RESEARCH CENTRE

Paris

2021
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

Contents

Team PI.R2 1

1 Team members, visitors, external collaborators 2

2 Overall objectives 4

3 Research program 4

3.1 Proof theory and the Curry-Howard correspondence 4
3.1.1 Proofs as programs 4
3.1.2 Towards the calculus of constructions 4
3.1.3 The Calculus of Inductive Constructions 5
3.2 The development of Coq 5
3.2.1 The underlying logic and the verification kernel 6
3.2.2 Programming and specification languages 6
3.2.3 Standard library 6
3.2.4 Tactics 7
3.2.5 Extraction 7
3.2.6 Documentation 7
3.2.7 Proof development infrastructure 7
3.3 Dependently typed programming languages 7
3.3.1 Type-checking and proof automation 8
3.4 Around and beyond the Curry-Howard correspondence 8
3.4.1 Control operators and classical logic 8
3.4.2 Sequent calculus 8
3.4.3 Abstract machines 9
3.4.4 Delimited control 9
3.5 Effective higher-dimensional algebra 9
3.5.1 Higher-dimensional algebra 9
3.5.2 Higher-dimensional rewriting 9
3.5.3 Squier theory 9

4 Application domains 10

5 Social and environmental responsibility 10

5.1 Footprint of research activities 10

6 Highlights of the year 10

6.1 Awards 11
6.2 Towards PiCube 11

7 New software and platforms 11

7.1 New software 11
7.1.1 Coq 11
7.1.2 Rewr 12
7.1.3 Catex 13
7.1.4 Cox 13
7.1.5 coqbot 13
7.1.6 jsCoq 14
7.1.7 coq-serapi 14
7.1.8 pyCoq 15

8 New results **15**
8.1 Effects in proof theory and programming . . . 15
8.1.1 Computational contents of the axiom of choice . . . 15
8.1.2 Proof-search in proof nets . . . 15
8.1.3 Algebraic Logic Programming . . . 15
8.2 Reasoning and programming with infinite data . . . 15
8.2.1 Proof theory of non-wellfounded and circular proofs . . . 16
8.2.2 On the semantics of finitary and non-wellfounded proofs . . . 16
8.2.3 Quantum programming languages with inductive and coinductive types . . . 17
8.3 Effective higher-dimensional algebra . . . 17
8.3.1 Coherent presentations of monoids . . . 17
8.3.2 Polygraphs and opetopes . . . 18
8.3.3 Foundations and formalisation of higher algebra . . . 18
8.4 Metatheory and development of Coq . . . 18
8.4.1 Dependent pattern-matching . . . 18
8.4.2 Software engineering aspects of the development of Coq . . . 18
8.4.3 Software infrastructure and Tools . . . 19
8.5 Formalisation and verification . . . 19
8.5.1 Lexing and regular expressions in Coq . . . 19
8.5.2 Hofstadter nested recursive functions and Coq . . . 19
8.5.3 Sensitivity Conjecture in Coq . . . 20
8.5.4 Proofs of algorithms on graphs . . . 20
8.5.5 Iterated parametricity and semi-cubical sets . . . 20
8.5.6 Verified Datalog programs with applications to low-level binary analysis . . . 20
8.5.7 Infinitary Proofs and Parity Automaton . . . 20
8.5.8 Real-time Digital Signal Processing . . . 20
8.5.9 Mechanism design . . . 21

9 Bilateral contracts and grants with industry **21**
9.1 Bilateral contracts with industry . . . 21

10 Partnerships and cooperations **21**
10.1 International initiatives . . . 21
10.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program . . . 21
10.2 International research visitors . . . 21
10.2.1 Visits of international scientists . . . 21
10.3 National initiatives . . . 22
10.4 Regional initiatives . . . 22

11 Dissemination **22**
11.1 Promoting scientific activities . . . 22
11.1.1 Scientific events: organisation . . . 22
11.1.2 Scientific events: selection . . . 22
11.1.3 Journal . . . 23
11.1.4 Invited talks . . . 23
11.1.5 Research administration . . . 23
11.2 Teaching - Supervision - Juries . . . 23
11.2.1 Courses . . . 23
11.2.2 Supervision . . . 23
11.3 Popularization . . . 25
11.3.1 Internal or external Inria responsibilities . . . 25
11.3.2 Articles and contents . . . 25

12 Scientific production **25**
12.1 Major publications . . . 25
12.2 Publications of the year . . . 26
12.3 Other . . . 27
12.4 Cited publications . . . 27

Team PI.R2

Creation of the Team: 2021 January 01

Keywords

Computer sciences and digital sciences

A2.1.1. – Semantics of programming languages

A2.1.4. – Functional programming

A2.1.11. – Proof languages

A2.4.3. – Proofs

A7.2. – Logic in Computer Science

A7.2.3. – Interactive Theorem Proving

A7.2.4. – Mechanized Formalization of Mathematics

A8.1. – Discrete mathematics, combinatorics

A8.4. – Computer Algebra

Other research topics and application domains

B6.1. – Software industry

1 Team members, visitors, external collaborators

Research Scientists

- Alexis Saurin [Team leader, CNRS, 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 [Inria, Emeritus, HDR]
- Paul-André Melliès [CNRS, Senior Researcher]

Faculty Members

- Pierre Letouzey [Université de Paris, Associate Professor]
- Daniela Petrisan [Université de Paris, from Feb 2021]

PhD Students

- Antoine Allioux [Inria, Jan 2021]
- Esaie Bauer [Université de Paris]
- Vincent Blazy [Université de Paris]
- Felix Castro [Université de Paris]
- Kostia Chardonnet [Université Paris-Saclay]
- El Mehdi Cherradi [Conseil général de l'économie, de l'industrie, de l'énergie et des technologies]
- Abhishek De [Université de Paris]
- Alen Duric [Université de Paris]
- Colin Gonzalez [Nomadic Labs, CIFRE]
- Farzad Jafarrahmani [Université de Paris]
- Hugo Moeneclaey [Université Paris-Saclay]

Technical Staff

- Thierry Martinez [Inria, Engineer]
- Théo Zimmermann [Inria, Engineer]
- Daniel de Rauglaudre [Inria, Engineer]

Interns and Apprentices

- Jeremy Damour [Inria, Jun 2021]
- Naomi Jacquet [Inria, from Feb 2021 until Jun 2021]
- Amel Kebbouche [Université de Paris]

Administrative Assistants

- Christelle Guiziou [Inria]
- Anne Mathurin [Inria]
- Scheherazade Rouag [Inria, from Oct 2021]

External Collaborator

- Yann Régis-Gianas [Université de Paris, Dec 2021]

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 [63] 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 [54], then by Howard and de Bruijn at the end of the 60’s [75, 44], 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 [51, 52] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [93].

3.1.2 Towards the calculus of constructions

The λ-calculus, defined by Church [50], 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 [40].

To explain the Curry-Howard correspondence, it is important to distinguish between intuitionistic and classical logic: following Brouwer at the beginning of the 20th 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 [85].

In 1985, Coquand and Huet [51, 52] in the Formel team of INRIA-Rocquencourt explored an alternative approach based on Girard-Reynolds’ system F [64, 89]. 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 [53] 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 [94] 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 [41], the more advanced Certified Programming with Dependent Types book by Chlipala [49] 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 [65] 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 [81] and Reynolds [88] and started to be studied in an abstract way in the 80's by Felleisen *et al* [61], leading to Parigot's $\lambda\mu$-calculus [87], 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 [82] and Krivine's abstract machine for call-by-name evaluation [78, 77]. 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 [62]: 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 [39]) 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 [56]).

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 [74], 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 [92], and independently by Burroni [45], 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 [79, 84, 66, 67], and beyond: Petri nets [69] and formal proofs of classical and linear logic have been expressed in this framework [68]. 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 [42].

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 [36] and Squier [90, 91]: 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], [73, 70, 72].

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 [80, 37]). 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 [71][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

The team published five papers in the LICS 2021 conference, authored by Antoine Allioux (with Eric Finster and Mathieu Sozeau), Thomas Ehrhard and Farzad Jafarrahmani, Hugo Herbelin (with Nuria Brede), Paul-André Melliès and Hugo Moeneclaey ; and a POPL 2022 paper authored by Paul-André Melliès (with Arthur Vale, Zhong Shao, Jérémie Koenig and Léo Stefanesco).

6.1 Awards

Coq was awarded early 2022 the open science free software award in the Scientific and Technical category (see website.

6.2 Towards PiCube

During 2021, one of the main creative aspects of the team's work has been to set up the new team and to design the new scientific proposal for the follow-up team that will be created after the end of $\pi.r^2$. The purpose of the team is to take advantage of the most recent advances of

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

in order to reduce the "technological gap" which currently separates the vernacular language used by the working mathematicians in their daily practice and the formal language used today in a proof assistant such as Coq, Agda or Lean.

By building on these converging lines and combining them with an active involvement to the Coq ecosystem at all levels, and a firm commitment towards the formalisation of mathematics, the ambition of the Picube project is to provide the foundations for a new generation of proof assistants

- based on a structured notion of mathematical document, which reflects and guides the creative process of each working mathematician in a personalised way, and integrates the dynamic and incremental nature of mathematics,
- allowing transport of proofs and concepts across libraries possibly relying on different definitions of basic notions, such as the rationals or reals,
- better adapted to information retrieval techniques and to statistical methods as well as to probabilistic algorithms coming from machine learning.

The Picube team is organised in five research axis:

1. Fundamental Structures of Logic and of 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.

and includes three new members: Thomas Ehrhard (DR CNRS), Paul-André Melliès (DR CNRS and future team manager) and Daniela Petrişan (MdC UPC).

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 8.14 integrates many usability improvements, as well as an important change in the core language. The main changes include:

- The internal representation of match has changed to a more space-efficient and cleaner structure, allowing the fix of a completeness issue with cumulative inductive types in the type-checker. The internal representation is now closer to the user-level view of match, where the argument context of branches and the inductive binders "in" and "as" do not carry type annotations.

- A new "coqnative" binary performs separate native compilation of libraries, starting from a .vo file. It is supported by coq_makefile.

- Improvements to typeclasses and canonical structure resolution, allowing more terms to be considered as classes or keys.

- More control over notation declarations and support for primitive types in string and number notations.

- Removal of deprecated tactics, notably omega, which has been replaced by a greatly improved lia, along with many bug fixes.

- New Ltac2 APIs for interaction with Ltac1, manipulation of inductive types and printing.

Many changes and additions to the standard library in the numbers, vectors and lists libraries. A new signed primitive integers library Sint63 is available in addition to the unsigned Uint63 library.

News of the Year: Coq version 8.14 integrates many usability improvements, as well as an important change in the core language. See the changelog at https://coq.inria.fr/refman/changes.html#version-8-14 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

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, Yann Régis-Gianas, Alexis Saurin.

8.1.1 Computational contents of the axiom of choice

Hugo Herbelin developed in collaboration with Nuria Brede (U. Potsdam) a unified approach of the underlying logical structure of choice and bar induction principles [23]. The work was presented at LICS 2021, TYPES 2021 and at the Proof Society Virtual Seminar.

8.1.2 Proof-search in proof nets

Building on the work initiated in 2020, Alexis Saurin expanded his work on proof-net construction building on an interpretation a general view of sequentialization as a converse operation to proof construction for a class of correctness criteria. This approach contrast to focusing proof-search which reduces the non-determinism of the search for a proof by adding sequentializability constraints in sequent proofs (sometimes refered to as hypersequentialized in this case) preserving the completeness of the resulting proof-search space. After considering paraproofnet-search as a dual operation to the parsing correctness criterion, he is currently investigating how proof-structures contractibility properties, which can also be viewed as a distributed sequentialization of a proof-graph, can lead to proof construction mechanisms.

Late 2021, he started a collaboration with Aurore Alcolei and Luc Pellissier on a related topic, namely an approach to interactive proof construction in game-semantical frameworks (concurrent games, ludics...) to express the search for proofs as specified by a set of counter-strategies presented as proof-nets, or more abstractly desequentialized strategies.

8.1.3 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.

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. 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 [38] 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 will be submitted during the first semester of 2022.

An advantage of this approach is to abstract from the precise choice of a validity condition making the proof quite robust *wrt.* modifications of the validity conditions. For instance it adapts nicely to bouncing validity.

Proof nets for non-wellfounded proofs. Together with Luc Pellissier (LACL, Université Paris Est-Créteil), Abhishek De and Alexis Saurin generalized infinets [55] (proof-nets for non-wellfounded proofs of μMLL, relaxing problematic restrictions of the original presentation, namely the need to consider only finitely many cuts. Considering potentially infinite (non-wellfounded) proof-objects, it is indeed natural to allow for infinitely many cut rules (typically when a cut occurs within a cycle of a regular proof). Their new results, published at PPDP[24], show how to integrate infinitely many cuts, while proposing a new presentation of the productive cut-elimination on μMLL infinets as well as for its sequent calculus. A journal version is being written.

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. A paper has been submitted early 2022.

8.2.2 On the semantics of finitary and non-wellfounded proofs

Denotational semantics On the semantics of finitary and non-wellfounded proofs. In 2021 Thomas Ehrhard, motivated by an earlier wook [58] (long version to appear in the journal LMCS in 2022), has developed 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. 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. An article [57] is currently submitted to a journal.

With his PhD student Farzad Jafarrahmani, Ehrhard has developed a categorical semantics of Linear Logic with least and greatest fixpoints of types, with logical fixpoint rules generalizing those introduced by David Baelde, adapting Park's rules. They have also developed a concrete example of such categories, the non-uniform totality spaces, where least and greatest fixpoints have distinct interpretations. This work has been published [59].

Thomas Ehrhard, Farzad Jafarrahmani and Alexis Saurin extended the previous work to polarized Linear Logic with fixed-points. One of our objectives is to develop Linear Logic foundations to inductive and coinductive types in Coq.

Last, they presented to TLLA 2021 [60] the first results on extending the above interpretation to circular proofs unveiling the denotational counterpart of the validity condition of circular proofs.

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.

With his Master student Naim Favier, Alexis Saurin studied polarization properties of linear logics from the point of view of phase semantics, obtainined a semantic proof of focusing in the case of MALL that he is currently working at lifting to $\mu MALL$.

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 [35] 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. This was presented in TLLA 2021.

In a collaboration with Valiron and Vilmart, Chardonnet investigated an asynchronous model of Geometry of Interaction for the pure ZX-Calculus, a graphical language for quantum computation, and its extension to ground-processes. This GoI semantics takes the form of a Token Machine. They showed how to connect this new semantics to the usual standard interpretation of the ZX-diagrams. This was published in MFCS 2021 [47].

In addition, in a collaboration with Lemonnier and Valiron, he presented a categorical semantics for reversible computation. Focusing on a typed, functional reversible language based on Theseus, they discuss how join inverse rig categories do not in general capture pattern-matching, the core construct used in Theseus to enforce reversibility and then derive a categorical structure to add to join inverse rig categories in order to capture pattern-matching, showing how such a structure makes an adequate model for reversible pattern-matching. This work was published in MFPS 2021 [46].

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 [30].

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).

8.3.3 Foundations and formalisation of higher algebra

Antoine Allioux (PhD started in February 2018), Eric Finster, Yves Guiraud and Matthieu Sozeau continued to explore the development of higher algebra in type theory, based on an extension of type theory with a universe of strict polynomial monads. Their approach and their work on internalising the ∞-groupoids associated to any type was presented at LICS 2021 and at TYPES 2021. They are now concentrating on developing the theory of $(\infty,1)$-categories in this new framework [34].

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

In January 2020, Théo Zimmermann was recruited 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.

From a technical standpoint, in 2021, Théo Zimmermann has collaborated with Cyril Cohen (Inria Stamp) to create the Coq Nix Toolbox, which allows using the Nix package manager to contribute to and maintain Coq projects. In particular, this tool supports generating Continuous Integration (CI) configurations to test a project with its *reverse dependencies* (the projects that depend on it). This work was presented at the Coq Workshop 2021 [32]. He 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, the bot that Théo Zimmermann has created and maintains. This integration has allowed many Coq developers and contributors to benefit from automatic test-case reduction from CI failures. Théo Zimmermann is in the process of writing and submitting a paper on the topic, with Jason Gross and Adam Chlipala. The bot itself has also been the topic of another submission [31], with the various contributors to the bot as co-authors. In particular, the second co-author, Julien Coolen, was Théo Zimmerman's intern during the summer of 2020.

In June 2021, Théo Zimmermann supervised 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 [27].

From an empirical research standpoint, Théo Zimmermann has continued his collaboration with Jean-Rémy Falleri (from LaBRI) on understanding Community Package Maintenance Organizations. They have published a *registered report* about it at ICSME 2021 [29], and are thus expected to submit a full journal version in 2022.

Finally, during the last three months of 2021, Théo Zimmermann has coordinated an *ad hoc* working group to prepare the Coq Community Survey that is being run in the beginning of 2022, with objectives to get an updated picture of the Coq community and to inform future decisions of the Coq development team. Emilio J. Gallego Arias also participated 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 [76]). 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 [48], 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 [43] 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 [86]) 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 conference paper submission early 2022.

9 Bilateral contracts and grants with industry

9.1 Bilateral contracts with industry

An 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 started a three-year research engineer position in January 2020 funded by this contract, continuing 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.

A CIFRE PhD application with the goal of developing an assistant for the verification of software using Coq based on machine-learning techniques has been submitted to the ANRT, in collaboration with the Equisafe.io enterprise. The prospective student, Thomas Binetruy-Pic, is expected to start his PhD first half of 2022.

10 Partnerships and cooperations

10.1 International initiatives

10.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program

VIP (Verification, Interaction and Proofs), from 2017 coordinated by Ying JIANG, involving as partners the Chinese Academy of Sciences.

10.2 International research visitors

10.2.1 Visits of international scientists

International visits to the team.

- Jim Lipton, professor at Wesleyan university visited the team for 2 week in July 2021.
- Niccolo Veltri, researcher at the Department of Software Science of Tallinn University of Technology, visited the team in november 2021.
- Aurore Alcolei, from the University of Bologna, visited the team in december 2021.

Research stays abroad.

- Pierre-Louis Curien has been staying 6 weeks at the Mathematical Research Institute in Oberwolfach (August 2021, and November-December 2021) as a Research Fellow. This stay allowed him to prepare new material for his M2 course on homotopical algebra and higher categories at Université de Paris (parcours LMFI).
- Abhishek De visited Anupam Das in Birmingham.
- Emilio J. Gallego Arias visited Jim Lipton at Wesleyan University in December 2021.

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 Topologie Algébrique**, federating French researchers working on classical topics of algebraic topology and homological algebra, such as homotopy theory, group homology, K-theory, deformation theory, and on more recent interactions of topology with other themes, such as higher categories and theoretical computer science.

Alexis Saurin is member of the S^3 ANR project coordinated by Christine Tasson (Sorbonne Université) as well as the **QuaReMe** coordinated by Matteo Mio (ENS Lyon).

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.

10.4 Regional initiatives

Thomas Ehrhard and Alexis Saurin are members of the Emergence de la Ville de Paris **RealiSe** project, aiming at a convergence between the studies on the semantics of functional programs (and their probabilistic extensions) and of reactive and synchronous programs. The project is led by Christine Tasson, from LIP6, and also involves memebers of PARKA team.

Thomas Erhard, Emilio J. Gallego Arias, and Paul-André Melliès have started a collaboration with the Inria SIERRA team in order to relate programming languages methods with machine learning and optimization techniques.

11 Dissemination

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

- Alexis Saurin co-organized the 2021 annual meeting of the Scalp working group in IUT Fontainebleau-Sénart (3 days).

- Thomas Ehrhard and Pierre-Louis Curien are part of the organizing committee of the Logic and interaction weeks taking place in CIRM early 2022.

11.1.2 Scientific events: selection

- Daniela Petrisan was a member of the program committee of RAMICS 2021 international conference.

- Alexis Saurin was a member of the program committee of the PPDP 2021 international conference.

- Emilio J. Gallego Arias was a member of the program committe of the BCCA 2021 international conference.

11.1.3 Journal

- Pierre-Louis Curien is editor-in-chief of the journal Mathematical Structures in Computer Science.

11.1.4 Invited talks

Hugo Herbelin has given an invited talk at the Proof Theory Virtual Seminar on june 16th 2021, entited "On the logical structure of choice and bar induction principles".

Pierre-Louis Curien has given an invited talk "Coherent presentations of monoids with a Garside family" at the workshop Braids and beyond, in the memory of Patrick Dehornoy (Caen, September 8-10, 2021, website).

Jean-Jacques Lévy was invited to give a talk "Vive la Recherche en Informatique !" at the seminar of LMF (Laboratoire des Méthodes Formelles) in ENS Saclay (October 15).

Jean-Jacques Lévy gave a tutorial on "Finite Developments in the Lambda-Calculus" at ISR 2021 (the 12th International School on Rewriting), Madrid, 5-16 July 2021. [hal-03566512]

Alexis Saurin has given an invited talk at the Proof Theory Virtual Seminar, on november 17th, entitled "Virtuous circles in proofs" .

11.1.5 Research administration

Alexis Saurin was a member of the CRCN and ISFP hiring committee for INRIA Saclay.

Alexis Saurin is a member of the research council for the Faculté des sciences of Université de Paris.

Thomas Ehrhard, Hugo Herbelin, Paul-André Melliès and Alexis Saurin seat and the scientific council of UFR d'informatique, Université de Paris.

11.2 Teaching - Supervision - Juries

11.2.1 Courses

Pierre-Louis Curien has been teaching a course "Homotopical algebra and higher categories" in the Master LMFI of Université de Paris in 2020-2021.

Thomas Ehrhard and Paul-André Melliès taught the course on denotational semantics at Master MPRI of Université de Paris in the first and second semester 2020-2021 and 2021-2022.

Hugo Herbelin and Hugo Moeneclaey taught a class on Homotopy Type Theory in the Master LMFI of Université de Paris in the second semester of 2020-2021.

Pierre Letouzey taught the course on functional programming, proof assistants and M2 LMFI.

Paul-André Melliès taught the course on lambda-calculus and categories at ENS Ulm (M1 MPRI).

Daniela Petrisan taught in the automata course of M2 MPRI.

Alexis Saurin has been teaching a course in proof theory in the Master LMFI of Université de Paris in the first semester of 2021-2022.

11.2.2 Supervision

PhD supervision

- PhD started: 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 started: Giulia Manara, Towards parallel cut elimination in MELL proof structures, started in October 2021, supervised by Thomas Ehrhard.

- PhD started: Vincent Moreau, on the connections between higher-order automata and topological duality, started in September 2021, supervised by Paul-André Melliès.

- PhD started: Sarah Reboullet, Parametricity and Univalence, started in September 2021, supervised by Hugo Herbelin.

- PhD started: 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.
- 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.
- 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.
- 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.
- PhD in progress: Abhishek De, Proof-nets for fixed-point logics and non-well-founded proofs, Université de Paris, since October 2018, supervised by Alexis Saurin.
- PhD in progress: Lucas Massoni Sguerra, Formal verification of mechanism design, Université PSL, since January 2018, supervised by Gérard Memmi, Pierre Jouvelot, and Emilio J. Gallego Arias.
- PhD defended: Rémi Nollet, Validity conditions for circular proofs, Université de Paris, defended in june 2021, supervised by Alexis Saurin and Christine Tasson.
- PhD defended: Axel Osmond, A categorical study of spectral duality, defended in December 2021, supervised by Paul-André Melliès
- PhD defended: Chaitanya Leena Subramaniam, From dependent type theory to higher algebraic structures, defended in September 2021, supervised by Paul-André Melliès.

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.
- Hugo Herbelin supervised Sarah Reboullet's LMFI M2 internship leading to her PhD topic on parametricity and univalence.
- Alexis Saurin supervised Esaie Bauer's LMFI M2 intenrship one the proof theory of LTL and reactive programming. (see above in the PhD supervision section.)

11.3 Popularization

11.3.1 Internal or external Inria responsibilities

Jean-Jacques Lévy is member of the Inria-Alumni's executive committee (6 meetings in 2021). He co-organized and organized 2 sessions about Quantum Computing (June 2) and Natural Language Processing (November 19).

11.3.2 Articles and contents

Pierre-Louis Curien, jointly with Nicolas Curien, has published a recreational mathematics article with title "Quand les nombres content autant qu'ils comptent" in the journal La Recherche (numéro 561, juillet-août 2021).

Jean-Jacques Lévy has sent an article [83] about Tracking Redexes in the Lambda-Calculus for a chapter in the future book FSP92 (the French Science of Programming), edited by Bertrand Meyer (ETHZ).

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: *\rmfamily Proceedings of the 12th Workshop on User Interfaces for Theorem Provers, \rmfamily 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] 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.

International peer-reviewed conferences

[23] N. Brede and H. Herbelin. 'On the logical structure of choice and bar induction principles'. In: LICS 2021 - 36th Annual Symposium on Logic in Computer Science. Rome / Virtual, Italy, 29th June 2021. URL: https://hal.inria.fr/hal-03144849.

[24] A. De, L. Pellissier and A. Saurin. 'Canonical proof-objects for coinductive programming: infinets with infinitely many cuts'. In: PPDP 2021: 23rd International Symposium on Principles and Practice of Declarative Programming. Tallinn, Estonia: ACM, 6th Sept. 2021, pp. 1–15. DOI: 10.1145/3479394.3479402. URL: https://hal.archives-ouvertes.fr/hal-03371935.

[25] E. J. Gallego Arias, P. Jouvelot, S. Ribstein and D. Desblancs. 'The W-calculus: A Synchronous Framework for the Verified Modelling of Digital Signal Processing Algorithms'. In: *FARM 2021: Proceedings of the 9th ACM SIGPLAN International Workshop on Functional Art, Music, Modelling, and Design*. FARM 2021 - 9th ACM SIGPLAN International Workshop on Functional Art, Music, Modelling, and Design. Virtual, South Korea, 22nd Aug. 2021. DOI: 10.1145/3471872.3472970. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-03322174.

[26] P.-A. Melliès. 'Asynchronous Template Games and the Gray Tensor Product of 2-Categories'. In: LICS 2021 - 36th Annual ACM/IEEE Symposium on Logic in Computer Science. Rome, Italy: IEEE, 29th July 2021. DOI: 10.1109/LICS52264.2021.9470758. URL: https://hal.inria.fr/hal-03455968.

Conferences without proceedings

[27] 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.archives-ouvertes.fr/hal-03404668.

[28] L. Massoni Sguerra, P. Jouvelot, E. J. Gallego Arias, G. Memmi and F. Coelho. 'Blockchain Performance Benchmarking: a VCG Auction Smart Contract Use Case for Ethereum and Tezos (Short Paper)'. In: FAB 2021 - Fourth International Symposium on Foundations and Applications of Blockchain. Davis / Virtual, United States, 7th May 2021. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-03210222.

[29] T. Zimmermann and J.-R. Falleri. 'A grounded theory of Community Package Maintenance Organizations-Registered Report'. In: ICSME 2021 - 37th International Conference on Software Maintenance and Evolution. Luxembourg City / Virtual, Luxembourg, 27th Sept. 2021. URL: https://hal.inria.fr/hal-03320556.

Reports & preprints

[30] P.-L. Curien, A. Đurić and Y. Guiraud. *Coherent presentations of a class of monoids admitting a Garside family*. 1st July 2021. URL: https://hal.archives-ouvertes.fr/hal-03276119.

[31] T. Zimmermann, J. Coolen, J. Gross, P.-M. Pédrot and G. Gilbert. *Extending the team with a project-specific bot*. 14th Dec. 2021. URL: https://hal.inria.fr/hal-03479327.

Other scientific publications

[32] C. Cohen and T. Zimmermann. *A Nix toolbox for reproducible Coq environments, Continuous Integration and artifact reuse*. Virtual, France, 2nd July 2021. URL: https://hal.inria.fr/hal-03366644.

12.3 Other

Educational activities

[33] J.-J. Levy. 'Finite Develpments in the Lambda Calculus'. Doctoral. Spain, 6th July 2021. URL: https://hal.inria.fr/hal-03566512.

12.4 Cited publications

[34] A. Allioux, E. Finster and M. Sozeau. 'Types are internal infinity-groupoids'. working paper or preprint. Jan. 2021. URL: https://hal.inria.fr/hal-03133144.

[35] 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.

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

[37] 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.

[38] 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.

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

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

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

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

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

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

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

[46] K. Chardonnet, L. Lemonnier and B. Valiron. 'Categorical Semantics of Reversible Pattern-Matching'. In: *MFPS*. Vol. 351. EPTCS. 2021, pp. 18–33.

[47] K. Chardonnet, B. Valiron and R. Vilmart. 'Geometry of Interaction for ZX-Diagrams'. In: *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*. Ed. by F. Bonchi and S. J. Puglisi. Vol. 202. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 30:1–30:16.

[48] 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 f́ur Informatik, Sept. 2019, 13:1–13:19. DOI: 10.4230/LIPIcs.ITP.2019.13. URL: https://hal.inria.fr/hal-02303987.

[49] 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.

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

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

[52] 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.

[53] 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.

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

[55] A. De and A. Saurin. 'Infinets: The parallel syntax for non-wellfounded proof-theory'. In: *TABLEAUX 2019 - 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. TABLEAUX 2019 - 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. London, United Kingdom, Sept. 2019. URL: https://hal.archives-ouvertes.fr/hal-02337286.

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

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

[58] T. Ehrhard. ‘Differentials and Distances in Probabilistic Coherence Spaces’. In: *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany.* Ed. by H. Geuvers. Vol. 131. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 17:1–17:17. DOI: 10.4230/LIPIcs.FSCD.2019.17. URL: https://doi.org/10.4230/LIPIcs.FSCD.2019.17.

[59] T. Ehrhard and F. Jafarrahmani. ‘Categorical models of Linear Logic with fixed points of formulas’. In: *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS.* IEEE, 2021, pp. 1–13.

[60] T. Ehrhard, F. Jafarrahmani and A. Saurin. ‘On relation between totality semantic and syntactic validity’. In: *5th International Workshop on Trends in Linear Logic and Applications (TLLA 2021).* Rome (virtual), Italy, June 2021. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-03271408.

[61] 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.

[62] 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.

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

[64] 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.

[65] 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.

[66] Y. Guiraud. ‘Présentations d’opérades et systèmes de réécriture’. PhD thesis. Univ. Montpellier 2, 2004.

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

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

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

[70] 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.

[71] 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.

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

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

[74] 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.

[75] 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.

[76] 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`.

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

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

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

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

[81] 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.

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

[83] J.-J. Levy. 'Tracking Redexes in the Lambda Calculus'. working paper or preprint. Feb. 2022. URL: `https://hal.inria.fr/hal-03564707`.

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

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

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

[87] 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`.

[88] 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.

[89] 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.

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

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

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

[93] 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`.

[94] 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`.