RESEARCH CENTRE

Inria Paris Center

2022

ACTIVITY REPORT

Team

PI.R2

Design, study and implementation of languages for proofs and programs

Inria teams are typically groups of researchers working on the definition of a common project, and objectives, with the goal to arrive at the creation of a project-team. Such project-teams may include other partners (universities or research institutions)

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Inria

Contents

Team PI.R2

Creation of the Team: 2022 November 30

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

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]
- 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]
- Alexis Saurin [CNRS, Researcher]

Faculty Members

- Pierre Letouzey [Université Paris Cité, Associate Professor]
- Daniela Petrisan [Université Paris Cité, Associate Professor]

PhD Students

- Esaïe Bauer [Université Paris Cité]
- Thomas Binetruy [EQUISAFE]
- Vincent Blazy [Université Paris Cité]
- Félix Castro [Université Paris Cité]
- El Cherradi [CGEIET]
- Abhishek De [INRIA]
- Alen Duric [Université Paris Cité]
- Farzad Jafar Rahmani [Université Paris Cité]
- Moana Jubert [INRIA, from Oct 2022]
- Hugo Moeneclaey [Université Paris Saclay]
- Vincent Moreau [Université Paris Cité]

Technical Staff

- Daniel De Rauglaudre [INRIA, until Jul 2022]
- Thierry Martinez [INRIA]
- Théo Zimmermann [INRIA, until Nov 2022]

Administrative Assistant

- Anne Mathurin [INRIA]

Todo list

2 Overall objectives

Since 2012, the research conducted in πr^2 has been 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. The team also expanded 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.

2021 is the final year of the project-team which shall be replaced, early 2022, by a newly created team, named PiCube, welcoming new members and exploring new research directions which will be presented in this report.

3 Research program

3.1 Proof theory and the Curry-Howard correspondence

3.1.1 Proofs as programs

Proof theory is the branch of logic devoted to the study of the structure of proofs. An essential contributor to this field is Gentzen [61] who developed in 1935 two logical formalisms that are now central to the study of proofs. These are the so-called "natural deduction", a syntax that is particularly well-suited to simulate the intuitive notion of reasoning, and the so-called "sequent calculus", a syntax with deep geometric properties that is particularly well-suited for proof automation.

Proof theory gained a remarkable importance in computer science when it became clear, after genuine observations first by Curry in 1958 [56], then by Howard and de Bruijn at the end of the 60's [73, 48], 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 [53, 54] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [90].

3.1.2 Towards the calculus of constructions

The λ-calculus, defined by Church [52], 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 [44].

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 [82].

In 1985, Coquand and Huet [53, 54] in the Formel team of INRIA-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [62, 86]. 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 [55] 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 [91] 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 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 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 [45], the more advanced Certified Programming with Dependent Types book by Chlipala [51] 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, 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.

3.3 Dependently typed programming languages

Dependently typed programming (shortly DTP) is an emerging concept referring to the diffuse and broadening tendency to develop programming languages with type systems able to express program properties finer than the usual information of simply belonging to specific data-types. The type systems of dependently-typed programming languages allow to express properties *dependent* of the input and the output of the program (for instance that a sorting program returns a list of same size as its argument). Typical examples of such languages were the Cayenne language, developed in the late 90's at Chalmers University in Sweden and the DML language developed at Boston. Since then, various new tools have been proposed, either as typed programming languages whose types embed equalities (Ωmega at Portland, ATS at Boston, ...) or as hybrid logic/programming frameworks (Agda at Chalmers University, Twelf at Carnegie, Delphin at Yale, OpTT at U. Iowa, Epigram at Nottingham, ...).

DTP contributes to a general movement leading to the fusion between logic and programming. Coq, whose language is both a logic and a programming language which moreover can be extracted to pure

ML code plays a role in this movement and some frameworks combining logic and programming have been proposed on top of Coq (Concoqtion at Rice and Colorado, Ynot at Harvard, Why in the ProVal team at INRIA, Iris at MPI-Saarbrucken). It also connects to Hoare logic, providing frameworks where pre- and post-conditions of programs are tied with the programs.

DTP approached from the programming language side generally benefits of a full-fledged language (e.g. supporting effects) with efficient compilation. DTP approached from the logic side generally benefits of an expressive specification logic and of proof methods so as to certify the specifications. The weakness of the approach from logic however is generally the weak support for effects or partial functions.

3.3.1 Type-checking and proof automation

In between the decidable type systems of conventional data-types based programming languages and the full expressiveness of logically undecidable formulae, an active field of research explores a spectrum of decidable or semi-decidable type systems for possible use in dependently typed programming languages. At the beginning of the spectrum, this includes, for instance, the system F's extension ML_F of the ML type system or the generalisation of abstract data types with type constraints (G.A.D.T.) such as found in the Haskell programming language. At the other side of the spectrum, one finds arbitrary complex type specification languages (e.g. that a sorting function returns a list of type "sorted list") for which more or less powerful proof automation tools exist – generally first-order ones.

3.4 Around and beyond the Curry-Howard correspondence

For two decades, the Curry-Howard correspondence has been limited to the intuitionistic case but since 1990, an important stimulus spurred on the community following Griffin's discovery that this correspondence was extensible to classical logic. The community then started to investigate unexplored potential connections between computer science and logic. One of these fields is the computational understanding of Gentzen's sequent calculus while another one is the computational content of the axiom of choice.

3.4.1 Control operators and classical logic

Indeed, a significant extension of the Curry-Howard correspondence has been obtained at the beginning of the 90's thanks to the seminal observation by Griffin [63] that some operators known as control operators were typable by the principle of double negation elimination ($\neg\neg A \Rightarrow A$), a principle that enables classical reasoning.

Control operators are used to jump from one location of a program to another. They were first considered in the 60's by Landin [79] and Reynolds [85] and started to be studied in an abstract way in the 80's by Felleisen *et al* [59], leading to Parigot's $\lambda\mu$-calculus [84], a reference calculus that is in close Curry-Howard correspondence with classical natural deduction. In this respect, control operators are fundamental pieces to establish a full connection between proofs and programs.

3.4.2 Sequent calculus

The Curry-Howard interpretation of sequent calculus started to be investigated at the beginning of the 90's. The main technicality of sequent calculus is the presence of *left introduction* inference rules, for which two kinds of interpretations are applicable. The first approach interprets left introduction rules as construction rules for a language of patterns but it does not really address the problem of the interpretation of the implication connective. The second approach, started in 1994, interprets left introduction rules as evaluation context formation rules. This line of work led in 2000 to the design by Hugo Herbelin and Pierre-Louis Curien of a symmetric calculus exhibiting deep dualities between the notion of programs and evaluation contexts and between the standard notions of call-by-name and call-by-value evaluation semantics.

3.4.3 Abstract machines

Abstract machines came as an intermediate evaluation device, between high-level programming languages and the computer microprocessor. The typical reference for call-by-value evaluation of λ-calculus is Landin's SECD machine [80] and Krivine's abstract machine for call-by-name evaluation [76, 75]. A typical abstract machine manipulates a state that consists of a program in some environment of bindings and some evaluation context traditionally encoded into a "stack".

3.4.4 Delimited control

Delimited control extends the expressiveness of control operators with effects: the fundamental result here is a completeness result by Filinski [60]: any side-effect expressible in monadic style (and this covers references, exceptions, states, dynamic bindings, ...) can be simulated in λ-calculus equipped with delimited control.

3.5 Effective higher-dimensional algebra

3.5.1 Higher-dimensional algebra

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

In the last decades, the importance of higher-dimensional categories has grown fast, mainly with the new trend of *categorification* that currently touches algebra and the surrounding fields of mathematics. Categorification is an informal process that consists in the study of higher-dimensional versions of known algebraic objects (such as higher Lie algebras in mathematical physics [43]) 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 [57]).

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

3.5.2 Higher-dimensional rewriting

Higher-dimensional categories are algebraic structures that contain, in essence, computational aspects. This has been recognised by Street [89], and independently by Burroni [49], 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 [77, 81, 64, 65], and beyond: Petri nets [67] and formal proofs of classical and linear logic have been expressed in this framework [66]. 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 [46].

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 [40] and Squier [87, 88]: Squier has proved that, if a monoid M can be presented by a *finite*, *terminating* and *confluent* rewriting system, then its third integral homology group $H_3(M,\mathbb{Z})$ is finitely generated and the monoid M has *finite derivation type* (a property of homotopical nature). This allowed him to conclude that finite convergent rewriting systems were not a universal solution to decide the word problem of finitely generated monoids. Since then, Yves Guiraud and Philippe Malbos have shown that this connection was part of a deeper unified theory when formulated in the higher-dimensional setting [12, 13], [71, 68, 70].

In particular, the computational content of Squier's proof has led to a constructive methodology to produce, from a convergent presentation, *coherent presentations* and *polygraphic resolutions* of algebraic structures, such as monoids [12] and algebras [11]. 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 [78, 41]). 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 [8], and in homological algebra [69][11]. See [10] for a summary of these results.

4 Application domains

The application domains of the 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 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 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

6.1 Creation of the Picube INRIA research team

One main event and highlight of this year 2022 has been the creation of the Picube team the 1st of December, after its formal presentation to the project committee in May, and its official validation in November after examination by a committee of experts from INRIA, CNRS and Université Paris Cité. As it was already the case with the PiR2 team, the new Picube team is a joint research team between

Université Paris Cité, INRIA Paris and the CNRS. The Picube team is hosted by the Institut de Recherche en Informatique Fondamentale (IRIF) lab and all researchers of the team are thus members of the IRIF lab. The purpose of the Picube research team is to take advantage of the most recent advances in

- type theory and foundations of mathematics: homotopy type theory, realizability and forcing, differential linear logic,
- programming language semantics: computational effects, differential and probabilistic programming,
- architecture and design of proof assistants: formalisation of mathematics, unification and symbolic elaboration techniques

in order to reduce the gap which currently separates the the vernacular language used by the working 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 of the PiR2 team, and integrates to it a number of new ingredients in the direction of certified mathematics, differential and probabilistic programming, and machine learning. The project is structured into five scientific areas of focus:

1. **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 the Working Mathematician,**
4. **Formalisation and Linguistics of Mathematics,**
5. **Higher Dimensional Algebra and Synthetic Homotopy Theory.**

Except for Yves Guiraud (CR INRIA) who decided to join the Ouragan INRIA team and IMJ-PRG, all the former members of PiR2 are also members of the Picube team. The Picube research team incorporates moreover three new members: Thomas Ehrhard (DR CNRS), Daniela Petrişan (MdC Université Paris Cité) and Paul-André Melliès (DR CNRS) who will become the team leader. Thomas Ehrhard is a specialist of proof theory and programming language semantics. He will bring to the team his deep understanding of the emerging connections between linear logic and functional analysis, with promising connections to differential and probabilistic programming. Daniela Petrişan is a specialist of categorical and topological methods in logic, automata theory and programming language semantics. She will bring to the team her expertise on topological and metric interpretations of type theory and process calculi, with the hope of building a unified framework integrating coalgebraic methods and dependent type theory. Paul-André Melliès is a specialist of category theory and programming language semantics, with a fascination for the numerous emerging connections between Martin-Löf type theory, homotopy theory, linear logic, game semantics and higher-order automata theory.

In the continuation of the work of the PiR2 team, the Picube research team will contribute to the education of new generations of students taking the lead in proof assistant technology and formalisation of mathematics and computer science. We will 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.

The work of the team was affected by several problems:

1. Dysfunctions at Inria at large such as for instance
    - Poor functioning of Eksae that complicated significantly the work of our administrative assistant and the preparation of the budget,

- Very late reimbursement of travel expenses (for instance one of our PhD got his travel expenses reimbursed after nearly one year),

2. Research is less and less considered as our main mission.
3. The veiled and constant criticisms on our Evaluation Committee we are all proud of.
4. The uncertainty about the future of the Institute which is not able to fulfill legal obligations such as providing its social report in due time or which does not communicate accurately about its budget, except when publicly calling it "intenable" (meaning "unsustainable")."

7 New software and platforms

7.1 New software

7.1.1 Coq

Name: 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 integrated development environment (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 Contributions: Coq version 8.16 integrates changes to the Coq kernel and performance improvements along with a few new features. We highlight some of the most impactful changes here:

The guard checker (see Guarded) now ensures strong normalization under any reduction strategy.

Irrelevant terms (in the SProp sort) are now squashed to a dummy value during conversion, fixing a subject reduction issue and making proof conversion faster.

Introduction of reversible coercions, which allow coercions relying on meta-level resolution such as type-classes or canonical structures. Also allow coercions that do not fullfill the uniform inheritance condition.

Generalized rewriting support for rewriting with Type-valued relations and in Type contexts, using the Classes.CMorphisms library.

Added the boolean equality scheme command for decidable inductive types.

Added a Print Notation command.

Incompatibilities in name generation for Program obligations, eauto treatment of tactic failure levels, use of ident in notations, parsing of module expressions.

Standard library reorganization and deprecations.

Improve the treatment of standard library numbers by Extraction.

See https://coq.inria.fr/refman/changes.html#version-8-16 for a detailed changelog.

News of the Year: Coq version 8.16 integrates changes to the Coq kernel and performance improvements along with a few new features. See the detailed changes at https://coq.inria.fr/refman/changes.html#version-8-16 for an overview of the new features and changes, along with the full list of contributors.

URL: http://coq.inria.fr/

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 Pedrot, Clément Pit-Claudel, Kazuhiko Sakaguchi, Vincent Semeria, Michael Soegtrop, Arnaud Spiwack, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Anton Trunov, Li-Yao Xia, Theo Zimmermann, Gaetan Gilbert

Partners: CNRS, Université Paris-Sud, ENS Lyon, Université Paris-Diderot

7.1.2 Rewr

Name: 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.

URL: https://www.imj-prg.fr/~yves.guiraud/programmes/rewr

Publications: hal-00326974, hal-00531242, hal-00682233, hal-00818253, hal-00932845, hal-01141226

Contact: Yves Guiraud

Participants: Yves Guiraud, Samuel Mimram

7.1.3 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.

URL: https://www.imj-prg.fr/~yves.guiraud/programmes/catex

Contact: Yves Guiraud

Participant: Yves Guiraud

7.1.4 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.

URL: https://www.imj-prg.fr/~yves.guiraud/programmes/cox

Publications: hal-00682233, hal-00818253

Contact: Yves Guiraud

Participant: Yves Guiraud

7.1.5 coqbot

Keywords: Web API, Automation, Software engineering

Functional 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.

Release Contributions: The Julien Coolen's internship final release.

Added

Integrate with Jason Gross' coq-bug-minimizer tool. Merge a branch in the coq repository if some conditions are met, by writing @coqbot: merge now in a comment. Parameterize the bot with a configuration file. Installation as a GitHub App is supported. Report CI status checks with the Checks API when using the GitHub app. Report errors of jobs in allow failure mode when the Checks API is used.

Changed

Refactored the architecture of the application and of the bot-components library Always create a merge commit when pushing to GitLab. More informative bot merge commit title for GitLab CI.

URL: https://github.com/coq/bot/

Contact: Theo Zimmermann

7.1.6 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 Contributions: - Port to Coq 8.14 - Improved packaging system for libraries - Improved display and interaction - Settings panel

URL: https://github.com/ejgallego/jscoq

Publication: hal-01425752

Contact: Emilio Jesus Gallego Arias

Participants: Emilio Jesus Gallego Arias, Shachar Itzhaky

Partners: Mines ParisTech, Technion, Israel Institute of Technology

7.1.7 coq-serapi

Keywords: Interaction, Coq, Ocaml, Data centric, User Interfaces, GUI (Graphical User Interface), Toolkit

Scientific 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).

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 Contributions: - Support for Coq 8.15 -

News of the Year: In 2021, SerAPI has seen a sizable increase of users, and many bugs and improvements have been made in response to users requests. Also, SerAPI has been updated to support Coq 8.13, 8.14, and 8.15 versions.

URL: https://github.com/ejgallego/coq-serapi

Publication: hal-01384408

Contact: Emilio Jesus Gallego Arias

Participants: Karl Palmskog, Theo Zimmermann, Shachar Itzhaky, Jason Gross

Partner: KTH Royal Institute of Technology

7.1.8 pyCoq

Keywords: Coq, Python

Functional Description: PyCoq is a set of bindings and libraries allowing to interact with the Coq interactive proof assistant from inside Python 3.

Release Contributions: Initial release

URL: https://github.com/ejgallego/pycoq

Contact: Emilio Jesus Gallego Arias

Participant: Thierry Martinez

8 New results

8.1 Effects in proof theory and programming

Participants: Félix Castro, Emilio Jesús Gallego Arias, Hugo Herbelin, Jean-Jacques Lévy, Paul-André Melliès, Alexis Saurin.

8.1.1 An analysis of the constructive content of Henkin's proof of Gödel's completeness theorem

Together with Ilik, Herbelin submitted a paper analyzing the constructive content of Henkin's proof of Gödel's completeness theorem

Gödel's completeness theorem for classical first-order logic is one of the most basic theorems of logic and Henkin's proof method is probably the most widely taught. Central to any foundational course in logic, it connects the notion of valid formula to the notion of provable formula. In this paper, they survey a few standard formulations and proofs of this completeness theorem before focusing on the formal description of a slight modification of Henkin's proof within intuitionistic second-order arithmetic.

In the context of the completeness of intuitionistic logic with respect to various semantics such as Kripke or Beth semantics, it is standard to follow the Curry-Howard correspondence and to interpret the proofs of completeness as programs which turn proofs of validity for these semantics into proofs of derivability. They apply this approach to Henkin's proof to phrase it as a program which transforms any proof of validity with respect to Tarski semantics into a proof of derivability. This sheds an "effective" light on the relation between Tarski semantics and syntax: proofs of validity are syntactic objects that we can manipulate and compute with, just like ordinary syntax.

8.1.2 Differentials and distances in probabilistic coherence spaces

Ehrhard published a paper developing the differential aspects of probabilistic coherence spaces, a denotational model of Linear Logic which provides a faithful account of stochastic programs. In this model programs are represented as analytic functions which can be written as powerseries with non-negative coefficients and such functions can be deriveted an arbitrary number of times, whatever be their type. Thomas Ehrhard explored two related applications of the corresponding derivatives. First he showed how derivatives allow to compute the expectation of execution time in the weak head reduction of probabilistic PCF (pPCF). Next he applied a general notion of "local" differential of morphisms to the proof of a Lipschitz property of these morphisms allowing in turn to relate the observational distance on pPCF terms to a distance the model is naturally equipped with. This suggests that extending probabilistic programming languages with derivatives, in the spirit of the differential lambda-calculus, could be quite meaningful.

In more recent developments, currently submitted [58], Ehrhard has developed a categorical and syntactical framework for such differential models of Linear Logic, where addition is only partially defined: the fundamental observation is that, even if the differential calculus requires addition as it is well known, one does not need all of them and many models of Linear Logic feature enough additions for hosting a fully-fledged differential calculus. This shows that, contrarily to what was believed earlier, differential Linear Logic and the differential lambda-calculus are compatible with deterministic computations.

8.1.3 Layered and Object-Based Game Semantics

Large-scale software verification relies critically on the use of compositional languages, semantic models, specifications, and verification techniques. In collaboration with Zhong Shao's CertiKOS group (Yale) and Léo Stefanesco (MPI Kaiserslautern), Melliès has developed [23] a layered and object-based game semantics based on coherence spaces. In the resulting game semantics of low-level concurrent code, every program is interpreted as a concurrent and possibly non-deterministic strategy connecting an underlay signature to an overlay signature. The interpretation of the low-level code relies on a non-commutative form $A \mapsto \dagger A$ of the exponential modality of Linear Logic, equipped with permutations in order to encode concurrency, revisiting an idea promoted by Uday Reddy in the 1990s. After formulating layer implementations as regular maps between object spaces, a notion of concurrent object space is designed, where sequential traces may be identified modulo permutation of independent operations. As illustrated by a ticket lock implementation, the model is used to express protected shared object concurrency, in a simple model based on regular maps between concurrent object spaces. This work aims at developing a compositional model for certified abstraction layers which can be used to build certified heterogeneous systems such as CertiKOS, taking advantage of the compositionality principles of denotational semantics.

8.1.4 Tracking Redexes in the Lambda Calculus

Levy wrote a book chapter[36] to be published in 202 in which he reviews notions of residuals of redexes to keep track of redexes along reductions in the lambda calculus and families of redexes keep track of redexes created along these reductions. He discusses how their relation to a labeled-calculus and extends these properties to combinatory logic, term rewriting systems, process calculi and proofnets of linear logic.

8.1.5 Controlling sequentialization in proof nets

In a collaboration with Aurore Alcolei and Luc Pellissier, Alexis Saurin internalised the notion of jumps of linear logic proof-nets (which can be used as an alternative to boxes) in a slight extension of MLL. 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. In this still ongoing work, Alcolei, Pellissier and Saurin analyzed the logical strength of jumps by internalizing them in an extention of MLL where axioms on a specific formula introduce constraints on the possible sequentializations. The jumping formula needs

to be treated non-linearly, which they do either axiomatically, or by embedding it in a very controlled fragment of multiplicative-exponential linear logic, uncovering the exponential logic of sequentialization.

8.1.6 Algebraic Logic Programming

Emilio J. Gallego Arias and Jim Lipton continued work on algebraic models of proof search, in particular they have developed a notion of *step-indexed* tabular alegory which provides an improved semantic setting for the proof search machine developed in Gallego's PhD.

Jim Lipton visited the team in the summer 2022.

8.2 Reasoning and programming with infinite data

Participants: Esaie Bauer, Kostia Chardonnet, Abhishek De, Thomas Ehrhard, Farzad Jafarrahmani, Alexis Saurin.

8.2.1 Proof theory of non-wellfounded and circular proofs

Validity conditions of infinitary and circular proofs and cut-elimination. In collaboration with David Baelde, Amina Doumane and Denis Kuperberg, Alexis Saurin published at LICS 2022 [25] a new validity criterion for circular and non-wellfoudned proofs that extends the original validity condition considered by Baelde, Doumane and Saurin in CSL 2016 [1]. This new condition is better-behaved wrt. the cut rules in that it takes into account the cut-axiom interaction in sequent proofs, allowing progressing threads to "bounce" on axioms and cut. This much more flexible criterion takes inspirations in Girard's geometry of interaction and works on additive proof-nets. The paper establishes cut-elimination and study the decidability properties of the validity condition. While the full bouncing validity is undecidable, they exhibit a hierarchy of criteria "of bounded heights" which are all decidable and the union of which corresponds to bouncing validity (which is, therefore, semi-decidable)

Alexis Saurin generalized the cut-elimination-theorem for non-wellfounded proofs of multiplicative additive linear logic with least and greatest fixed points (μMALL) that he obtained with David Baelde and Amina Doumane [42] to full linear logic, accomodating a treatment of the exponentials. This goes by a fixed-point encoding of LL exponentials which allowed him, using results from infinitary rewriting, to lift the cut-elimination result to for μLL. Moreover, designing embedding of μLJ and μLK in μLL he obtained similar cut-elimination theorems for those logics as well. A paper has been submitted early 2023.

Proof nets for non-wellfounded proofs. Abhishek De completed his PhD on proof-nets for circular and non-wellfounded proofs, that he defended in december 1st.

Decision problems of linear logic with fixed points. In a collaboration with Anupam Das, Abhishek De and Alexis Saurin investigated the decision problems for variants of linear logic with fixed-points. Decision problems for fragments of linear logic exhibiting 'infinitary' behaviour (such as exponentials) are notoriously complicated. In this work, they addressed the decision problems for variations of linear logic with fixed points (muMALL), in particular, recent systems based on 'circular' and 'non-wellfounded' reasoning. In particular, they show that muMALL is undecidable.

More explicitly, they show that the general non-wellfounded system is Π^0_1-hard via a reduction to the non-halting of Minsky machines. It is thus is strictly stronger than its circular counterpart which is in Σ^0_1. Moreover, they showed that the restriction of these systems to theorems with only the least fixed points is already Σ^0_1-complete via a reduction to the reachability problem of alternating vector addition systems with states. This implies that both the circular system and the finitary system (with explicit (co)induction) are Σ^0_1-complete. Those results were published at FSCD 2022 [26]

8.2.2 On the semantics of finitary and non-wellfounded proofs

Denotational semantics of finitary and non-wellfounded proofs. Thomas Ehrhard, Farzad Jafarrahmani and Alexis Saurin extended the previous work by Ehrhard and Jafarrahmani to polarized Linear Logic with fixed-points. One of their objectives is to develop Linear Logic foundations to inductive and coinductive types in Coq.

They also extended the denotational semantics of μLL to non-wellfounded proofs and are currently investigating how to specifically interpret circular derivations. A draft has been submitted.

Phase semantics for linear logic with fixed points. The truth semantics of linear logic (i.e. phase semantics) is often overlooked despite having a wide range of applications and deep connections with several denotational semantics. In phase semantics one is concerned about the provability of formulas rather than the contents of their proofs (or refutations).

Abhishek De, Farzad Jafarrahmani and Alexis Saurin extended the phase semantics of MALL to $\mu MALL$ with explicit (co)induction ($\mu MALL$), proving soundness and completeness theorems. The completeness theorem provides a cut-admissibility result for $\mu MALL$.

They also considered a constructive fragment that yields a Tait-style wellfounded system ($\mu MALL^{\omega}$) for which they defined the phase semantics together with soundness and completeness results and proved a cut-elimination theorem for this system. This paper was published at FSTTCS 2022 [27].

8.2.3 Quantum programming languages with inductive and coinductive types

Chardonnet's PhD research focuses on extending quantum programming languages with inductive and coinductive types, under the hypothesis of quantum control (as in QML [39] compared to classical control). In 2021, Chardonnet, Saurin and Valiron developed their work on a language 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 particular, they studied the expressiveness of a restricted validity condition for a class of type isomorphisms, relating it with the class of recursive primitive permutations by Paolini, Piccolo and Roversi. A preliminary version was presented in TLLA 2021. The extended new version will be published in CSL'23 [33].

8.3 Effective higher-dimensional algebra

Participants: Antoine Allioux, Pierre-Louis Curien, Alen Đurić.

8.3.1 Coherent presentations of monoids

The work of Alen Đurić, Pierre-Louis Curien and Yves Guiraud on coherent presentations of monoids admitting a Garside family has been submitted, and presented at the workshop "Braids and beyond" held in memory of Patrick Dehornoy in September 2021 [22].

8.3.2 Polygraphs and opetopes

Pierre-Louis Curien has found a new, elementary, proof of the isomorphism between many-to-one polygraphs on one hand, and opetopic sets on the other hand. This result had been proved quite indirectly by Harnik, Makkai, and Zawadowski in 2008. A more direct proof was given by Cédric Ho Tanh (former student of the team) in his PhD thesis (2019), with a reference to some results of Simon Henry. The new proof is entirely self-contained, and, more importantly, unveils invariants of the polygraphic syntax. It will be presented at the 2022 Workshop on Polynomial Functors to be held in April 2022 at the Topos Institute (virtually).

8.4 Metatheory and development of Coq

Participants: Vincent Blazy, Félix Castro, Emilio Jesús Gallego Arias, Hugo Herbelin, Pierre Letouzey, Thierry Martinez, Hugo Moeneclaey, Yann Régis-Gianas, Théo Zimmermann.

Vincent Blazy, Hugo Herbelin and Pierre Letouzey continued a work aiming at making explicit the universe subtyping in the Calculus of Constructions (PhD thesis of Vincent Blazy). The first goal is to detect more easily each use of the Prop-Type cumulativity in Coq, with potential application to Coq extraction and also to the mathematical foundations.

8.4.1 Dependent pattern-matching

Thierry Martinez carried on full 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. Together with Meven Bertrand and Hugo Herbelin, they almost reached the point of submitting a paper describing the implementation.

8.4.2 Software engineering aspects of the development of Coq

Théo Zimmermann was recruited in January 2020 on a three-year fixed term position to contribute both to the collaborative maintenance and evolution effort around Coq and its community, and to further investigate these software engineering aspects through empirical methods.

Théo Zimmermann is the initial author and main maintainer of coqbot, the bot used for everyday maintenance tasks in the Coq project. During the CoqDev project, the bot has been signicantly improved and has received contributions from Coq developers (Jason Gross, Pierre-Marie Pédrot, Gaetan Gilbert and Ali Caglayan more recently) as well as an intern that Théo Zimmermann supervised in 2020 (Julien Coolen, funded through the CoqDev project, now an engineer at Nomadic Labs). This has allowed to add many new features, in particular related to pull request testing and management (merging approved pull requests, closing stale pull requests, continuous integration with a reduced or extended suite of tests, including many external projects, etc.). The team published a paper about the followed approach [24] in IEEE Software (2022). The article describes some main features of the bot and the design choices that were made and how these design choices help with the maintenance and the evolution of the bot, by making it easier to adapt the bot to new use cases and to involve Coq contributors in the bot development.

Théo Zimmermann has also collaborated with Jason Gross (from MIT CSAIL) on integrating the bug minimizer created by Jason Gross in Coq's CI infrastructure, by relying on coqbot. This has allowed coqbot to automatically propose to produce reduced test cases for compatibility issues detected on external Coq projects by the continuous integration system, thus making it easier to understand what the compatibility issues are, when modifying Coq. Furthermore, this has allowed to conduct the first empirical evaluation of the bug minimizer and to improve it further based on the issues detected during this evaluation. They published their work, [28] at ITP 2022. This is the first formal publication on the bug minimizer itself.

Théo Zimmermann supervised in June 2021 the internship of Jérémy Damour, who was tasked with several contributions to the Hydras & Co. project of Pierre Castéran. This work resulted in a publication at the national conference JFLA 2022 [30].

Finally, Théo Zimmermann has coordinated an *ad hoc* working group to prepare and then analyze the Coq Community Survey. The survey was held in February 2022 and received 466 responses. It allowed to get an up-to-date picture of the Coq community and feedback on which to base future development decisions. Since the survey, the working group has been processing the responses and publishing partial results in the form of blog posts and a presentation at the Coq workshop by Ana Borges. Emilio J. Gallego Arias took an active part in this working group.

8.4.3 Software infrastructure and Tools

Emilio J. Gallego Arias continued work on revamping Coq's build system as to implement a workflow based on the state-of-the-art, industrial build system Dune. Many improvements were made including

porting the OCaml parts of Coq to Dune, which allowed the team to remove large parts of custom build code, and with Ali Caglayan, Coq's test suite was made incremental. Additionally, Emilio J. Gallego Arias coordinated the release of Dune version 2.9. Many other improvements as to make Coq more modular and better prepared for upcoming incremental and multi-threaded type-checking were also made.

Hugo Herbelin, Emilio J. Gallego Arias and Théo Zimmermann, helped by members from Gallinette (Nantes) and Stamp (ex-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, amounting to hundreths of proposals in the form of pull requests. Moreover, we organized a beginner-focused community Hackathon in early 2022, including a diversity session, with peak attendance of over 100 contributors. Similar community events are planned later on.

Emilio J. Gallego Arias and Shachar Itzaky continued the development of the education-targeted tool jsCoq, which saw in 2021 5 new releases bringing many new features and refinements, and in particular a new backend that has made us declare the tool "production ready" for the first time.

Emilio J. Gallego Arias also maintained the coq-serapi tool, used in a few labs as the standard communication API with Coq to perform experiments (including machine learning ones). In collaboration with Thierry Martinez, Gallego Arias also released a pyCoq package which is specifically targeted at learning and software engineering researchers using Coq for their experiments.

8.5 Formalisation and verification

Participants: Emilio Jesús Gallego Arias, Pierre Letouzey, Jean-Jacques Lévy, Daniel de Rauglaudre, Yann Régis-Gianas, Alexis Saurin.

8.5.1 Lexing and regular expressions in Coq

Pierre Letouzey continued working on a Coq formalisation started with Yann Régis-Gianas, on regular expressions (with complement and conjunction) and their Brzozowski derivatives. Many techniques have been attempted to prove correct the exact details used in a real-world implementation (ml-ulex), but a complete proof of this implementation is still elusive.

8.5.2 Hofstadter nested recursive functions and Coq

Pierre Letouzey continued this year the study of a family of nested recursive functions proposed by D. Hofstadter in his book "Gödel Escher Bach". Some earlier conjectures have been proved. In particular, the appearance of a Rauzy fractal during this work is now better understood. The formalization of these proofs are pending, requiring quite some matrix theory and complex polynomials. Another important conjecture states that this family of nested functions is increasing. Despite some progress, this conjecture still lacks a complete proof. More details on this site.

8.5.3 Sensitivity Conjecture in Coq

Daniel de Rauglaudre pursued his formalization in Coq of the Sensitivity Conjecture (which became a Theorem in 2019 thanks to Hao Huang [74]). The sensitivity conjecture remained an open-problem for more than thirty years, aiming to relate the sensitivity of a Boolean function results to its input values to other complexity measures of Boolean functions, such as block sensitivity. De Rauglaudre started to formalize Huang's very succinct proof of the conjecture.

For proving some lemmas in this theorem, numerous formalizations in Linear Algebra (matrices, determinants, eigenvalues, permutations, sorting etc.) have been implemented. In this context, a study of algebra of ring-like structures has been started, and some syntax of iterators have been studied and added. This development is available a here.

8.5.4 Proofs of algorithms on graphs

Jean-Jacques Lévy pursues his work about formal proofs of graph algorithms. The goal is to provide computer-checked proofs of algorithms that remain human readable. At ITP 2019 [50], they presented an

article with Chen Ran, Cyril Cohen, Stephan Merz and Laurent Théry on three different ways of proving such an algorithm in Why3, Coq and Isabelle/HOL. By publishing the entire proofs, they encouraged the community to compare our proofs with the ones possible in other machine-checked proof systems.

Jean-Jacques Lévy now remodels his proof with new versions of Why3 and also plan to compare the existing Coq proof using Mathcomp/ssreflect with a proof using Coq classics. He still works on a proof of implementation of Tarjan SCC algorithm with imperative programming and memory pointers.

8.5.5 Iterated parametricity and semi-cubical sets

Hugo Herbelin and Ramkumar Ramachandra carried on their formalization in Coq of an original dependently-typed construction of semi-cubical sets inspired by the parametricity translation. This continued to highly stressed the limits of Coq, especially in terms of second-order unification, higher-order rewriting, efficiency.

8.5.6 Verified Datalog programs with applications to low-level binary analysis

Emilio J. Gallego Arias continued collaboration with Stefania Dumbrava and Cody Roux on the use of our verified Datalog engine [47] for the analysis of low-level binary code. In particular, using metacoq we have developed a method to translate datalog programs to Coq proof friendly specifications while preserving the semantic correspondence with the verified engine. This allows us to specify analysis as efficient datalog programs, but to prove properties about them using a more convenient native to Coq representation.

8.5.7 Infinitary Proofs and Parity Automaton

Esaïe Bauer, Emilio J. Gallego Arias and Alexis Saurin have started a Coq formalization of infinitary proofs and their validity checking using Parity Automaton. They have started from the proof methodology developed for the the math-comp library, but this particular topic poses many interesting challenges from the point of view of proof engineering, in particular related to the formalization of infinite graphs and automaton in a *natural* way.

8.5.8 Real-time Digital Signal Processing

Emilio J. Gallego Arias and Pierre Jouvelot presented their work on a formalized synchronous language for linear DSP processors at the FARM 2021 conference (part of ICFP), which was held virtually. The development produced a paper and uses techniques from the programming language literature such as logical relations to prove that every well typed program is linear (in the linear algebra sense). This opens up the door to many other interesting developments which are being discussed now.

8.5.9 Mechanism design

Emilio J. Gallego Arias collaborated with Pierre Jouvelot on the formalized verification of the general Vickrey-Clarke-Groves mechanism (see for example [83]) using Coq, designing a Coq-based framework for the specification and refinement of mechanisms, covering classical examples from the literature. This has resulted in a draft [35] already presented as a poster at EC'22 [38].

Participants: Emilio Gallego, Hugo Herbelin, Paul-André Melliès, Alexis Saurin, Théo Zimmermann.

9 Bilateral contracts and grants with industry

9.1 Bilateral contracts with industry

The team has an ongoing industrial contract started with Nomadics Lab aiming at improving the development of Coq (continuous integration, merging of pull requests, bug tracking, improving the release

process, ...) and of its package ecosystem (for instance building documented best practices, tools and easy installers for newcomers). Theo Zimmermann decided to quit in the end of 2022 his 3-year engineer position (started in January 2020 funded by this contract) after getting a maître de conférences position in Telecom Paris Tech. His role has been to pursue 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. The team plans to continue having scientific collaboration with Zimmermann on this topic.

10 Partnerships and cooperations

Participants: Kostia Chardonnet, Pierre-Louis Curien, Abhishek De, Alen Duric, Thomas Ehrhard, Emilio Jesus Gallego Arias, Hugo Herbelin, Farzad Jafar-Rahmani, Pierre Letouzey, Paul-Andre Mellies, Alexis Saurin.

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

Other international visits to the team

Bahareh Afshari

Status Professor

Institution of origin: Götenburg University

Country: Sweden

Dates: 1 to 3 December 2022

Context of the visit: participated to Abhishek De's PhD defense and participated to RECIPROG workshop where she gave a talk on interpolation in the modal μ-calculus.

Mobility program/type of mobility: research stay.

Shachar Itzhaky

Status Senior Lecturer

Institution of origin: Technion University

Country: Israël

Dates: 18 to 25 September 2022

Context of the visit: gave a research seminar on Monday 19 September and worked with members of the Picube team during his stay in Paris.

Mobility program/type of mobility: research stay.

Patricia Johann

Status Full Professor

Institution of origin: Appalachian State University, North Carolina

Country: United States

Dates: 18 to 25 October 2022

Context of the visit: She worked with members of the Picube team during her stay and also participated as an examiner in Hugo Moeneclaey's thesis defence.

Mobility program/type of mobility: research stay.

Ambrus Kaposi

Status Associate Professor

Institution of origin: Eötvös Loránd University in Budapest

Country: Hungary

Dates: 18 to 20 October 2022

Context of the visit: He gave a research seminar on Wednesday 19 October and worked with members of the Picube team during his stay in Paris. He also participated as an examiner in Hugo Moeneclaey's thesis defence.

Mobility program/type of mobility: research stay.

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.

Alexis Saurin is member of the S^3 ANR project coordinated by Christine Tasson (Sorbonne Université).

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 (aka. 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 will contribute 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 $\pi.r^2$ is located, LIP in Lyon, LIS in Marseille and LS2N in Nantes. More informations on the project can be found at this website.

Two project workshops were organized in 2022: in Lyon in May 2022 and in Paris in December 2022.

In collaboration with Riccardo Brasca 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, Pierre Letouzey, Paul-André Melliès and Alexis Saurin submitted an Emergence Recherche project to 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 coq communities.

11 Dissemination

Participants: Antoine Allioux, Esaïe Bauer, Vincent Blazy, Kostia Chardonnet, El Mehdi Cherradi, Pierre-Louis Curien, Alen Duric, Thomas Ehrhard, Emilio Jesus Gallego Arias, Hugo Herbelin, Farzad Jafar-Rahmani, Pierre Letouzey, Giulia Manara, Paul-Andre Mellies, Vincent Moreau, Sarah Reboullet, Alexis Saurin, Clément Théron.

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 2023 (Journées nationales du GDR-IM) early april 2023.

11.1.2 Journal

Member of the editorial boards Paul-André Melliès is a member of the editorial board of the journal Theoretical Computer Science.

11.1.3 Invited talks

Paul-André Melliès and Alexis Saurin gave an invited lecture in the linear logic winter school held at CIRM from 24 to 28th january 2022 during the Logic and interaction weeks (CIRM 2022).

Alexis Saurin was an invited speaker at the Logic in Computer Science special session of the Logic Colloquium in Reykjavick (LC 2022), 27th june to 1st july 2022.

11.1.4 Leadership within the scientific community

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

11.1.5 Research administration

Alexis Saurin is an elected member in conseil académique de la faculté des sciences de l'Université Paris Cité and of the commission recherche.

Paul-André Melliès is a member of the conseil de laboratoire de l'Institut de Recherche en Informatique Fondamentale (IRIF).

11.2 Teaching - Supervision - Juries

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.

Master supervision

- Pierre-Louis Curien, jointly with François Métayer, supervised the M2 internship (Université de Paris, parcours Mathématiques Fondamentales) of Aloÿs Dufour, on various notions of contractibility for simplicial sets. Aloÿs Dufour is now a doctoral student under the supervision of Damiano Mazza at Université Sorpbonne Paris Nord.
- Pierre-Louis Curien, jointly with Samuel Mimram, supervised the M2 internship (Sorbonne Université, parcours Mathématiques Fondamentales) of Naomi Jacquet on model structures for presentations and Lawvere theories. Unfortunately, Naomi experienced serious health problems and regretfully had to abandon.

11.2.2 Juries

Alexis Saurin was member of comité de sélection MCF for UFR de mathématiques in Université Paris Cité for hiring a MCF in mathematics for research integration to IRIF.

11.3 Popularization

11.3.1 Education

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).

12 Scientific production

12.1 Major publications

[1] D. Baelde, A. Doumane and A. Saurin. 'Infinitary proof theory : the multiplicative additive case'. In: *Proceedings of CSL 2016*. Sept. 2016. URL: https://hal.archives-ouvertes.fr/hal-01339037.

[2] P.-L. Curien. 'Operads, clones, and distributive laws'. In: *Operads and Universal Algebra : Proceedings of China-France Summer Conference*. Ed. by C. Bai, L. Guo and J.-L. Loday. Nankai Series in Pure, Applied Mathematics and Theoretical Physics, Vol. 9. Tianjin, China: World Scientific, July 2010, pp. 25–50. URL: https://hal.archives-ouvertes.fr/hal-00697065.

[3] P.-L. Curien, R. Garner and M. Hofmann. 'Revisiting the categorical interpretation of dependent type theory'. In: *Theoretical computer Science* 546 (2014), pp. 99–119. URL: http://dx.doi.org/10.1007/s10990-007-9006-0.

[4] P.-L. Curien and 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. URL: https://hal.archives-ouvertes.fr/hal-00155295.

[5] P.-L. Curien and H. Herbelin. 'The duality of computation'. In: *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00)*. SIGPLAN Notices 35(9). Montreal, Canada: ACM, Sept. 2000, pp. 233–243. DOI: \url{http://doi.acm.org/10.1145/351240.351262}. URL: http://hal.archives-ouvertes.fr/inria-00156377/en/.

[6] P. Dehornoy and Y. Guiraud. 'Quadratic normalization in monoids'. In: *Internat. J. Algebra Comput.* 26.5 (2016), pp. 935–972. URL: https://doi.org/10.1142/S0218196716500399.

[7] E. J. Gallego Arias, B. Pin and 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*. Ed. by S. Autexier and P. Quaresma. Vol. 239. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2017, pp. 15–27. URL: http://dx.doi.org/10.4204/EPTCS.239.2.

[8] S. Gaussent, Y. Guiraud and P. Malbos. 'Coherent presentations of Artin monoids'. In: *Compositio Mathematica* 151.5 (2015), pp. 957–998. DOI: 10.1112/S0010437X14007842. URL: https://hal.archives-ouvertes.fr/hal-00682233.

[9] G. Gilbert, J. Cockx, M. Sozeau and N. Tabareau. 'Definitional Proof-Irrelevance without K'. In: *46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2019)*. POPL. Lisbon, Portugal, Jan. 2019. URL: https://hal.inria.fr/hal-01859964.

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

[11] Y. Guiraud, E. Hoffbeck and P. Malbos. 'Convergent presentations and polygraphic resolutions of associative algebras'. In: *Mathematische Zeitschrift* 293.1-2 (2019), pp. 113–179. DOI: 10.1007/s00209-018-2185-z. URL: https://hal.archives-ouvertes.fr/hal-01006220.

[12] Y. Guiraud and P. Malbos. 'Higher-dimensional normalisation strategies for acyclicity'. In: *Advances in Mathematics* 231.3-4 (2012), pp. 2294–2351. DOI: 10.1016/j.aim.2012.05.010. URL: https://hal.archives-ouvertes.fr/hal-00531242.

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

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

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

[16] H. Herbelin. 'On the Degeneracy of Sigma-Types in Presence of Computational Classical Logic'. In: *Proceedings of TLCA 2005*. Ed. by P. Urzyczyn. Vol. 3461. Lecture Notes in Computer Science. Springer, 2005, pp. 209–220.

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

[18] P. Letouzey. *Hofstadter's problem for curious readers*. Research Report. Université Paris Diderot ; INRIA Paris-Rocquencourt, Sept. 2015, p. 29. URL: https://hal.inria.fr/hal-01195587.

[19] 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, pp. xiv+589. URL: http://homotopytypetheory.org/book.

[20] Y. Régis-Gianas and 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)*. Vol. 5133. Lecture Notes in Computer Science. Springer, July 2008, pp. 305–335. URL: http://gallium.inria.fr/%5Ctextasciitilde%20fpottier/publis/regis-gianas-pottier-hoarefp.ps.gz.

[21] B. Ziliani and M. Sozeau. 'A comprehensible guide to a new unifier for CIC including universe polymorphism and overloading'. In: *Journal of Functional Programming* 27 (2017). DOI: 10.1017/S0956796817000028. URL: https://hal.inria.fr/hal-01671925.

12.2 Publications of the year

International journals

[22] P.-L. Curien, A. Đurić and Y. Guiraud. 'Coherent presentations of monoids with a right-noetherian Garside family'. In: *Journal of Homotopy and Related Structures* 18 (2023), pp. 115–152. DOI: 10.1007/s40062-023-00323-4. URL: https://hal.science/hal-03276119.

[23] A. Oliveira Vale, P.-A. Melliès, Z. Shao, J. Koenig and L. Stefanesco. 'Layered and Object-Based Game Semantics *'. In: *Proceedings of the ACM on Programming Languages* (16th Jan. 2022). DOI: 10.1145/3498703. URL: https://hal.inria.fr/hal-03456034.

[24] T. Zimmermann, J. Coolen, J. Gross, P.-M. Pédrot and G. Gilbert. 'The Advantages of Maintaining a Multitask, Project-Specific Bot: An Experience Report'. In: *IEEE Software* (3rd June 2022), pp. 2–7. DOI: 10.1109/MS.2022.3179773. URL: https://inria.hal.science/hal-03479327.

International peer-reviewed conferences

[25] D. Baelde, A. Doumane, D. Kuperberg and A. Saurin. 'Bouncing threads for circular and non-wellfounded proofs – Towards compositionality with circular proofs (Extended version)'. In: LICS '22: Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science. LICS '22: Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science. Haifa, Israel: Association for Computing Machinery, 2022. DOI: 10.1145/3531130.3533375. URL: https://hal.science/hal-03682126.

[26] 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.

[27] A. De, F. Jafarrahmani and A. Saurin. 'Phase Semantics for Linear Logic with Least and Greatest Fixed Points'. In: FSTTCS 2022 - 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science. Vol. 250. Chennai, India, 14th Dec. 2022, 35:1–35:23. DOI: 10.4230/LIPIcs.FSTTCS.2022.35. URL: https://hal.science/hal-04061582.

[28] J. Gross, T. Zimmermann, M. Poddar-Agrawal and A. Chlipala. 'Automatic Test-Case Reduction in Proof Assistants: A Case Study in Coq'. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. 13th International Conference on Interactive Theorem Proving (ITP 2022). Vol. 237. 13th International Conference on Interactive Theorem Proving (ITP 2022). Haifa, Israel, 7th Aug. 2022, 18:1–18:18. DOI: 10.4230/LIPIcs.ITP.2022.18. URL: https://hal.inria.fr/hal-03586813.

[29] P.-A. Melliès. 'A Functorial Excursion Between Algebraic Geometry and Linear Logic'. In: *LICS 2022 : Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS 2022 - 37th Annual ACM/IEEE Symposium on Logic in Computer Science. Haifa, Israel, 2nd Aug. 2022. DOI: 10.1145/3531130.3532488. URL: https://hal.science/hal-03874304.

Conferences without proceedings

[30] P. Castéran, J. Damour, K. Palmskog, C. Pit-Claudel and T. Zimmermann. 'Hydras & Co.: Formalized mathematics in Coq for inspiration and entertainment'. In: Journées Francophones des Langages Applicatifs: JFLA 2022. St-Médard d'Excideuil, France, Feb. 2022. URL: https://hal.science/hal-03404668.

[31] P.-A. Melliès and N. Zeilberger. 'Parsing as a lifting problem and the Chomsky-Schützenberger representation theorem'. In: MFPS 2022 - 38th conference on Mathematical Foundations for Programming Semantics. Ithaca, NY, United States, 11th July 2022. URL: https://hal.science/hal-03702762.

[32] K. Palmskog, E. Tassi and T. Zimmermann. 'Reliably Reproducing Machine-Checked Proofs with the Coq Platform'. In: RRRR 2022 - Workshop on Reproducibility and Replication of Research Results. Munich, Germany, 2nd Apr. 2022. URL: https://hal.inria.fr/hal-03592675.

Reports & preprints

[33] K. Chardonnet, A. Saurin and B. Valiron. *A Curry-Howard Correspondence for Linear, Reversible Computation*. 8th Aug. 2022. URL: https://hal.science/hal-03747425.

[34] K. Chardonnet, M. de Visme, B. Valiron and R. Vilmart. *The Many-Worlds Calculus*. 3rd Aug. 2022. DOI: 10.48550/arXiv.2206.10234. URL: https://hal.science/hal-03654190.

[35] P. Jouvelot and E. J. Gallego Arias. *A Foundational Framework for the Specification and Verification of Mechanism Design*. E-458.PDF. MINES ParisTech - PSL Research University, 12th May 2022. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-03666871.

[36] J.-J. Levy. *Tracking Redexes in the Lambda Calculus*. 10th Feb. 2022. URL: https://hal.inria.fr/hal-03564707.

Other scientific publications

[37] A. de Almeida Borges, J.-R. Falleri, J. Fehrle, E. J. Gallego Arias, É. Martin-Dorel, K. Palmskog, A. Serebrenik and T. Zimmermann. *Coq Community Survey 2022: Summary of Results*. Haifa, Israel, 12th Aug. 2022. URL: https://inria.hal.science/hal-03914602.

[38] P. Jouvelot and E. J. Gallego Arias. 'A Foundational Framework for the Specification and Verification of Mechanism Design'. In: EC'22 - Twenty-Third ACM Conference on Economics and Computation. Colorado, United States, 11th July 2022. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-03715847.

12.3 Cited publications

[39] T. Altenkirch and J. Grattage. 'A functional quantum programming language'. In: *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*. 2005, pp. 249–258. DOI: 10.1109/LICS.2005.1.

[40] D. J. Anick. 'On the Homology of Associative Algebras'. In: *Trans. Amer. Math. Soc.* 296.2 (1986), pp. 641–659.

[41] D. Ara and F. Métayer. 'The Brown-Golasiński Model Structure on strict ∞-groupoids revisited'. In: *Homology, Homotopy and Applications* 13.1 (2011), pp. 121–142.

[42] D. Baelde, A. Doumane and A. Saurin. 'Infinitary Proof Theory: the Multiplicative Additive Case'. In: *CSL*. Vol. 62. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 42:1–42:17.

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

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

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

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

[47] A. Bonifati, S. Dumbrava and E. J. Gallego Arias. 'Certified Graph Maintenance with Regular Datalog'. In: *Theory and Practice of Logic Programming* (2018).

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

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

[50] R. Chen, C. Cohen, J.-J. Levy, S. Merz and 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*. Ed. by J. Harrison, J. O'Leary and A. Tolmach. Vol. 141. Portland, United States: Schloss Dagstuhl–Leibniz-Zentrum f̈ur Informatik, Sept. 2019, 13:1–13:19. DOI: 10.4230/LIPIcs.ITP.2019.13. URL: https://hal.inria.fr/hal-02303987.

[51] 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.

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

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

[54] 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.

[55] 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.

[56] H. B. Curry, R. Feys and W. Craig. *Combinatory Logic*. Vol. 1. §9E. North-Holland, 1958.

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

[58] T. Ehrhard. 'Coherent differentiation'. In: *CoRR* abs/2107.05261 (2021). arXiv: 2107.05261. URL: https://arxiv.org/abs/2107.05261.

[59] M. Felleisen, D. P. Friedman, E. Kohlbecker and B. F. Duba. 'Reasoning with continuations'. In: *First Symposium on Logic and Computer Science*. 1986, pp. 131–141.

[60] A. Filinski. 'Representing Monads'. In: *Conf. Record 21st ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, POPL'94*. Portland, OR, USA: ACM Press, Jan. 1994, pp. 446–457.

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

[62] J.-Y. Girard. 'Une extension de l'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types'. In: *Second Scandinavian Logic Symposium*. Ed. by J. Fenstad. Studies in Logic and the Foundations of Mathematics 63. North Holland, 1971, pp. 63–92.

[63] T. G. Griffin. 'The Formulae-as-Types Notion of Control'. In: *Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL '90*. San Francisco, CA, USA, 17-19 Jan 1990: ACM Press, 1990, pp. 47–57.

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

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

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

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

[68] Y. Guiraud and P. Malbos. 'Identities among relations for higher-dimensional rewriting systems'. In: *Séminaires et Congrès, Société Mathématique de France* 26 (2011), pp. 145–161.

[69] Y. Guiraud, E. Hoffbeck and P. Malbos. 'Confluence of linear rewriting and homology of algebras'. In: *3rd International Workshop on Confluence*. Vienna, Austria, July 2014. URL: https://hal.archives-ouvertes.fr/hal-01105087.

[70] Y. Guiraud and P. Malbos. 'Coherence in monoidal track categories'. In: *Math. Structures Comput. Sci.* 22.6 (2012), pp. 931–969.

[71] Y. Guiraud and P. Malbos. 'Higher-dimensional categories with finite derivation type'. In: *Theory Appl. Categ.* 22.18 (2009), pp. 420–478.

[72] 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.

[73] 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.

[74] H. Huang. 'Induced subgraphs of hypercubes and a proof of the Sensitivity Conjecture'. In: *Annals of Mathematics* 190.3 (2019), pp. 949–955. URL: https://www.jstor.org/stable/10.4007/annals.2019.190.3.6.

[75] J.-L. Krivine. 'A call-by-name lambda-calculus machine'. In: *Higher Order and Symbolic Computation* (2005).

[76] J.-L. Krivine. 'Un interpréteur du lambda-calcul'. Unpublished. 1986.

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

[78] Y. Lafont, F. Métayer and K. Worytkiewicz. 'A Folk Model Structure on Omega-Cat'. In: *Advances in Mathematics* 224.3 (2010), pp. 1183–1231.

[79] P. Landin. *A generalisation of jumps and labels*. Tech. rep. ECS-LFCS-88-66. Reprinted in `Higher Order and Symbolic Computation`, 11(2), 1998. UNIVAC Systems Programming Research, Aug. 1965.

[80] P. Landin. 'The mechanical evaluation of expressions'. In: *The Computer Journal* 6.4 (Jan. 1964), pp. 308–320.

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

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

[83] N. Nisan, T. Roughgarden, É. Tardos and V. V. Vazirani. *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[84] M. Parigot. 'Free Deduction: An Analysis of "Computations" in Classical Logic'. In: *Logic Programming, Second Russian Conference on Logic Programming*. Ed. by A. Voronkov. Vol. 592. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer, Sept. 1991, pp. 361–380. URL: http://www.informatik.uni-trier.de/~ley/pers/hd/p/Parigot:Michel.html.

[85] J. C. Reynolds. 'Definitional interpreters for higher-order programming languages'. In: *ACM '72: Proceedings of the ACM annual conference*. Boston, Massachusetts, United States: ACM Press, 1972, pp. 717–740.

[86] 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.

[87] C. C. Squier. 'Word problems and a homological finiteness condition for monoids'. In: *J. Pure Appl. Algebra* 49.1-2 (1987), pp. 201–217.

[88] C. Squier, F. Otto and Y. Kobayashi. 'A finiteness condition for rewriting systems'. In: *Theoret. Comput. Sci.* 131.2 (1994), pp. 271–294.

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

[90] 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.

[91] 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.