficiently, whilst endowing them with a distance compatible with the probabilistic and metric approach to Type Theory explained in the previous paragraphs. This *incremental* point of view on mathematical proof construction will allow us to set up learning tools based on statistical analysis of the behaviour of users in the way they build proofs [69] rather than on the form of the proofs themselves. A key principle of our mathematical document format is to serve as a good basis for understanding sets of concepts, theorems, and proofs and their evolution as a data set that could be used by state of the art learning methods [49] to help the document and proof writers. In this direction, there are many possibilities, including improving search, type-checking and conversion, and suggestions on proof structure and tactics. We consider working in cooperation with Vincent Silès (Facebook Paris) on the use of learning tools and the automated and interactive guidance of users.

3.6 Architecture and design of a proof assistant for mathematicians

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 [76] 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 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 Picube (Paris), Stamp (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 Apple (M. Soegtrop). The latest released versions have typically a couple of dozens of contributors (e.g. 40 for 8.8, 54 for 8.9, ...).

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 [28], the more advanced Certified Programming with Dependent Types book by Chlipala [38] 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.

3.6.1 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.

3.6.2 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.

3.6.3 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.

3.6.4 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.

3.6.5 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.

3.6.6 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.

3.6.7 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, jsCoq, 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.

3.7 Formalization and linguistics of mathematics

In this research topic, we aim at contributing to the formalisation of mathematics by developing direct interactions with the mathematical community. The current period is more than ever propitious for involving colleagues from mathematics in a formalisation process. Indeed, more and more mathematicians express a strong interest for the formalisation and growing expectations for the benefits it could have for their research. We view this as a direct consequence of the maturity gradually acquired by proof assistants together with the impressive work of conviction carried out by late Vladimir Voevodsky around HoTT, and to striking results such as the formalisation by Georges Gonthier of the four-colour theorem [53] and Feit-Thompson theorem [52, 54].

However, to actually formalise a large body of contemporary mathematics remains truly a research issue as it requires to improve the design of proof assistants. Most notably, there is a need for a true work of linguists in order to fill the gap between vernacular proof languages and formal proofs, which is a requirement for fostering a dynamic and sustainable community ranging from computer science to pure mathematics. In addition to the skills and energies gathered in the team itself, we benefit from our unique position in the Sophie Germain building of the Université de Paris, within the IRIF and in the immediate neighbourhood of IMJ-PRG, a laboratory in pure mathematics. The activity of the team in this topic will focus on lowering the cost of starting and pursuing formalisation by mathematicians.

3.7.1 Formalisation of mathematics.

We will work for this in a proactive way, in close collaboration with IRIF and IMJ-PRG gathering motivated mathematicians and computer scientists willing to formalise the mathematics they teach at the University, or on which they conduct their research work. Indeed, we aim at formalising classical mathematics curricula, pieces of contemporary mathematics as well as mathematical tools implemented in theoretical computer science. We will draw inspiration from the usually heuristic practice of mathematics, aiming to make the writing and reading of Coq documents altogether more intuitive, more straightforward and more flexible. This software development and mechanisation work will be combined with a study of the linguistic structure of mathematical documents, in collaboration with Philippe de Groote (Sémagramme): we think that there is a need to better understand the linguistic structures at work in the daily life of a mathematician and their formal nature.

More specifically, among the subjects of formalisation of mathematics we plan to carry in the team, three in particular should be mentioned:

- On the mathematical side, formalisation of parts of contemporary mathematics, in connection with members of the IMJ-PRG.

- On the CS side, we plan to formalise and certify algorithms on graphs (Jean-Jacques Lévy), with the aim of establishing links with other IRIF members working on graph algorithms, in particular with Laurent Viennot; we also intend to certify developments in the theory of automata and formal languages or proof theory as used at IRIF, in particular with Thomas Colcombet.

- We also plan to contribute to an ambitious formalisation project aiming at designing a large Coq mathematical library covering undergraduate mathematics. This project is of course of a broader scope and will involve other research teams, within INRIA or elsewhere in France and abroad.

This research topic will therefore be based at the same time on a formalisation activity internal to the team, and on a long-term work of animation and construction of a scientific community involved in formalisation. We plan to contribute to the Coq training of our maths and CS colleagues (from PhD students to post-docs and those holding a permanent position), and not only grad students as it is more commonly the case, in particular through the organisation of regular sessions of working groups dedicated to helping colleagues in their formalisation tasks and by considering the opportunity to set up thematic schools in collaboration with the other teams of the institute contributing to Coq. A medium-term objective could be to achieve that some pure mathematics modules are based directly on formalised content and that some mathematics tutorials take the form of Coq exercise sessions, starting from existing works on Coq [29, 66, 39]. This will require to develop close collaborations with the different communities of other proof assistants, especially those designed to be well-adapted to the formalisation of mathematics (Lean, Isabelle and Agda in particular). Understanding linguistic aspects of mathematical proofs will be a key to the success of our project.

3.7.2 Linguistics of proofs.

Formalising inevitably leads to a shift of perspective on what a mathematical proof is. From a mathematical standpoint, it is conventional, even if developments are emerging, to be satisfied of the mere possibility of a formalisation (which is never even sketched out) typically in the set-theoretical formalism, and to focus on the construction of a natural language discourse that is precise enough to convince the reader. From the proof assistant standpoint, the seminal vision which initially aimed only at making effective this virtuality, tends to evolve. New lines of communication and convergence are emerging: a proof is no longer strictly made up of logical inference rules, such as introducing or eliminating a connective, but it is made of more abstract entities such as the use of a lemma, the replacement of equals by equals, reasoning by induction, simplifying a polynomial expression, decomposing a formula in atoms, etc. With the arrival of a new generation of interactive proof engines [72, 26], a proof is no longer seen strictly as a tree derivation, but as a graph whose nodes can be refined with a reasonable degree of freedom; moreover, the order in which these transformations are applied interactively appears in the "incremental" format of the machine maths document. This is in line with the historical evolution of proof methods in order to escape from the "low level" of logic and get closer to more abstract conceptual levels used by humans. Our investigations will follow this line as we plan to analyse the linguistic structure of mathematical texts, with the ambition to develop the levels of abstraction which would eventually allow a direct formal understanding of a mathematical text at the level of mathematical discourse in which it is expressed. Examples of the sorts of linguistic structures we plan to analyse are "reasoning by analogy", reasoning modulo isomorphism, or even modulo inclusion, and of course reasoning modulo general equational theories in general.

On this natural (mathematical) language processing part, we plan a collaboration with Philippe de Groote of the Sémagramme team (Inria Nancy & Loria), to identify the necessary structures for a flexible formalisation of vernacular mathematical proofs in order to implement this linguistic structure within Coq, procedural and discursive in nature rather than tree-like as described above. One of the objectives will be to be able to formalise in a more transparent and direct way mathematical texts, such Bourbaki's Éléments de mathématiques.

Regarding the design and development of a general mathematical library, it is undoubtedly too early to describe the directions we will take, but we have some ongoing discussions with Assia Mahboubi (Gallinette), Yves Bertot and Cyril Cohen (Scalp) around these essential questions. We also wish to develop the team's scientific interactions with Michael Soegtrop (Apple) and we closely follow his projects on proving algorithms of symbolic computing, around constructive reals and the formal integrator *Rubi*. We

are also in contact with mathematicians from IMJ-PRG, and in particular with Antoine Chambert-Loir, who expressed a keen interest in formalisation.

We want to develop fruitful discussions with the communities of other proof assistants, such as Agda, Isabelle or Lean.

3.8 Higher-dimensional algebra and homotopic type theory

3.8.1 Higher-dimensional algebra

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

In the last decades, the importance of higher-dimensional categories has grown fast, mainly with the new trend of *categorification* that currently touches algebra and the surrounding fields of mathematics. Categorification is an informal process that consists in the study of higher-dimensional versions of known algebraic objects (such as higher Lie algebras in mathematical physics [23]) 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 [46]).

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 [60], it has been observed recently that the so-called "identity types" are naturally equiped with a structure of ∞-groupoid: the 1-cells are the proofs of equality, the 2-cells are the proofs of equality between proofs of equality, and so on. The striking resemblance with the fundamental ∞-groupoid of a topological space led to the conjecture that homotopy type theory could serve as a replacement of set theory as a foundational language for different fields of mathematics, and homotopical algebra in particular.

3.8.2 Higher-dimensional rewriting

Higher-dimensional categories are algebraic structures that contain, in essence, computational aspects. This has been recognised by Street [73], and independently by Burroni [35], 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 [62, 63, 55, 56], and beyond: Petri nets [58] and formal proofs of classical and linear logic have been expressed in this framework [57]. 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 [31].

4 Application domains

The application domains of the Picube team researchers range from the formalization of mathematical theories and computational systems using the Coq proof assistant to the design of programming languages with rich type systems and effects (stateful, concurrent, probabilistic) and the design and analysis of certified program transformations.

5 Social and environmental responsibility

5.1 Footprint of research activities

The environmental impact of the team is mainly two sorts:

- travel footprint to attend conferences or for longer-term visits
- secondly, computer resources notably those affected to the series of benchmark tests which are run before integrated new features in the Coq system.

Members of the Picube team are committed to decreasing the environmental impact of our research. In the IRIF lab environment, a working group investigates the footprint of our scientific community and its practices (notably numerous international conferences) and the potential medium and long-term evolution that can be made. Several members of the team and active contributors or interested followers of the WG. As an achievement of this working group, recommendations have been made at the IRIF level to encourage every lab member to travel by train rather than by plane when the travel duration is not significantly longer by train.

6 Highlights of the year

In the continuation of the work of the PiR2 team, the Picube research team wishes to contribute to the education of new generations of students taking the lead in proof assistant technology and formalisation of mathematics and computer science. We benefit from the fact that Picube is a joint project team with the IRIF lab of Université de Paris, within the Fondation des Sciences Mathématiques de Paris (FSMP) and with an active participation of its members to the Master Logique Mathématique et Fondements de l'Informatique (LMFI) and the Master Parisien de Recherche en Informatique (MPRI) both taught in the Bâtiment Sophie Germain where the IRIF lab is located. We believe that the development of a formal corpus of mathematics is a foundational challenge potentially as important as the Bourbaki enterprise initiated in the late 1930s.

7 New software, platforms, open data

The Coq Proof Assistant Web site: http://www.coq.inria.fr.

Self-assessment:

- Software Family `research`: Software as a Vector for Knowledge (see SAE, Section 3.1).
- Audience: `universe`: wide-audience software (aims to be usable by a wide public, to become the reference software in its area, etc.).
- Evolution and maintenance: `lts`: long term support.
- Duration of the Development (Duration): since (long before) the start of the team.
- Free Description: Coq is an interactive proof assistant based on the CIC (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 integrated development environment (IDE).

  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, ...

  Contact: Matthieu Sozeau

  Participants: Yves Bertot, Frederic Besson, Tej Chajed, Cyril Cohen, Pierre Corbineau, Pierre Courtieu, Maxime Denes, Jim Fehrle, Julien Forest, Emilio Jesús Gallego Arias, Gaetan Gilbert,

Georges Gonthier, Benjamin Grégoire, Jason Gross, Hugo Herbelin, Vincent Laporte, Olivier Laurent, Assia Mahboubi, Kenji Maillard, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Clément Pit-Claudel, Kazuhiko Sakaguchi, Vincent Semeria, Michael Soegtrop, Arnaud Spiwack, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Anton Trunov, Li-Yao Xia, Theo Zimmermann.

jsCoq Web site: https://coq.vercel.app.

Self-assessment:

- Software Family `vehicle`: Software as a Vehicle for Research (see SAE, Section 3.2).
- Audience: `community`: large audience software, usable by people inside and outside the field with a clear and strong dissemination, validation, and support action plan;
- Evolution and maintenance: `lts`: long term support.
- Duration of the Development (Duration): 8 years
- Free 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.

SerAPI Web site: https://github.com/ejgallego/coq-serapi.

Self-assessment:

- Software Family `vehicle`: Software as a Vehicle for Research (see SAE, Section 3.2).
- Audience: `community`: large audience software, usable by people inside and outside the field with a clear and strong dissemination, validation, and support action plan;
- Evolution and maintenance: `basic`: basic maintenance to keep the software alive;
- Duration of the Development (Duration): 7 years
- Free 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).

  Contact: Emilio Jesus Gallego Arias

  Participant: Emilio Jesus Gallego Arias, Thierry Martinez

coqbot Web site: https://github.com/coq/bot/. Self-assessment:

- Software Family `vehicle`: Software as a Vehicle for Research (see SAE, Section 3.2).
- Audience: `partners`: to be used by people inside and outside the team but without a clear and strong dissemination and support action plan;
- Evolution and maintenance: `basic`: basic maintenance to keep the software alive;
- Duration of the Development (Duration): 3 years
- Free Description: This software is a bot to help and automatize the development of the Coq proof assistant on the GitHub platform. It is written in OCaml and provides numerous features: synchronization between GitHub and GitLab to allow the use of GitLab for automatic testing (continuous integration), management of milestones on issues, management of the backporting process, merging of pull request upon request by maintainers, etc.

  Most of the features are used only for the development of Coq, but the synchronization with GitLab feature is also used in dozens of independent projects.

  Contact: Theo Zimmermann

coq-lsp Web site: https://github.com/ejgallego/coq-lsp.

Self-assessment:

- Software Family `vehicle`: Software as a Vehicle for Research (see SAE, Section 3.2).
- Audience: `partners`
- Evolution and maintenance: `basic`: basic maintenance to keep the software alive;
- Duration of the Development (Duration): 1 year
- Free Description: coq-lsp is a Language Server and Visual Studio Code extension for the Coq Proof Assistant. Experimental support for Vim and Neovim is also available in their own projects.
  Contact: Emilio Jesus Gallego Arias
  Participant: Ali Caglayan, Emilio J. Gallego Arias, Shachar Itzhaky

PyCoq Web site: https://github.com/ejgallego/pycoq.

Self-assessment:

- Software Family `vehicle`: Software as a Vehicle for Research (see SAE, Section 3.2).
- Audience: `partners`
- Evolution and maintenance: `basic`: basic maintenance to keep the software alive;
- Duration of the Development (Duration): 1 year
- Free Description: PyCoq is a set of bindings and libraries allowing to interact with the Coq interactive proof assistant from inside Python 3.
  Contact: Emilio Jesus Gallego Arias
  Participant: Emilio Jesus Gallego Arias, Thierry Martinez

8 New results

Participants: Thomas Ehrhard, Emilio Jesus Gallego Arias, Hugo Herbelin, Farzad Jafar-Rahmani, Paul-Andre Mellies, Vincent Moreau, Alexis Saurin.

8.1 The fundamental structures of logic and mathematical reasoning

8.1.1 Proof-theory of logics with induction and coinduction.

Saurin has completed the line of work aiming at building a full fledge circular proof theory with two new results:

The cut-elimination result for $\mu\mathsf{MALL}^\infty$ has been extended to extended to full $\mu\mathsf{LL}^\infty$ by Saurin [10], by studying the properties of a well-known fixed-point encoding of LL exponentials. This allows, using techniques from infinitary rewriting, to lift $\mu\mathsf{MALL}^\infty$ cut-elimination result to $\mu\mathsf{LL}^\infty$. As a by-product, cut-elimination for non-wellfounded $\mu\mathsf{LJ}^\infty$ and $\mu\mathsf{LK}^\infty$ was obtained as well. This approach is quite robust w.r.t. modifications of the validity conditions, allowing for instance to be adapted to bouncing validity. This approach also sheds a new light on LL exponentials by providing a natural setting in which non-uniform exponentials do live. This last point was the topics of the LMFI M2 internship of Adrien Shalit jointly supervised by Petrisan and Saurin during the spring and summer 2023.

Together with Bauer, Saurin finally completed the above result providing the first syntactic cut-elimination result for a (circular) proof system containing the full modal μ-calculus: to achieve this, they design a linear version of the modal μ-calculus and prove cut-elimination for a family of linear fixed-point logics with super-exponential, building on previous results by Bauer and Laurent. This result not-only allows to cover the case of the modal μ-calculus but also treat most of the known light logics in the literature, extended with least and greatest fixed-points. A pre-print is available and currently submitted.

8.1.2 Reversible computation with inductive types

Chardonnet, Saurin and Valiron developed a term calculus for a class of type isomorphisms for circular $\mu\mathsf{MALL}^\infty$, from which they extract a language for reversible computation with inductive types [7, 37]. In the paper published at CSL 2023, they studied the expressiveness of the reversible calculus, its connection with type isomorphisms of circular $\mu\mathsf{MALL}^\infty$, providing a first step before moving forward to modelling quantum computations with structured datatypes.

8.1.3 Comparing finitary and infinitary proof systems for $\mu\mathsf{MALL}$.

Following up on their previous work with Das on the decision problems for various fragments of $\mu\mathsf{MALL}$ [18], they compared in a new work the relative expressivity, at the provability level, of the (finitely-branching) non-wellfounded proofs, $\mu\mathsf{MALL}^\infty$, with infinitely branching well-founded proofs built on the language of $\mu\mathsf{MALL}$. With this aim, they could establish that $\mu\mathsf{MALL}^\infty$ is strictly contained in $\mu\mathsf{MALL}_\omega$ with, as an immediate consequence, that $\mu\mathsf{MALL}^\infty$ does not enjoy a finite model property in any adequate semantics. Those results have been presented and published at FSTTCS 2023 [45].

8.1.4 Controlling sequentialization in proof nets.

In a collaboration with Aurore Alcolei and Luc Pellissier, published at MFPS 2023 [3], Alexis Saurin internalized the notion of jumps of linear logic proof-nets (which can be used as an alternative to boxes) in a slight extension of MLL (multiplicative linear logic). Jumps, which have been extensively studied by Faggian and di Giamberardino (building on prior work by Curien and Faggian on L-nets), can express intermediate degrees of sequentialization between a sequent calculus proof and a fully desequentialized proof-net. The logical strength of jumps is analyzed by internalizing them in an extension of MLL where axioms on a specific formula introduce constraints on the possible sequentializations. The jumping formula needs to be treated non-linearly, which is done either axiomatically, or by embedding it in a very controlled fragment of multiplicative-exponential linear logic, uncovering the exponential logic of sequentialization.

8.1.5 An operadic account of context-free grammars and languages

Melliès and Zeilberger develop in [8] a categorical framework based on non-symmetric operads (= multicategories) to describe context-free grammars and establish the Chomsky-Schützenberger representation theorem for context-free languages. In this approach, a context-free grammar is defined as a functor from an operad freely generated by a species of production rules, to an operad of spliced words introduced for the first time in this paper. A notion of automaton on a category $\mathscr{C}$ generalising the usual notion of automaton on words is formulated there as a functor from $\mathscr{Q} \to \mathscr{C}$ satisfying a unique-factorisation-lifting as well as a fibrewise finiteness property, where $\mathscr{Q}$ describes the category of runs over the automaton. One main benefit of the approach is that it provides a more high-level account of traditional aspects of automata-theory. In particular, the fact that the intersection of a context-free language with a regular language is context-free is derived from the fact that pullbacks computed in the category of operads preserves context-free grammars seen as functors. Similarly, the Chomsky-Schützenberger representation theorem is derived from the observation that the functor which turns a category into its spliced arrow operad has a left adjoint functor which turns any operad to its category of contours. In particular, the fact that the unit of the adjunction transports every operad to its spliced operad of contours platys a fundamental role in the argument. This work is part of a long-term project of integrating refinement systems, environment machines and automata theory in a single and unified framework.

8.2 Differential and probabilistic tools for programming, reasoning and learning

8.2.1 Coherent differentiation

Differential Linear Logic (DiLL) has been introduced by Ehrhard and Regnier in the mid 2000's. This extension of linear logic provides a new interpretation of the exponentials, turning them into a modality related to communication rather than to the sole resource replicability and erasing. DiLL also provided a new understanding of resource calculi and of intersection types, allowing to apply to programs and

proofs an operation of approximation which is a syntactic version of the standard Taylor expansion of functions. Until 2021, it seemed that DiLL was doomed to feature a strong form of nondeterminism due to the interaction between differentiation and the structural rule of contraction (Leibniz rule), making it incompatible with stable or sequential denotational interpretations of programs. The article [1] presents Coherent Differentiation, a new denotational setting discovered in 2021, featuring differentiation operations while being compatible with stable and sequential interpretations. A syntactic account of this new approach to differentiation has also been proposed by the author in a subsequent paper.

8.2.2 Causality structures in probabilistic rewriting

The question of describing the causal structure of symbolic rewriting systems such as the lambda-calculus is at the heart of an old and vibrant connection established in the 1980s between proof theory and concurrency theory. The idea is that the process of reducing beta-redexes generates causal structures similar to what one finds in Petri nets and process calculi such as CCS or the π-calculus. Typically, the reduction of one beta-redex u can create a beta-redex v whose reduction can in turn create a third beta-redex w, and so on. These causal structures give rise to the notion of family designed by Jean-Jacques Lévy in his theory of optimality. They also appear in term rewriting and graph rewriting and play a fundamental role in the description of the stochastic behaviour of probabilistic rewriting. Together with Nicolas Behr (IRIF) and Noam Zeilberger (LIX), Melliès has developed in [6] a framework where these causality structures can be described in a unified way in the language of double categories. The idea is to compose rewriting rules seen as representable presheaves under the action of context extension. Composition is defined using a generalisation to double categories of the usual Day convolution tensor product of presheaves on a monoidal category. One main novelty of the work is to integrate for the first time unification and rewriting in the same categorical framework.

In joint work with van Gool, Melliès and Moreau introduced in [2] a notion of profinite λ-calculus which provides a new topological account of λ-terms, different from the usual Scott domain topology and based on new insights on the connection between typed λ-calculus and automata theory. The construction provides to every type A a compactification of the set of λ-terms of type A defined as a profinite limit of the interpretations of the type A in the category of finite sets. The profinite limit may be also understood as providing the Stone dual of the boolean algebra of regular languages of λ-terms of type A introduced by Salvati [70]. The categorical definition of profinite λ-terms is identified in [2] with a definition based on parametricity. This establishes an important and unexpected connection between polymorphism and automata theory. The profinite completion of the Church encoding of finite words in a given alphabet Σ is shown to coincide with the usual notion of profinite monoid. Then, in the continuation of this work, Moreau establishes in joint work with Nguyen [9] that the definition of regular language of simply typed λ-terms does not depend on the choice of (locally finite) cartesian closed category where the λ-calculus is interpreted, as long as the category of interpretation has enough points. They also establish the important unifying result that the semantic definition by Salvati [70] of regular languages of simply typed λ-terms coincides with the purely syntactic definition by Hillebrand and Kanellakis [59].

8.3 Architecture and design of a proof assistant for mathematicians

8.3.1 Survey on the Coq community

The goal of the article [4] is to better understand Coq users and community, and to make informed decisions about the research and development challenges around a system such as Coq. A key point on the paper is the multidisciplinary research team, bringing together researchers from several areas in Computer and Social sciences. Inspired by previous Surveys in Coq, OCaml, and open source world, a survey was designed and ran for the Coq community totaling 109 questions, and obtaining 466 answers, to this date, the largest survey done on users of Interactive Theorem Provers. The data were analyzed using rigorous regression methods common on the social sciences.

9 Bilateral contracts and grants with industry

Participants: Guillaume Baudart, Pierre-Louis Curien, Abhishek De, Alen Duric, Thomas Ehrhard, Emilio Jesus Gallego Arias, Hugo Herbelin, Farzad Jafar-Rahmani, Pierre Letouzey, Paul-Andre Mellies, Vincent Moreau, Alexis Saurin, Gabriel Scherer.

The team has not undertaken any sustained transfer activities, however we have informal but regular scientific contacts with industrial users in several companies, such as Apple (Michael Soegtrop who works on the safety of cyber-physical systems is a regular Coq contributor), Tweag I/O (Arnaud Spiwack is a close collaborator), Nomadic labs and Tezos (in particular with Yann Regis-Gianas), OpenAI (with discussions and visits by Stanislas Polu, now working at Dust, a startup company which he co-founded).

10 Partnerships and cooperations

Participants: Guillaume Baudart, Pierre-Louis Curien, Abhishek De, Alen Duric, Thomas Ehrhard, Emilio Jesus Gallego Arias, Hugo Herbelin, Farzad Jafar-Rahmani, Pierre Letouzey, Paul-Andre Mellies, Vincent Moreau, Alexis Saurin, Gabriel Scherer.

10.1 International initiatives

10.1.1 Participation in other International Programs

Thomas Ehrhard chairs the french-italian GDRI on Linear Logic which is finishing this year. Paul-André Melliès and Alexis Saurin are also members of the GDRI.

Thomas Ehrhard and Paul-André Melliès are international members of the EPSRC project "Resources in Computation" chaired by Samson Abramsky (UCL) and Anuj Dawar (Cambridge).

10.2 International research visitors

Olivier Laurent

Status Senior Researcher

Institution of origin: ENS Lyon, CNRS

Country: France

Dates: 25 January 2023

Context of the visit: He participated as an examiner in Farzad Jafarrahmani's PhD thesis defence.

Nicola Gambino

Status Senior Researcher

Institution of origin: University of Leeds

Country: United Kingdom

Dates: from 24 to 26 January 2023

Context of the visit: He participated as an examiner in Farzad Jafarrahmani's PhD thesis defence.

Christine Tasson

Status Senior Researcher

Institution of origin: Sorbonne Université

Country: France

Dates: from 24 to 27 January 2023

Context of the visit: She participated as an examiner in Farzad Jafarrahmani's PhD thesis defence.

Marcelo Fiore

Status Senior Researcher

Institution of origin: University of Cambridge

Country: United Kingdom

Dates: from 24 to 25 January 2023

Context of the visit: He participated as an examiner in Farzad Jafarrahmani's PhD thesis defence.

Ali Caglayan

Status PhD Thesis

Institution of origin: University of Gothenburg

Country: Sweden

Dates: from 30 January to 2 February 2023

Context of the visit: He worked with Emilio Gallego on the Coq proof development.

Nima Rasekh

Status Postdoc student

Institution of origin: Max Planck Institute for Mathematics in Bonn

Country: Germany

Dates: from 17 to 23 February 2023

Context of the visit: He gave a talk at the semantics working group and worked with El Mehdi Cherradi on a joint paper.

Ambroise Lafont

Status Postdoc student

Institution of origin: University of Cambridge

Country: United Kingdom

Dates: from 19 to 24 March 2023

Context of the visit: He worked with the members of the Picube team.

Olivier Laurent

Status Postdoc student

Institution of origin: Max Planck Institute for Mathematics in Bonn

Country: France

Dates: 3 April 2023

Context of the visit: He worked with the members of the Picube team.

André Joyal

Status Senior Researcher

Institution of origin: Université de Québec à Montréal

Country: Canada

Dates: from 28 May to 12 June 2023

Context of the visit: He worked with the members of the Picube team on the semantics of homotopy type theory.

Aurore Alcolei

Status Postdoc student

Institution of origin: Universita degli Studi di Bologna,

Country: Italy

Dates: from 30 June to 5 July 2023

Context of the visit: She worked with Alexis Saurin on a joint paper.

10.3 National initiatives

Pierre-Louis Curien, Thomas Ehrhard, Emilio J. Gallego Arias, Hugo Herbelin, Paul-André Melliès 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 (see website here).

Pierre-Louis Curien and Paul-André Melliès are members of the **GDR Homotopie**, 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.

Kostia Chardonnet, Abhishek De, Thomas Ehrhard, Farzad Jafarrahmani, Hugo Herbelin, Paul-André Melliès, Daniela Petrisan and Alexis Saurin (coordinator) are members of the four year RECIPROG project. RECIPROG is an ANR collaborative project (a.k.a. PRC) started in the fall 2021-2022 and running till the end of 2025. ReCiProg aims at extending the proofs-as-programs correspondence to recursive programs and circular proofs for logic and type systems using induction and coinduction. The project aims at contributing both to the necessary theoretical foundations of circular proofs and to the software development allowing to enhance the use of coinductive types and coinductive reasoning in the Coq proof assistant: such coinductive types present in the current state of the art serious defects that the project will aim at solving.

The project is coordinated by Alexis Saurin and has four sites: IRIF in Paris where the team Picube is located, LIP in Lyon, LIS in Marseille and LS2N in Nantes.

Hugo Herbelin participates in the **Inria Challenge LiberAbaci**. LiberAbaci is a collaborative project aimed at improving the accessibility of the Coq interactive proof system for an audience of mathematics students in the early academic years. The lead is Yves Bertot and the involved teams are: Cambium (Paris), Camus (Strasbourg), Gallinette (Nantes) PiCube (Paris), Spades (Grenoble), Stamp (Sophia Antipolis), Toccata (Saclay), LIPN (Laboratoire d'Informatique de Paris Nord).

10.4 Regional initiatives

In collaboration with Riccardo Brasca (coordinator) and Antoine Chambert-Loir, two mathematicians specialists in number theory working at the Institut de Mathématiques de Jussieu Paris Rive Gauche (IMJ-PRG), Hugo Herbelin (coordinator), Pierre Letouzey, Paul-André Melliès and Alexis Saurin initiated and launched an Emergence Recherche project of the Université Paris Cité, APRAPRAM. The aim of the project is to contribute to a formalization of Fermat's last theorem in the special case of regular primes, targeting a cross-fertilization between the Lean and the Coq/Rocq communities.

11 Dissemination

Participants: Esaïe Bauer, Vincent Blazy, Kostia Chardonnet, El Mehdi Cherradi, Pierre-Louis Curien, Thomas Ehrhard, Emilio Jesus Gallego Arias, Hugo Herbelin, Pierre Letouzey, Paul-Andre Mellies, Vincent Moreau, Sarah Reboullet, Alexis Saurin.

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

Member of the organizing committees Alexis Saurin is member of the organizing committee of the annual Scalp meeting, to be held at CIRM in February 2023.

Emilio Gallego, Hugo Herbelin, Paul-André Melliès and Alexis Saurin, together with Chantal Keller and Marie Kerjean, are members of the organizing committee of the thematic day on proof assistants to be held at JNIM 2024 (Journées nationales du GDR-IM) end of March 2024.

11.1.2 Scientific events: selection

Member of the editorial boards Pierre-Louis Curien is the editor-in-chief of the journal Mathematical Structures in Computer Science. Thomas Ehrhard is an editor of this journal, and Paul-André Melliès is editor of the journal Theoretical Computer Science. Paul-André Melliès is a member of the editorial board of the journal Theoretical Computer Science.

11.1.3 Invited talks

Paul-André Melliès gave an invited lecture in Nice as part of the "Journées Homotopiques" on the occasion of Clemens Berger's 60th birthday.

Paul-André Melliès gave an invited lecture at the "Resources and Co-Resources" workshop at the University of Cambridge from 17 to 19 July 2023.

Alexis Saurin gave an invited tutorial at TLLA 2023, 7th International Workshop on Trends in Linear Logic and Applications, Roma, 1 and 2 July 2023.

11.1.4 Leadership within the scientific community

Alexis Saurin is co-chair of the Scalp working group in GDR-IM (GT Scalp).

11.2 Teaching - Supervision - Juries

Pierre-Louis Curien taught a course on homotopic algebra and higher categories in LMFI (Logique mathématiques et fondements de l'informatique) second-year Master, Université Paris Cité.

Pierre Letouzey taught a course on Coq in LMFI (Logique mathématiques et fondements de l'informatique) second-year Master, Université Paris Cité.

Alexis Saurin taught a lecture on Second-order quantification and fixed-points in logic in LMFI (Logique mathématiques et fondements de l'informatique) second-year master, Université Paris Cité.

Hugo Herbelin and Paul-André Melliès taught a course on homotopy type theory in LMFI (Logique mathématiques et fondements de l'informatique) second-year Master, Université Paris Cité.

Together with Michele Pagani (IRIF), Paul-André Melliès and Thomas Ehrhard taught a course on denotational semantics and linear logic at MPRI (Master Parisien de Recherche en Informatique) second-year Master, Université Paris Cité.

Paul-André Melliès taught a course on lambda-calculus and categories at MPRI (Master Parisien de Recherche en Informatique) first-year Master, at ENS Paris (Ecole Normale Supérieure).

11.2.1 Supervision

PhD supervision

- PhD in progress: Esaie Bauer, on the Curry-Howard correspondence between temporal logic proofs and reactive programming, Université de Paris, started in September 2021, supervised by Alexis Saurin and Thomas Ehrhard.
- PhD in progress: Giulia Manara, Towards parallel cut elimination in MELL proof structures, started in October 2021, supervised by Thomas Ehrhard.
- PhD in progress: Vincent Moreau, on the connections between higher-order automata and topological duality, started in September 2021, supervised by Paul-André Melliès and Sam van Gool.
- PhD in progress: Sarah Reboullet, Parametricity and Univalence, started in September 2021, supervised by Hugo Herbelin.
- PhD in progress: Clément Théron, sémantique interactive et vectorielle de la programmation probabiliste et différentielle, started in September 2021, supervised by Thomas Ehrhard and Paul-André Melliès
- PhD in progress: Vincent Blazy, Fine-graine structure of universe subtyping in Coq and applications to program certification and mathematical foundations, Université de Paris, started in September 2020, supervised by Hugo Herbelin and Pierre Letouzey.
- PhD in progress: Kostia Chardonnet, Inductive and coinductive types in quantum programming languages, Université Paris Saclay, started in November 2019, supervised by Alexis Saurin and Benoît Valiron. Defense planned january 2023
- PhD in progress: El Medhi Cheraddi, Computational and interactive interpretation of homotopy type theory, started in september 2021, supervised by Paul-André Melliès.
- PhD in progress: Alen Đurić, Normalisation for monoids and higher categories, Université de Paris, started in October 2O19, supervised by Yves Guiraud and Pierre-Louis Curien. Defense planned in 2023
- PhD in progress: Farzad Jafar-Rahmani, Denotational semantics of circular and non-wellfounded proofs, Université de Paris, started in October 2019, supervised by Thomas Ehrhard and Alexis Saurin. Defense planned in january 2023
- PhD in progress: Hugo Moeneclaey, Syntax of spheres in homotopy type theory, Université de Paris, started in September 2019, supervised by Hugo Herbelin.
- PhD in progress: Antoine Allioux, Opetopes in Type Theory, Université de Paris, since March 2018, supervised by Yves Guiraud and Matthieu Sozeau. The defense should take place in july 2023.
- PhD defended: Abhishek De, Proof-nets for fixed-point logics and non-well-founded proofs, Université de Paris, since October 2018, supervised by Alexis Saurin.

12 Scientific production

12.1 Publications of the year

International journals

[1] T. Ehrhard. 'Coherent differentiation'. In: *Mathematical Structures in Computer Science* 33.4-5 (Apr. 2023), pp. 259–310. DOI: 10.1017/S0960129523000129. URL: https://hal.science/hal-03282799.

[2] S. van Gool, P.-A. Melliès and V. Moreau. 'Profinite lambda-terms and parametricity'. In: *Electronic Notes in Theoretical Informatics and Computer Science* Volume 3 - Proceedings of MFPS XXXIX (23rd Nov. 2023). DOI: 10.46298/entics.12280. URL: https://hal.science/hal-04540079.

International peer-reviewed conferences

[3] A. Alcolei, L. Pellissier and A. Saurin. 'The exponential logic of sequentialization'. In: *Entics*. MFPS XXXIX - 39th Conference on the Mathematical Foundations of Programming Semantics. Vol. Volume 3 - Proceedings of ENTiCS. Bloomington (Indiana), United States, 23rd Nov. 2023. DOI: 10.46298/entics.12419. URL: https://hal.science/hal-04308279.

[4] A. de Almeida Borges, A. C. Artís, J.-R. Falleri, E. J. Gallego Arias, É. Martin-Dorel, K. Palmskog, A. Serebrenik and T. Zimmermann. 'Lessons for Interactive Theorem Proving Researchers from a Survey of Coq Users'. In: *Leibniz International Proceedings in Informatics*. 14th International Conference on Interactive Theorem Proving (ITP 2023). Vol. 268. Leibniz International Proceedings in Informatics (LIPIcs) 12. Białystok, Poland: Dagstuhl Publishing, 2023, pp. 1–18. DOI: 10.4230/LIPIcs.ITP.2023.12. URL: https://telecom-paris.hal.science/hal-04098856.

[5] E. Beffara, F. Castro, M. Guillermo and É. Miquey. 'Concurrent Realizability on Conjunctive Structures'. In: FSCD 2023 - 8th International Conference on Formal Structures for Computation and Deduction. Rome, Italy, 2023. URL: https://inria.hal.science/hal-04083002.

[6] N. Behr, P.-A. Melliès and N. Zeilberger. 'Convolution Products on Double Categories and Categorification of Rule Algebras'. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. FSCD 2023 - 8th International Conference on Formal Structures for Computation and Deduction. Vol. 260. 8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023). Rome, Italy: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28th June 2023, 17:1–17:20. DOI: 10.4230/LIPIcs.FSCD.2023.17. URL: https://hal.science/hal-04222049.

[7] K. Chardonnet, A. Saurin and B. Valiron. 'A Curry-Howard Correspondence for Linear, Reversible Computation'. In: *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*. CSL 2023 - 31st EACSL Annual Conference on Computer Science Logic. Varsovie (Warsaw), Poland: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.CSL.2023.13. URL: https://hal.science/hal-04308283.

[8] P.-A. Melliès and N. Zeilberger. 'Parsing as a lifting problem and the Chomsky-Schützenberger representation theorem'. In: *Electronic Notes in Theoretical Informatics and Computer Science*. MFPS 2022 - 38th conference on Mathematical Foundations for Programming Semantics. Vol. Volume 1 - Proceedings of... Ithaca, NY, United States, 22nd Feb. 2023. DOI: 10.46298/entics.10508. URL: https://hal.science/hal-03702762.

[9] V. Moreau and L. T. D. Nguyễn. 'Syntactically and Semantically Regular Languages of λ-Terms Coincide Through Logical Relations'. In: 32nd EACSL Annual Conference on Computer Science Logic (CSL 2024). Vol. 288. Leibniz International Proceedings in Informatics (LIPIcs). Naples, Italy: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 40:1–40:22. DOI: 10.4230/LIPIcs.CSL.2024.40. URL: https://hal.science/hal-04447910.

[10] A. Saurin. 'A Linear Perspective on Cut-Elimination for Non-wellfounded Sequent Calculi with Least and Greatest Fixed-Points'. In: *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. TABLEAUX 2023 - 32nd International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. Vol. 14278. LNCS - LNAI - Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence. Prague, Czech Republic: Springer Nature Switzerland, 14th Sept. 2023, pp. 203–222. DOI: 10.1007/978-3-031-43513-3_12. URL: https://hal.science/hal-04308897.

Doctoral dissertations and habilitation theses

[11] A. Allioux. 'Higher Structures in Homotopy Type Theory'. Université Paris Cité, 17th July 2023. URL: https://theses.hal.science/tel-04335842.

[12] A. Đurić. 'Quadratic normalisations and coherent presentations of monoids'. Université Paris Cité, 13th Apr. 2023. URL: https://hal.science/tel-04185552.

Reports & preprints

[13] E. Bauer and A. Saurin. *Cut-elimination for the circular modal mu-calculus: linear logic and super exponentials to the rescue*. 8th Mar. 2024. URL: https://hal.science/hal-04496648.

[14] H. Herbelin and D. Ilik. *An analysis of the constructive content of Henkin's proof of Gödel's completeness theorem*. 21st Jan. 2024. URL: https://inria.hal.science/hal-04408312.

[15] H. Herbelin and R. Ramachandra. *A parametricity-based formalization of semi-simplicial and semi-cubical sets*. 24th Jan. 2023. URL: https://inria.hal.science/hal-03963929.

[16] P.-A. Melliès and N. Zeilberger. *The categorical contours of the Chomsky-Schützenberger representation theorem*. 29th Dec. 2023. URL: https://hal.science/hal-04399404.

[17] A. Saurin. *A linear perspective on cut-elimination for non-wellfounded sequent calculi with least and greatest fixed points (extended version)*. 2023. URL: https://hal.science/hal-04169137.

12.2 Cited publications

[18] A. Das, A. De and A. Saurin. 'Decision Problems for Linear Logic with Least and Greatest Fixed Points'. In: 7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022). 7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022). Haifa, Israel, 2nd Aug. 2022. DOI: 10.4230/LIPIcs.FSCD.2022.20. URL: https://hal.science/hal-03867393.

[19] M. Abadi and G. D. Plotkin. 'A simple differentiable programming language'. In: *Proc. ACM Program. Lang.* 4.POPL (2020), 38:1–38:28. DOI: 10.1145/3371106. URL: https://doi.org/10.1145/3371106.

[20] A. Aizawa and M. Kohlhase. 'Mathematical Information Retrieval'. In: *Evaluating Information Retrieval and Access Tasks: NTCIR's Legacy of Research Impact*. Ed. by T. Sakai, D. W. Oard and N. Kando. Singapore: Springer Singapore, 2021, pp. 169–185. DOI: 10.1007/978-981-15-5554-1_12. URL: https://doi.org/10.1007/978-981-15-5554-1_12.

[21] A. A. de Amorim, M. Gaboardi, J. Hsu and S.-y. Katsumata. 'Probabilistic Relational Reasoning via Metrics'. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. IEEE, 2019, pp. 1–19. DOI: 10.1109/LICS.2019.8785715. URL: https://doi.org/10.1109/LICS.2019.8785715.

[22] C. Angiuli, E. Morehouse, D. R. Licata and R. Harper. 'Homotopical patch theory'. In: *J. Funct. Program.* 26 (2016), e18. DOI: 10.1017/S0956796816000198. URL: https://doi.org/10.1017/S0956796816000198.

[23] J. Baez and A. Crans. 'Higher-dimensional algebra. VI. Lie 2-algebras'. In: *Theory Appl. Categ.* 12 (2004), pp. 492–538.

[24] J. Bao, S. Docherty, J. Hsu and A. Silva. 'A Logic to Reason about Dependence and Independence'. In: *CoRR* abs/2008.09231 (2020). arXiv: 2008.09231. URL: https://arxiv.org/abs/2008.09231.

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

[26] B. Barras, C. Tankink and E. Tassi. 'Asynchronous Processing of Coq Documents: From the Kernel up to the User Interface'. In: *Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015, Proceedings*. Ed. by C. Urban and X. Zhang. Vol. 9236. Lecture Notes in Computer Science. Springer, 2015, pp. 51–66. DOI: 10.1007/978-3-319-22102-1_4. URL: https://doi.org/10.1007/978-3-319-22102-1%5C_4.

[27] G. Barthe, J. Hsu and K. Liao. 'A probabilistic separation logic'. In: *Proc. ACM Program. Lang.* 4.POPL (2020), 55:1–55:30. DOI: 10.1145/3371123. URL: https://doi.org/10.1145/3371123.

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

[29] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development, Coq'Art:the Calculus of Inductive Constructions*. Springer-Verlag, 2004.

[30] F. Bonchi, B. König and D. Petrisan. 'Up-To Techniques for Behavioural Metrics via Fibrations'. In: *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*. Ed. by S. Schewe and L. Zhang. Vol. 118. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 17:1–17:17. DOI: 10.4230/LIPIcs.CONCUR.2018.17. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2018.17.

[31] G. Bonfante and Y. Guiraud. 'Polygraphic Programs and Polynomial-Time Functions'. In: *Logical Methods in Computer Science* 5.2 (2009), pp. 1–37.

[32] N. de Bruijn. *AUTOMATH, a language for mathematics*. Tech. rep. 66-WSK-05. Technological University Eindhoven, Nov. 1968.

[33] A. Brunel, M. Gaboardi, D. Mazza and S. Zdancewic. 'A Core Quantitative Coeffect Calculus'. In: *Programming Languages and Systems*. Ed. by Z. Shao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 351–370.

[34] A. Brunel, D. Mazza and M. Pagani. 'Backpropagation in the simply typed lambda-calculus with linear negation'. In: *Proc. ACM Program. Lang.* 4.POPL (2020), 64:1–64:27. DOI: 10.1145/3371132. URL: https://doi.org/10.1145/3371132.

[35] A. Burroni. 'Higher-dimensional word problems with applications to equational logic'. In: *Theoretical Computer Science* 115.1 (July 1993), pp. 43–62.

[36] Y. Cai, P. G. Giarrusso, T. Rendel and K. Ostermann. 'A theory of changes for higher-order languages: incrementalizing λ-calculi by static differentiation'. In: *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*. Ed. by M. F. P. O'Boyle and K. Pingali. ACM, 2014, pp. 145–155. DOI: 10.1145/2594291.2594304. URL: https://doi.org/10.1145/2594291.2594304.

[37] K. Chardonnet. 'Towards a Curry-Howard Correspondence for Quantum Computation'. PhD thesis. Jan. 2023.

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

[39] A. Chlipala. *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*. The MIT Press, 2013.

[40] A. Church. 'A set of Postulates for the foundation of Logic'. In: *Annals of Mathematics* 2 (1932), pp. 33, 346–366.

[41] T. Coquand. 'Une théorie des Constructions'. Dissertation. University Paris 7, Jan. 1985.

[42] T. Coquand and G. Huet. 'Constructions : A Higher Order Proof System for Mechanizing Mathematics'. In: *EUROCAL'85*. Vol. 203. Lecture Notes in Computer Science. Linz: Springer Verlag, 1985.

[43] T. Coquand and C. Paulin-Mohring. 'Inductively defined types'. In: *Proceedings of Colog'88*. Ed. by P. Martin-Löf and G. Mints. Vol. 417. Lecture Notes in Computer Science. Springer Verlag, 1990.

[44] H. B. Curry, R. Feys and W. Craig. §9E.

[45] A. Das, A. De and A. Saurin. 'Comparing Infinitary Systems for Linear Logic with Fixed Points'. In: *43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2023, December 18-20, 2023, IIIT Hyderabad, Telangana, India*. Ed. by P. Bouyer and S. Srinivasan. Vol. 284. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 40:1–40:17. DOI: 10.4230/LIPICS.FSTTCS.2023.40. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2023.40.

[46] P. Deligne. 'Action du groupe des tresses sur une catégorie'. In: *Invent. Math.* 128.1 (1997), pp. 159–175.

[47] T. Ehrhard. 'An introduction to Differential Linear Logic: proof-nets, models and antiderivatives'. In: *CoRR* abs/1606.01642 (2016). arXiv: 1606.01642. URL: http://arxiv.org/abs/1606.01642.

[48] M. Gaboardi, A. Haeberlen, J. Hsu, A. Narayan and B. C. Pierce. 'Linear dependent types for differential privacy'. In: *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*. Ed. by R. Giacobazzi and R. Cousot. ACM, 2013, pp. 357–370. DOI: 10.1145/2429069.2429113. URL: https://doi.org/10.1145/2429069.2429113.

[49] T. Gauthier, C. Kaliszyk, J. Urban, R. Kumar and M. Norrish. 'TacticToe: Learning to Prove with Tactics'. In: *Journal of Automated Reasoning* (2020). DOI: 10.1007/s10817-020-09580-x. URL: https://doi.org/10.1007/s10817-020-09580-x.

[50] G. Gentzen. 'Untersuchungen uber das logische Schliessen'. In: *Mathematische Zeitschrift* 39 (1935), pp. 176–210, 405–431.

[51] 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*. Ed. by J. Fenstad. Studies in Logic and the Foundations of Mathematics 63. North Holland, 1971, pp. 63–92.

[52] G. Gonthier. 'Engineering mathematics: the odd order theorem proof'. In: *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*. Ed. by R. Giacobazzi and R. Cousot. ACM, 2013, pp. 1–2. DOI: 10.1145/2429069.2429071. URL: https://doi.org/10.1145/2429069.2429071.

[53] G. Gonthier. 'Formal proof–the four-color theorem'. In: *Notices of the AMS* 55.11 (2008), pp. 1382–1393.

[54] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. L. Roux, A. Mahboubi, R. O'Connor, S. O. Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi and L. Théry. 'A Machine-Checked Proof of the Odd Order Theorem'. In: *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*. Ed. by S. Blazy, C. Paulin-Mohring and D. Pichardie. Vol. 7998. Lecture Notes in Computer Science. Springer, 2013, pp. 163–179. DOI: 10.1007/978-3-642-39634-2_14. URL: https://doi.org/10.1007/978-3-642-39634-2%5C_14.

[55] Y. Guiraud. 'Présentations d'opérades et systèmes de réécriture'. PhD thesis. Univ. Montpellier~2, 2004.

[56] Y. Guiraud. 'Termination Orders for 3-Dimensional Rewriting'. In: *Journal of Pure and Applied Algebra* 207.2 (2006), pp. 341–371.

[57] Y. Guiraud. 'The Three Dimensions of Proofs'. In: *Annals of Pure and Applied Logic* 141.1–2 (2006), pp. 266–295.

[58] Y. Guiraud. 'Two Polygraphic Presentations of Petri Nets'. In: *Theoretical Computer Science* 360.1–3 (2006), pp. 124–146.

[59] G. G. Hillebrand and P. C. Kanellakis. 'On the Expressive Power of Simply Typed and Let-Polymorphic Lambda Calculi'. In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 1996, pp. 253–263. DOI: 10.1109/LICS.1996.561337. URL: https://doi.org/10.1109/LICS.1996.561337.

[60] M. Hofmann and T. Streicher. 'The groupoid interpretation of type theory'. In: *Twenty-five years of constructive type theory (Venice, 1995)*. Vol. 36. Oxford Logic Guides. Oxford Univ. Press, New York, 1998, pp. 83–111.

[61] W. A. Howard. 'The formulae-as-types notion of constructions'. In: *to H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Unpublished manuscript of 1969. Academic Press, 1980.

[62] Y. Lafont. 'Towards an Algebraic Theory of Boolean Circuits'. In: *Journal of Pure and Applied Algebra* 184 (2003), pp. 257–310.

[63] P. Malbos. 'Critères de finitude homologique pour la non convergence des systèmes de réécriture de termes'. PhD thesis. Univ. Montpellier~2, 2004.

[64] P. Martin-Löf. *A theory of types*. Tech. rep. 71-3. University of Stockholm, 1971.

[65] S. Mimram and C. D. Giusto. 'A Categorical Theory of Patches'. In: *Proceedings of the Twenty-ninth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2013, New Orleans, LA, USA, June 23-25, 2013*. Ed. by D. Kozen and M. W. Mislove. Vol. 298. Electronic Notes in Theoretical Computer Science. Elsevier, 2013, pp. 283–307. DOI: 10.1016/j.entcs.2013.09.018. URL: https://doi.org/10.1016/j.entcs.2013.09.018.

[66] B. C. Pierce, A. A. de Amorim, C. Casinghino, M. Gaboardi, M. Greenberg, C. Hriţcu, V. Sjöberg and B. Yorgey. *Logical Foundations*. Software Foundations series, volume 1. Version 5.5. http://www.cis.upenn.edu/~bcpierce/sf. Electronic textbook, May 2018.

[67] T. U. F. Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

[68] J. C. Reynolds. 'Towards a theory of type structure'. In: *Symposium on Programming*. Ed. by B. Robinet. Vol. 19. Lecture Notes in Computer Science. Springer, 1974, pp. 408–423.

[69] T. Ringer, A. Sanchez-Stern, D. Grossman and S. Lerner. 'REPLica: REPL Instrumentation for Coq Analysis'. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2020. New Orleans, LA, USA: Association for Computing Machinery, 2020, pp. 99–113. DOI: 10.1145/3372885.3373823. URL: https://doi.org/10.1145/3372885.3373823.

[70] S. Salvati. 'Recognizability in the Simply Typed Lambda-Calculus'. In: *Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-24, 2009. Proceedings*. Ed. by H. Ono, M. Kanazawa and R. J. G. B. de Queiroz. Vol. 5514. Lecture Notes in Computer Science. Springer, 2009, pp. 48–60. DOI: 10.1007/978-3-642-02261-6_5. URL: https://doi.org/10.1007/978-3-642-02261-6%5C_5.

[71] A. Simpson. 'Category-theoretic Structure for Independence and Conditional Independence'. In: *Proceedings of the Thirty-Fourth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2018, Dalhousie University, Halifax, Canada, June 6-9, 2018*. Ed. by S. Staton. Vol. 341. Electronic Notes in Theoretical Computer Science. Elsevier, 2018, pp. 281–297. DOI: 10.1016/j.entcs.2018.03.028. URL: https://doi.org/10.1016/j.entcs.2018.03.028.

[72] A. Spiwack. 'Verified Computing in Homological Algebra. (Calculs vérifiés en algèbre homologique)'. PhD thesis. École Polytechnique, Palaiseau, France, 2011. URL: https://tel.archives-ouvertes.fr/pastel-00605836.

[73] R. Street. 'Limits Indexed by Category-Valued 2-Functors'. In: *Journal of Pure and Applied Algebra* 8 (1976), pp. 149–181.

[74] T. C. D. Team. *The Coq Proof Assistant, version 8.7.1*. Dec. 2017. DOI: 10.5281/zenodo.1133970. URL: https://doi.org/10.5281/zenodo.1133970.

[75] M. Wenzel. 'Interaction with Formal Mathematical Documents in Isabelle/PIDE'. In: July 2019, pp. 1–15. DOI: 10.1007/978-3-030-23250-4_1.

[76] T. Zimmermann and 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, Sept. 2019. URL: https://hal.inria.fr/hal-01951176.