RESEARCH CENTRE

Paris

IN PARTNERSHIP WITH:

CNRS, Université Denis Diderot (Paris 7)

2020

ACTIVITY REPORT

Project-Team

PI.R2

Design, study and implementation of languages for proofs and programs

IN COLLABORATION WITH: Institut de Recherche en Informatique Fondamentale

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Contents

Project-Team PI.R2

Creation of the Team: 2009 January 01, updated into Project-Team: 2011 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]
- Emilio Jesus Gallego Arias [Inria, Starting Research Position]
- Yves Guiraud [Inria, Researcher, HDR]
- Hugo Herbelin [Inria, Senior Researcher, HDR]
- Jean-Jacques Lévy [Inria, Emeritus, HDR]

Faculty Members

- Pierre Letouzey [Université de Paris, Associate Professor]
- Yann Régis Gianas [Université de Paris, Associate Professor, Until september 1st 2020, (on leave at Nomadic Labs), HDR]

Post-Doctoral Fellow

- Amar Hadzihasanovic [Sorbonne Université, until Nov 2020]

PhD Students

- Antoine Allioux [Inria]
- Vincent Blazy [Université de Paris, from Sep 2020]
- Félix Castro [Université de Paris, from Sep 2019, Cotutelle avec l'université de la république d'Uruguay, codirecteur Alexandre Miquel]
- Kostia Chardonnet [Université Paris-Saclay]
- El Mehdi Cherradi [Conseil général de l'économie, de l'industrie, de l'énergie et des technologies, from Sep 2020]
- Abhishek De [Université de Paris]
- Colin Gonzalez [Nomadic Labs, CIFRE]
- Cedric Ho Thanh [Université de Paris, until Sep 2020]
- Farzad Jafar Rahmani [Université de Paris]
- Hugo Moeneclaey [Université Paris-Saclay, from Sep 2019]
- Alen Ðurić [Université de Paris]

Technical Staff

- Thierry Martinez [Inria, Engineer, from Oct 2020]
- Théo Zimmermann [Inria, Engineer]
- Daniel de Rauglaudre [Inria, Engineer]

Interns and Apprentices

- Julien Coolen [Inria, from Jun 2020 until Aug 2020]

Administrative Assistant

- Anne Mathurin [Inria]

2 Overall objectives

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

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

3 Research program

3.1 Proof theory and the Curry-Howard correspondence

3.1.1 Proofs as programs

Proof theory is the branch of logic devoted to the study of the structure of proofs. An essential contributor to this field is Gentsen [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 [56], then by Howard and de Bruijn at the end of the 60's [75, 46], that proofs had the very same structure as programs: for instance, natural deduction proofs can be identified as typed programs of the ideal programming language known as λ-calculus.

This proofs-as-programs correspondence has been the starting point to a large spectrum of researches and results contributing to deeply connect logic and computer science. In particular, it is from this line of work that Coquand and Huet's Calculus of Constructions [53, 54] stemmed out – a formalism that is both a logic and a programming language and that is at the source of the Coq system [96].

3.1.2 Towards the calculus of constructions

The λ-calculus, defined by Church [52], is a remarkably succinct model of computation that is defined via only three constructions (abstraction of a program with respect to one of its parameters, reference to such a parameter, application of a program to an argument) and one reduction rule (substitution of the formal parameter of a program by its effective argument). The λ-calculus, which is Turing-complete, i.e. which has the same expressiveness as a Turing machine (there is for instance an encoding of numbers as functions in λ-calculus), comes with two possible semantics referred to as call-by-name and call-by-value evaluations. Of these two semantics, the first one, which is the simplest to characterise, has been deeply studied in the last decades [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 [86].

In 1985, Coquand and Huet [53, 54] in the Formel team of INRIA-Rocquencourt explored an alternative approach based on Girard-Reynolds' system F [64, 92]. This formalism, called the Calculus of Constructions, served as logical foundation of the first implementation of Coq in 1984. Coq was called CoC at this time.

3.1.3 The Calculus of Inductive Constructions

The first public release of CoC dates back to 1989. The same project-team developed the programming language Caml (nowadays called OCaml and coordinated by the Gallium team) that provided the expressive and powerful concept of algebraic data types (a paragon of it being the type of lists). In CoC, it was possible to simulate algebraic data types, but only through a not-so-natural not-so-convenient encoding.

In 1989, Coquand and Paulin [55] designed an extension of the Calculus of Constructions with a generalisation of algebraic types called inductive types, leading to the Calculus of Inductive Constructions (CIC) that started to serve as a new foundation for the Coq system. This new system, which got its current definitive name Coq, was released in 1991.

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

3.2 The development of Coq

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

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

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

In the last seven years, Hugo Herbelin and Matthieu Sozeau coordinated the development of the system, the official coordinator hat passed from Hugo to Matthieu in August 2016. The ecosystem and development model changed greatly during this period, with a move towards an entirely distributed development model, integrating contributions from all over the world. While the system had always been open-source, its development team was relatively small, well-knit and gathered regularly at Coq working groups, and many developments on Coq were still discussed only by the few interested experts.

The last years saw a big increase in opening the development to external scrutiny and contributions. This was supported by the "core" team which started moving development to the open GitHub platform (including since 2017 its bug-tracker [98] 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 [42], the more advanced Certified Programming with Dependent Types book by Chlipala [51] and the recent book on the Mathematical Components library by Mahboubi, Tassi et al. provide resources for gradually learning the tool.

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

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

We next briefly describe the main components of Coq.

3.2.1 The underlying logic and the verification kernel

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

3.2.2 Programming and specification languages

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

3.2.3 Standard library

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

3.2.4 Tactics

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

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

3.2.5 Extraction

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

3.2.6 Documentation

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

3.2.7 Proof development infrastructure

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

3.3 Dependently typed programming languages

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

DTP contributes to a general movement leading to the fusion between logic and programming. Coq, whose language is both a logic and a programming language which moreover can be extracted to pure ML code plays a role in this movement and some frameworks combining logic and programming have been proposed on top of Coq (Concoqtion at Rice and Colorado, Ynot at Harvard, Why in the ProVal team at INRIA, Iris at MPI-Saarbrucken). It also connects to Hoare logic, providing frameworks where pre- and post-conditions of programs are tied with the programs.

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

3.3.1 Type-checking and proof automation

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

3.4 Around and beyond the Curry-Howard correspondence

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

3.4.1 Control operators and classical logic

Indeed, a significant extension of the Curry-Howard correspondence has been obtained at the beginning of the 90's thanks to the seminal observation by Griffin [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 [82] and Reynolds [91] and started to be studied in an abstract way in the 80's by Felleisen *et al* [60], leading to Parigot's $\lambda\mu$-calculus [89], 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 [83] and Krivine's abstract machine for call-by-name evaluation [79, 78]. 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 [61]: 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 [59]).

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 [95], and independently by Burroni [48], 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 [80, 85, 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 [44].

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 [93, 94]: 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 [81, 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

6.1 Awards

Pierre-Louis Curien was the recipient of the Grand Prix Inria - Académie des Sciences 2020. (`https://www.academie-sciences.fr/fr/Laureats/prix-inria-academie-des-sciences-2020.html`)

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 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: Some highlights from this release are:

- Introduction of primitive persistent arrays in the core language, implemented using imperative persistent arrays. - Introduction of definitional proof irrelevance for the equality type defined in the SProp sort. - Many improvements to the handling of notations, including number notations, recursive notations and notations with bindings. A new algorithm chooses the most precise notation available to print an expression, which might introduce changes in printing behavior.

See the Zenodo citation for more information on this release: https://zenodo.org/record/4501022#.YB00r5NKjlw

News of the Year: Coq version 8.13 integrates many usability improvements, as well as extensions of the core language. The main changes include: - Introduction of primitive persistent arrays in the core language, implemented using imperative persistent arrays. - Introduction of definitional proof irrelevance for the equality type defined in the SProp sort. - Cumulative record and inductive type declarations can now specify the variance of their universes. - Various bugfixes and uniformization of behavior with respect to the use of implicit arguments and the handling of existential variables in declarations, unification and tactics. - New warning for unused variables in catch-all match branches that match multiple distinct patterns. - New warning for Hint commands outside sections without a locality attribute, whose goal is to eventually remove the fragile default behavior of importing hints only when using Require. The recommended fix is to declare hints as export, instead of the current default global, meaning that they are imported through Require Import only, not Require. See the following rationale and guidelines for details. - General support for boolean attributes. - Many improvements to the handling of notations, including number notations, recursive notations and notations with bindings. A new algorithm chooses the most precise notation available to print an expression, which might introduce changes in printing behavior. - Tactic improvements in lia and its zify preprocessing step, now supporting reasoning on boolean operators such as Z.leb and supporting primitive integers Int63. - Typing flags can now be specified per-constant / inductive. - Improvements to the reference manual including updated syntax descriptions that match Coq's grammar in several chapters, and splitting parts of the tactics chapter to independent sections.

See the changelog for an overview of the new features and changes, along with the full list of contributors. https://coq.github.io/doc/v8.13/refman/changes.html#version-8-13

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

Authors: Bruno Barras, Yves Bertot, Frédéric Besson, Pierre Corbineau, Cristina Cornes, Judicaël Courant, Pierre Courtieu, Pierre Crégut, David Delahaye, Maxime Denes, Jean-Christophe Filliâtre, Julien Forest, Emilio Jesus Gallego Arias, Gaëtan Gilbert, Georges Gonthier, Benjamin Grégoire, Hugo Herbelin, Gérard Huet, Vincent Laporte, Pierre Letouzey, Assia Mahboubi, Pascal Manoury, Guillaume Melquiond, César Munoz, Chetan Murthy, Amokrane Saibi, Catherine Parent, Christine

Paulin Mohring, Pierre-Marie Pédrot, Loïc Pottier, Matthieu Sozeau, Arnaud Spiwack, Enrico Tassi, Laurent Théry, Benjamin Werner, Théo Zimmermann

Contacts: Hugo Herbelin, Matthieu Sozeau

Participants: Yves Bertot, Frédéric Besson, Tej Chajed, Cyril Cohen, Pierre Corbineau, Pierre Courtieu, Maxime Denes, Jim Fehrle, Julien Forest, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Georges Gonthier, Benjamin Grégoire, Jason Gross, Hugo Herbelin, Vincent Laporte, Olivier Laurent, Assia Mahboubi, Kenji Maillard, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, ClÉment Pit-Claudel, Kazuhiko Sakaguchi, Vincent Semeria, Michael Soegtrop, Arnaud Spiwack, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Anton Trunov, Li-Yao Xia, Théo Zimmermann

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

7.1.2 Equations

Keywords: Coq, Dependent Pattern-Matching, Proof assistant, Functional programming

Scientific Description: Equations is a tool designed to help with the definition of programs in the setting of dependent type theory, as implemented in the Coq proof assistant. Equations provides a syntax for defining programs by dependent pattern-matching and well-founded recursion and compiles them down to the core type theory of Coq, using the primitive eliminators for inductive types, accessibility and equality. In addition to the definitions of programs, it also automatically derives useful reasoning principles in the form of propositional equations describing the functions, and an elimination principle for calls to this function. It realizes this using a purely definitional translation of high-level definitions to core terms, without changing the core calculus in any way, or using axioms.

The main features of Equations include:

Dependent pattern-matching in the style of Agda/Epigram, with inaccessible patterns, with and where clauses. The use of the K axiom or a proof of K is configurable, and it is able to solve unification problems without resorting to the K rule if not necessary.

Support for well-founded and mutual recursion using measure/well-foundedness annotations, even on indexed inductive types, using an automatic derivation of the subterm relation for inductive families.

Support for mutual and nested structural recursion using with and where auxilliary definitions, allowing to factor multiple uses of the same nested fixpoint definition. It proves the expected elimination principles for mutual and nested definitions.

Automatic generation of the defining equations as rewrite rules for every definition.

Automatic generation of the unfolding lemma for well-founded definitions (requiring only functional extensionality).

Automatic derivation of the graph of the function and its elimination principle. In case the automation fails to prove these principles, the user is asked to provide a proof.

A new dependent elimination tactic based on the same splitting tree compilation scheme that can advantageously replace dependent destruction and sometimes inversion as well. The as clause of dependent elimination allows to specify exactly the patterns and naming of new variables needed for an elimination.

A set of Derive commands for automatic derivation of constructions from an inductive type: its signature, no-confusion property, well-founded subterm relation and decidable equality proof, if applicable.

Functional Description: Equations is a function definition plugin for Coq (supporting Coq 8.11 to 8.13, with special support for the Coq-HoTT library), that allows the definition of functions by dependent pattern-matching and well-founded, mutual or nested structural recursion and compiles them into core terms. It automatically derives the clauses equations, the graph of the function and its associated elimination principle.

Equations is based on a simplification engine for the dependent equalities appearing in dependent eliminations that is also usable as a separate tactic, providing an axiom-free variant of dependent destruction.

Release Contributions: This version of Equations is based on an improved simplification engine for the dependent equalities appearing during dependent eliminations that is also usable as a separate dependent elimination tactic, providing an axiom-free variant of dependent destruction and a more powerful form of inversion.

This is a bugfix release of version 1.2 working with Coq 8.11 to Coq 8.13 See https://mattam82.github.io/Coq-Equations/equations/2019/05/17/1.2.html for the 1.2 release notes. New in this version:

Fixed issue #297: dependent elimination simplification mistreated let-bindings Fixed issue #295: discrepancy between the syntax in the manual and the implementation Ensure that NoConfusion is derived before EqDec as it is necessary to solve the corresponding obligation.

News of the Year: Equations 1.2.3, released first in June 2020 brings bugfixes, a fixed grammar and more robust proof automation tactics.

URL: `http://mattam82.github.io/Coq-Equations/`

Publications: hal-01671777, hal-01248807, inria-00628862

Authors: Cyprien Mangin, Matthieu Sozeau, Matthieu Sozeau

Contact: Matthieu Sozeau

Participants: Matthieu Sozeau, Cyprien Mangin

7.1.3 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: `http://www.lix.polytechnique.fr/Labo/Samuel.Mimram/rewr`

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

Authors: Yves Guiraud, Samuel Mimram

Contact: Yves Guiraud

Participants: Yves Guiraud, Samuel Mimram

7.1.4 Catex

Keywords: LaTeX, String diagram, Algebra

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

URL: `https://www.irif.fr/~guiraud/catex/catex.zip`

Author: Yves Guiraud

Contact: Yves Guiraud

Participant: Yves Guiraud

7.1.5 Cox

Keywords: Computer algebra system (CAS), Rewriting systems, Algebra

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

URL: `https://www.irif.fr/~guiraud/cox/cox.zip`

Publications: hal-00682233, hal-00818253

Author: Yves Guiraud

Contact: Yves Guiraud

Participant: Yves Guiraud

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

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

Publication: hal-01425752

Contact: Emilio Jesus Gallego Arias

Participant: Emilio Jesus Gallego Arias

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

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

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

Publication: hal-01384408

Contact: Emilio Jesus Gallego Arias

Participant: Karl Palmskog

Partner: KTH Royal Institute of Technology

7.1.8 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: Théo Zimmermann

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 A theory of effects and resources

In collaboration with Thomas Letan (ANSSI), Yann Régis-Gianas developed and proved several properties of a simple web server implemented in Coq using FreeSpec, a compositional framework to verify effectful software components in Coq. This work has been presented at CPP 2020 [28, 22].

8.1.2 Explicit syntax for stores

Hugo Herbelin et Étienne Miquey developed a calculus of explicit stores describing the operations and rewriting rules needed to explicitly manage and type extensible environments as part of a syntactic calculus [27]. This is similar to calculi with explicit substitutions turning the meta-operation of substitution into a syntactic one, but adding here the possibility to dynamically expand or modify the substitution.

8.1.3 Computational contents of the axiom of choice

Félix Castro started his PhD under the joint supervision of Hugo Herbelin (INRIA) and Alexandre Miquel (IMERL, Facultad de Ingeniería, Universidad de la República, Uruguay) in September 2019. His PhD work focuses on the computational contents of Gödel's constructible universe, which, in particular, justifies the Axiom of Choice. (Previously, he worked on the formalisation of the ramified analytical hierarchy in classical second-order arithmetic.)

During his first year, he worked on the computational contents of logical translations by relativization (such as the syntactical translation implementing Gödel's constructible universe). His approach aims at establishing that, as the double-negation-translation justifies the addition of continuations in a programming language (representing proofs of an intuitionistic logic via the Curry-Howard correspondence), this kind of logical translation (by relativization) justifies the addition of a "quote" instruction in the programming counter-part of the logical system which is the source of the translation.

Hugo Herbelin developed in collaboration with Nuria Brede (U. Potsdam) a unified approach of the underlying logical structure of choice and bar induction principles [32].

8.1.4 Proof-search in proof nets

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

His approach consists in considering the correctness problem of proof-structure for the larger class of para-proof structures (ie proof-structure admitting a generalized axiom deriving any sequent) viewing proof-search and sequentialisation as dual aspects of partial proof structures (that is proof nets with open premisses). Revisiting in particular two related correctness criteria in the framework of para-proof-structures (contractibility and Lafont's parsing criterion), he obtains a proof-construction algorithm in which the proof space is not a search tree, as in sequent-calculus, but a dag allowing to share proof-construction paths, for which proof-construction corresponds to "unparsing". A paper is currently being written.

8.1.5 Algebraic Logic Programming

Emilio Jesús Gallego Arias collaborated with Jim Lipton from Wesleyan University on the development of algebraic models for proof search in the context of logic programming. In particular, we have extended the semantics of [62] to better account for the recursive definition of relations, using techniques inspired in step-indexing [43]. A key point of the extension is that the *index* category does account for the particular proof search strategy as understood in logic programming. For example, the category $\langle \mathbb{N}, < \rangle$ provides the categorical setting for breadth-first search.

8.2 Reasoning and programming with infinite data

Participants Kostia Chardonnet, Abhishek De, Colin Gonzalez, Farzad Jafarrahmani, Yann Régis-Gianas, Alexis Saurin.

8.2.1 Proof theory of non-wellfounded and circular proofs

Validity conditions of infinitary and circular proofs In collaboration with David Baelde, Amina Doumane and Denis Kuperberg, Alexis Saurin extended the proof theory of infinite and circular proofs for fixed-point logics in various directions by relaxing the validity condition necessary to distinguish sound proofs from invalid ones. The original validity condition considered by Baelde, Doumane and Saurin in CSL 2016 [1] rules out lots of proofs which are computationally and semantically sound and does not account for the cut-axiom interaction in sequent proofs. In the setting of sequent calculus, Alexis Saurin studied together with David Baelde, Amina Doumane and Denis Kuperberg a relaxed validity condition to allow infinite branches to be supported by threads which may leave the infinite branch, visiting other parts of the proofs and bounce on axioms and cuts. This allows for a much more flexible criterion, inspired from Girard's geometry of interaction. The most general form of this criterion does not ensure productivity in the sequent calculus due to a discrepancy between the sequential nature of proofs in sequent calculus and the parallel nature of threads. David Baelde, Amina Doumane, Denis Kuperberg and Alexis Saurin provided a slight restriction of the full bouncing validity which grants productivity and validity of the cut-elimination process. This restriction still strictly extends previous notions of validity and is actually expressive enough to be undecidable.

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

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

A pre-publication is available[38].

Proof nets for non-wellfounded proofs Abhishek De and Alexis Saurin pursued their investigation of non-wellfounded and circular proofs nets (in the multiplicative setting), building on first results published in 2019 [57]. Together with Luc Pellissier (LACL, Université Paris Est-Créteil), they generalized infinets solving some restrictions of the original presentation with respect to the use of the cut-inference. Considering potentially infinite (non-wellfounded) proof-objects, it is natural to allow for infinitely many cut rules (typically when a cut occurs within a cycle of a regular proof). Their original proposal for non-wellfounded proof-nets only allowed finitely many cuts. New results show how to integrate infinitely many cuts, while proposing a new presentation of the productive cut-elimination on $\mu MALL$ sequent calculus.

Regularization of non-wellfounded proofs In a collaboration with Anupam Das (university of Birmingham), Abhishek De and Alexis Saurin studied regularization of non-wellfounded proofs in the sequent calculus of fixed-point logics, that is determining under which condition the existence of a non-wellfounded proof ensures the existence of a circular proof. After studying the setting of classical μ-calculus and logics in which finiteness assumptions can be made of the space of sequents, they are currently investigating the case of linear logic with fixed points in which no such assumption can be made.

This collaboration started in the spring 2020, during the Covid crisis and a planned visit of De to Birmingham had to be postponed till now. It is expected to occur in 2021.

8.2.2 On the denotational semantics of finitary and non-wellfounded proofs

Farzad Jafar-Rahmani started his PhD under the supervision of Thomas Ehrhard and Alexis Saurin in October 2019. His PhD work will focus on the denotational semantics of finitary and circular proofs of linear logic with fixed points.

Together with Ehrhard, he developed a categorical semantics of μLL based on Seely categories and on strong functors acting on them, exhibiting two instances of this categorical setting: a relational semantics in which least and greatest fixed points are interpreted in the same way and a more refined interpretation in which sets are equipped with a notion of totality (non-uniform totality spaces), the relations preserving the totality. This later interpretation distinguishes least and greatest fixedpoints and shows that μLL enjoys a denotational form of normalization of proofs.

The three of them are currently working on the interpretation of circular proofs and of the denotational counterpart of the validity condition of circular proofs. Moreover, they developed a system L for a polarized calculus and its categorical semantics.

8.2.3 Quantum programming languages with inductive and coinductive types

Kostia Chardonnet started his PhD under the supervision of Alexis Saurin and Benoît Valiron in November 2019. 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). Chardonnet, Saurin and Valiron published a first work [26] presenting 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 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.

8.2.4 Simplifying smartcontract programming via stream programming and spreadsheets

Colin Gonzalez is preparing a CIFFRE PhD under the joint supervision of Yann Régis-Gianas, Adrien Guatto and Ralf Treinen. His work aim at programming smartcontracts more easily by taking inspiration from spreadsheets and stream programming. To do this, his goal is to build a certified spreadsheet compiler to Michelson's smartcontracts.

Building on the similarity between the type a smartcontract executions and the type of streams, they first defined a programming language (Lisa) allowing structured programming of spreadsheets to be translated to a stream algebra. First results are the design of a Lisa interpreter, a propotype compiler from a toy spreadsheet language to Lisa and a compiler from a fragment of Lustre to Lisa.

8.3 Effective higher-dimensional algebra

Participants Antoine Allioux, Pierre-Louis Curien, Alen Đurić, Yves Guiraud, Amar Hadzihasanović, Cédric Ho Thanh.

8.3.1 Rewriting methods in higher algebra

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

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

Building on [6], Yves Guiraud is currently finishing with Matthieu Picantin (Univ. Paris Diderot) a work that generalises already known constructions such as the bar resolution, several resolutions defined by Dehornoy and Lafont [58], and the main results of Gaussent, Guiraud and Malbos on coherent presentations of Artin monoids [8], to monoids with a Garside family. This allows an extension of the field of application of the rewriting methods to other geometrically interesting classes of monoids, such as the dual braid monoids.

Still with Matthieu Picantin, Yves Guiraud develops an improvement of the classical Knuth-Bendix completion procedure, called the KGB (for Knuth-Bendix-Garside) completion procedure. The original algorithm tries to compute, from an arbitrary terminating rewriting system, a finite convergent presentation, by adding relations to solve confluence issues. Unfortunately, this algorithm fails on standard examples, like most Artin monoids with their usual presentations. The KGB procedure uses the theory of Tietze transformations, together with Garside theory, to also add new generators to the presentation, trying to reach the convergent Garside presentation identified in [6]. The KGB completion procedure is partially implemented in the prototype Rewr, developed by Yves Guiraud and Samuel Mimram.

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

8.3.2 Coherent presentations of monoids

Alen Đurić, in collaboration with Pierre-Louis Curien and Yves Guiraud, has found a common generalisation of two distinct generalisations of a result originally due to Deligne, who had shown how to give coherent presentations of aspherical Artin monoids. Here, coherent means, given a presentation by generators and relations, that a number of generating "2-relations" between the relations is provided that suffices to show that "all diagrams involving the relations" can be paved by these generating 2-relations (this is the beginning of a higher-dimensional story). The two distinct generalisations have been given by Gaussent, Guiraud and Malbos and concern Artin groups without the aspherical restriction on the one hand, and so-called Garside monoids on the other hand. The common generalisation concerns a large family of monoids admitting a Garside family. Beyond the technical details, what is important in this new development is to have worked out general conditions that make the coherence argument work. This work is on the edge of being submitted to a journal.

8.3.3 Topological aspects of polygraphs

Amar Hadzihasanović has been a postdoc of the team (funded by FSMP) from end of November 2019 to end of September 2020. He has been working intensively on the study of shapes appropriate for the description of higher cells as needed in various approaches to higher categories and higher structures. During his stay, despite of the sanitary problems, Amar managed to considerably revise his last work "Diagrammatic sets and rewriting in weak higher categories". He also started a collaboration with Pierre-Louis Curien on a variation of the category of opetopes (see next section).

8.3.4 Opetopes

Cédric Ho Thanh defended his PhD thesis on October 15, 2020 [31]. Since November 2020, he is a postdoc in Ichiro Hasuo's lab, National Institute for Informatics, Tokyo. During the last year of his PhD, he pursued his collaboration with Chaitanya Leena Subramaniam on "opetopic algebras". Their most important result is to have found a common generalisation of the model category constructions of Joyal on simplicial sets on one hand, and of Moerdijk and coauthors on dendroidal sets. This work is part of his PhD thesis, and also submitted to journals[33].

The on-going collaboration of Pierre-Louis Curien and Amar Hadzihasanović mentioned above concerns a treatment of opetopes and opetopic sets by means of techniques from poset topology: those techniques are well-adapted for the description of an alternative notion of category of opetopes, where the objects are restricted to be positive opetopes, but the morphisms are more liberal: in this category, the treatment of degeneracies migrates from objects to morphisms. The current focus of the joint research is to find a good presentation of this alternative category by generators and relations.

8.3.5 Foundations and formalisation of higher algebra

Antoine Allioux (PhD started in February 2018), Eric Finster, Yves Guiraud and Matthieu Sozeau are exploring the development of higher algebra in type theory. To formalise higher algebra, one needs a new source of coherent structures in type theory. During the first year of Allioux's PhD, they studied an internalisation of polynomial monads (of which opetopes and ∞-categories are instances) in type theory, which ought to provide such a coherent algebraic structure, inspired by the work of Kock et al [77]. They later realised that this internalisation is however incoherent as presented in pure type theory, essentially because of its reliance on equality types.

Since then, they switched to a different view, instead extending type theory with a universe of strict polynomial monads. These strict structures allow, in turn, to present internally weak structures such as $(\infty, 1)$-categories. In particular, they were able to internalize the known result that types are ∞-groupoids [97, 84]. All attempts to formalize these weak structures in pure type theory had failed so far and it was conjectured that it was impossible to formalize them without modifying type theory. A paper has been submitted early 2021 [34]. They are now concentrating on developing the theory of $(\infty, 1)$-categories in this new framework.

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.

8.4.1 Meta-programming and metatheory of Coq

Vincent Blazy, Hugo Herbelin and Pierre Letouzey started a work aiming at making explicit the universe subtyping in the Calculus of Constructions (master internship of Vincent Blazy, then start of his PhD thesis). 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.

8.4.2 Homotopy type theory and synthetic homotopy

Hugo Herbelin and Hugo Moeneclaey worked on the explicit construction of a model for internal iterated parametricity using cubes. The progresses on this very technical task have been presented at the HoTT-UF'20 workshop.

Hugo Moeneclaey gave an alternative approach to building models for external iterated parametricity, sidestepping most technical problems using the theory of locally presentable categories. This method is applicable to build models for other interpretations of type theory.

Hugo Moeneclaey worked on a presentation of identity types in an arbitrary non-recursive higher inductive type as higher inductive types themselves. As an example the loop space of the circle is the type freely generated by a point and an auto-equivalence. This was done using the definition of higher inductive types by Kaposi and Kovacs. This work is still in progress.

8.4.3 Computational characterizations of relevant subsystems of second order arithmetic

Hugo Herbelin and Firmin Martin developed a variant of Girard-Reynolds' System F based on the axiom of separability rather than the axiom of comprehension. They derive from it a realisability semantics for the systems WKL_0 and ATR_0, two systems of the reverse mathematics of second order arithmetic which are precisely characterized by such a separability axiom (existence of a set separating any two disjoint formulas from a given class of arithmetic formulas).

8.4.4 Dependent pattern-matching

From August 2020, Thierry Martinez resumed 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. Significantly enough, he exploited OCaml's GADT to ensure strong static invariants of his implementation.

8.4.5 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. His practical contributions for 2020 include:

- a restructuration of Coq's reference manual and the review of many documentation pull requests (in collaboration with the external contributor Jim Fehrle) that led to significant improvements of the documentation of Coq (see the details in Coq's changelog for version 8.12);

- participation to the effort (led by the external contributor Michael Soegtrop) to create a "Coq platform", that will be the new official distribution method for both the Coq software and a selection of important Coq packages;

- diverse community management and tooling contributions with various community members, including the management of the coq-community organization (which he founded during his PhD thesis in 2018) and which now hosts over 50 projects by over 30 maintainers.

During the summer, Théo Zimmermann supervised the internship of Julien Coolen on the maintenance and evolution of coqbot. this bot, which was created by Théo Zimmermann during his PhD thesis in 2018, is relied on every day by the Coq developers for the management of the development activity on GitHub. This internship led to significant additions to the bot. One of the most impactful addition is a new feature allowing maintainers to merge pull requests with a comment. Previously, the maintainers had to use a script and set up a PGP key, and this had turned out to be a significant barrier of entry to the role of maintainer.

Théo Zimmermann's research focused on the model of community organizations for the collaborative maintenance of packages which he discovered during his PhD thesis and applied to create the coq-community organization mentioned above. He published and presented a paper about it in the ICSE workshop SoHEAL 2020 [29]. Since then, he has pursued this investigation further in collaboration with Jean-Rémy Falleri from LaBRI (Bordeaux).

Emilio J. Gallego Arias continued work on improving the incremental proof checking infrastructure for Coq, with a particular focus on the integration of Coq with the industrial-scale Dune build system. Jointly with the rest of the Dune development team, support for composition of Coq "theories" was added and released, together with many other fixes and improvements, including extraction and native compute support, and the full porting of the Coq codebase to Dune. This new tooling, despite its experimental nature, has seen a good adoption by users and researchers. Emilio J. Gallego Arias was invited to a one week "Dune Retreat" — funded by Jane Street — where extended discussion and work about this project was pursued with the rest of the Dune team.

Emilio J. Gallego Arias and Karl Palmskog released a new version of the Coq SerAPI tool, with improvements and new functionality to enable new applications such as [87]. Significant feedback from researchers has been gathered as to improve the tool, with a particular focus on mechanized-treatment of proofs by diverse tooling such as machine learning [50] or software engineering tooling.

Emilio J. Gallego Arias and Shachar Itzhaky released a new version of the Coq educational frontend jsCoq [7], with significant improvements both in the frontend, the backend, and the package system. In particular, jsCoq has reached the milestone where we consider that standard Coq textbooks such as Software Foundations run in a stable fashion. jsCoq has been used in several Coq courses in the 2020 year (we estimate at least in the dozens).

Emilio J. Gallego Arias maintains an ongoing collaboration with the Deducteam group at INRIA Saclay on the topic of interactive proof methods and standards; work in 2020 has included the co-supervision of two interns and improvements towards the specification and use of the Language Server Protocol in the context of interactive proof assistants.

8.4.6 Maintenance, development and coordination of the development of Coq

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.

Emilio J. Gallego Arias and Théo Zimmerman took the roles of release managers for the Coq 8.12 release cycle. Coq 8.12.0 was released in July, Coq 8.12.1 was released in November and Coq 8.12.2 was released in December.

Cyril Cohen (Stamp) and Théo Zimmermann replaced the Coq Gitter communication channels by a Zulip server supporting multiple topics. Hundreds of messages are posted every week.

In 2020, Hugo Herbelin contributed about 200 pull requests to Coq, covering maintenance, bug fixes, extensions of the support for notations, refinements of the phase of elaboration of user syntax to checked expresions, code cleanup (especially regarding implicit arguments and notations).

Emilio J. Gallego Arias contributed over 300 changes to Coq of diverse nature, most of them to improve the Coq codebase in preparation for further work on incremental and multi-core aware type checking. He also reviewed and merged over 200 pull requests, with around 2000 total interactions [comments, issues, etc...].

8.5 Formalisation and verification

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

8.5.1 Proofs and surfaces

The joint work of Pierre-Louis Curien with Jovana Obradović (former PhD student of the team and now researcher at the Institute of Mathematics of the Serbian Academy of Sciences), Zoran Petrić and other Serbian colleagues on designing a formal deductive system capturing the proofs of incidence theorems has been published in the journal Annals of Pure and Applied Logic [23].

8.5.2 A Coq formalisation of the first-order predicate calculus

In relation with a logic course for master students, Pierre Letouzey continued a Coq formalisation of the first-order predicate calculus, with the help of Samuel Ben Hamou (internship, 1st year ENS Saclay). The logical rules are expressed in a natural deduction style (with explicit contexts), with two possible low-level representations of formulas (quantifiers with names or "locally nameless"). After a completeness theorem last year, this time the use of specific theories like Peano and ZF have been investigated. This development is available at `https://gitlab.math.univ-paris-diderot.fr/letouzey/natded`

8.5.3 Lexing and regular expressions in Coq

Pierre Letouzey and Yann Régis-Gianas continued working on a Coq formalisation of regular expressions (with complement and conjunction) and their Brzozowski derivatives. This year, Antimorov derivatives have also been studied in this Coq formalization. Many techniques have also 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.

8.5.4 Sensitivity Conjecture in Coq

Daniel de Rauglaudre started a formalization in Coq 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, etc.) had to be 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 at at `https://github.com/roglo/coq_sensitivity`.

8.5.5 Proofs of algorithms on graphs

Jean-Jacques Lévy and Chen Ran (a PhD student at the Institute of Software, Beijing) pursued their work about formal proofs of graph algorithms. Their goal is to provide computer-checked proofs of algorithms that remain the human-readable. In 2019, they presented at ITP 2019 a joint paper with Cyril Cohen, Stephan Merz and Laurent Théry on this work [49]. This article compared formal proofs in three different systems (Why3, Coq, Isabelle/HOL) of Tarjan's linear-time algorithm (1972) computing the strongly connected components in directed graphs.

In July 2020, Chen Ran passed her PhD degree at Iscas, Beijing (adviser Zhang Wenhui). Jean-Jacques Lévy currently works on a proof of the implementation of this algorithm with imperative programming and memory pointers. He also plans to produce formal proofs of other abstract algorithms such as the Hopcroft-Tarjan (1972) linear-time algorithm for planarity testing in undirected graphs.

8.5.6 Iterated parametricity and semi-cubical sets

Hugo Herbelin and Ramkumar Ramachandra started the formalization in Coq of an original dependently-typed construction of semi-cubical sets inspired by the parametricity translation (see Section 8.4.2). This highly stressed the limits of Coq.

8.5.7 Datalog and low-level binary analysis

Emilio J. Gallego Arias collaborated with Stefania Dumbrava and Cody Roux on the use of our verified Datalog engine [45] for the analysis of low-level binary code; we have made significant progress towards the verification in Coq of the alias analysis proposed in [47].

8.5.8 Mechanism design

Emilio J. Gallego Arias collaborated with Pierre Jouvelot and Lucas Sguerra on the formalized verification of the general Vickrey-Clarke-Groves (see for example [88]) mechanism using Coq. The work has resulted in a paper submission early 2021.

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.

10 Partnerships and cooperations

10.1 International initiatives

10.1.1 Inria associate team not involved in an IIL

Pierre-Louis Curien is a member of the Equipe associée CRECOGI (Concurrent, Resourceful and Effectful COmputation, by Geometry of Interaction) coordinated by Ugo dal Lago (Focus team, Bologna), with the National Institute of Informatics, Tokyo (Ichiro Hasuo) which planned to have its final meeting as a Shonan meeting, Japan, in November 2020. This event has been postponed to November 2021.

Pierre-Louis Curien is coordinator on the French side of the project VIP (Verification, Interaction, and Proofs) (2017-2020) of the Inria - Chinese Academy of Sciences (CAS) program, with the Institute of Sotware of the CAS. The final meeting of the project, to be held in Beijing, has been postponed to 2021. The Chinese coordinator Ying Jiang has retired and been replaced as coordinator by Peng Wu.

10.2 International research visitors

10.2.1 Visits to international teams

Research stays abroad As part of his joint-PhD, Félix Castro spent most of 2020 abroad: after attending a summer school of COQ in Chile (CASS 2020) in January 2020, he spent the rest of the year in Uruguay, coming back to France in December 2020.

Pierre-Louis Curien was awarded a 6-week Research fellowship at the Oberwolfach Mathematical Research Institute (Germany) (split in two stays, in July and September 2020). He worked there on his new course on homotopical algebra and higher categories that he will teach at the master LMFI of Université de Paris.

Emilio J. Gallego Arias was awarded a 3-month Van Vleck Research Fellowship at Weysleyan University (Connecticut, USA) with the goal to continue work with Jim Lipton on the development of algebraic models for proof search. The stay was planned to take place February-April 2020, but was interrupted half-way due to the COVID crisis.

10.3 European initiatives

Pierre-Louis Curien and Alexis Saurin are members of GDRI-LL, an international joint GDR between France and Italy, supported by CNRS.

10.4 National initiatives

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

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

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

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

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

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

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

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

Yann Régis-Gianas is a member of the ANR COLIS dedicated to the verification of Linux Distribution installation scripts. This project is joint with members of VALS (Univ Paris Sud) and LIFL (Univ Lille). Yann Régis-Gianas published a paper about the verification of POSIX shell scripts at TACAS[41] and a journal paper presenting Morbig, a static parser for POSIX shell [90].

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).

10.5 Regional initiatives

Yves Guiraud is an elected member of the “Comité de Centre” of the “Centre INRIA de Paris”.

Alexis Saurin is a member of the Emergence – Ville de Paris project "Realise" led by Christine Tasson.

11 Dissemination

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

General chair, scientific chair Emilio Gallego, Hugo Herbelin and Théo Zimmermann chaired the Coq workshop 2020, which was planned as part of the IJCAR-FSCD joint conference and eventually happened entirely online due to the Covid-19 pandemic.

Yann Régis-Gianas was organizing chair of JFLA 2020, which took place in Gruissan from january 29th to february 1st.

11.1.2 Scientific events: selection

Chair of conference program committees Yann Régis-Gianas was co-chair of the JFLA 2020 program committee [30].

Member of conference program committees Théo Zimmermann was a program committee member for the 36th IEEE International Conference on Software Maintenance and Evolution (ICSME 2020).

Pierre-Louis Curien was a program committee member of the conference Formal Structures for Computation and Deduction (FSCD) 2020, as well as of its affiliated workshop "Geometric and Categorical Structures in Computation and Deduction”.

Reviewer Théo Zimmermann participated in the review process for the Haskell Conference 2020.

11.1.3 Journal

Member of the editorial boards Pierre-Louis Curien is editor-in-chief of the journal Mathematical Structrures in Computer Science.

11.1.4 Invited talks

Théo Zimmermann gave an invited talk at IRIT on the challenges in the collaborative evolution of Coq and its ecosystem.

Pierre-Louis Curien is an invited speaker at the Conference “Braids and beyond”, in memory of Patrick Dehornoy, which was planned in September 2020 in Caen and has been postponed to 2021.

Emilio J. Gallego Arias gave an invited talk at SystemX at the invitation of Kalpana Singh on the subject of applications to Coq to SystemX projects.

Hugo Herbelin gave an invited talk at the JFLA conference on cubical type theory.

11.1.5 Research administration

Yves Guiraud is an elected member of the “Conseil scientifique” of the “UFR de mathématiques de l’Université de Paris”

11.2 Teaching - Supervision - Juries

Teaching

- Master: Alexis Saurin, Outils classiques pour la correspondance de Curry-Howard, 24H, LMFI, Université de Paris. (spring 2020)
- Master: Alexis Saurin, Cours fondamental de théorie de la démonstration, 36H, LMFI, Université de Paris. (fall 2020)
- Master: Alexis Saurin, Programmation synchrone, 24H, M2 informatique, Université de Paris. (fall 2020)

- Master: Hugo Herbelin and Hugo Moeneclaey taught a class on Homotopy Type Theory, respectively 30H and 18H, LMFI, University de Paris. (winter 2020)
- Master: Pierre Letouzey teaches two short courses to the LMFI Master 2 students : "Programming in Coq" and "Introduction to computed-aided formal proofs" (2*24H). These two courses come in addition to Pierre Letouzey's regular duty as teacher in the Computer Science department of Université de Paris (including a course on Compilation to M2-Pro students and a course on computed-aided formal proofs to M1 students).
- Group tutorial: in his visit to Wesleyan university, Emilio J. Gallego Arias helped Jim Lipton on a "group tutorial" on Blockchain Technologies with 18 participating students. A group tutorial is a student-driven teaching format. The idea was to supervise a student on the use of Coq in the Smart Contract context but plans were altered by the sanitary crisis.

PhD Supervision

- PhD started: Vincent Blazy, Fine-grain 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 started: Colin Gonzalez, Certified compilation of spreadsheets as Tezos smart contract, Université de Paris, CIFRE contract started in September 2020, supervised by Yann Régis-Gianas, Benjamin Canou (Nomadic Labs), Adrien guatto and Ralf Treinen.
- PhD in progress: Félix Castro, Computational contents of the model of constructible sets, cotutelle between Université de Paris and Universidad de la República (Montevideo, Uruguay), started in September 2019, supervised by Hugo Herbelin and Alexandre Miquel.
- 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: 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 inprogress: 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 Soseau.
- 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 in progress: Rémi Nollet, Validity conditions for circular proofs, Université de Paris, since October 2016, supervised by Alexis Saurin and Christine Tasson.
- PhD ended: Cédric Ho Thanh, Opetopes for higher-dimensional rewriting and koszulity, Université de Paris, defended October 15 2020, supervised by Pierre-Louis Curien and Samuel Mimram.
- PhD ended: Simon Forest, Computational descriptions of higher categories, École Polytechnique, defended on 12 January 2021, supervised by Samuel Mimram (École Polytechnique) and Yves Guiraud

Internships

- Hugo Herbelin and Pierre Letouzey supervised the master internship of Vincent Blazy on an extension of the Calculus of Constructions by an explicit subtyping of Prop into an infinite universe hierarchy.
- Pierre Letouzey supervised the internship of Samuel Ben Hamou (1st year ENS Saclay) on the extensions of a 1st-order logic course certified in Coq.
- Théo Zimmermann supervised the internship of Julien Coolen (june - august 2020) on the maintenance and evolution of coqbot.
- Hugo Herbelin supervised the master internship of Firmin Martin (master 1) on a realizability semantics of the axiom of separability using a refined version of System F (see Section 8.1.3).
- Emilio J. Gallego co-supervised with Fréderic Blanqui two interns on the subject of user interfaces for the lambdapi theorem prover: François Lefoulon (1A, June-July 2020) and Ashish Kumar Barnawal (Dec-Jan 2020).

Juries

- Hugo Herbelin was a member of the PhD committee of Théo Winterhalter.

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] T. Letan, Y. Régis-Gianas, P. Chifflier and G. Hiet. 'Modular verification of programs with effects and effects handlers'. In: *Formal Aspects of Computing* (15th Dec. 2020). DOI: 10.1007/s00165-020-00523-2. URL: https://hal.archives-ouvertes.fr/hal-03107526.

[23] M. Milicevic, D. Baralic, J. Obradovic, Z. Petric, M. Zekic, R. Zivaljevic and P.-L. Curien. ‘Proofs and surfaces’. In: *Annals of Pure and Applied Logic* 171.9 (2020). URL: https://hal.archives-ouvertes.fr/hal-02410910.

[24] M. Sozeau, A. Anand, S. Boulier, C. Cohen, Y. Forster, F. Kunze, G. Malecha, N. Tabareau and T. Winterhalter. ‘The MetaCoq Project’. In: *Journal of Automated Reasoning* (Feb. 2020). DOI: 10.1007/s10817-019-09540-0. URL: https://hal.inria.fr/hal-02167423.

[25] M. Sozeau, S. Boulier, Y. Forster, N. Tabareau and T. Winterhalter. ‘Coq Coq Correct! Verification of Type Checking and Erasure for Coq, in Coq’. In: *Proceedings of the ACM on Programming Languages* (19th Jan. 2020), pp. 1–28. DOI: 10.1145/3371076. URL: https://hal.archives-ouvertes.fr/hal-02380196.

International peer-reviewed conferences

[26] K. Chardonnet, A. Saurin and B. Valiron. ‘Toward a Curry-Howard Equivalence for Linear, Reversible Computation’. In: RC 2020 - 12th international conference on Reversible Computation. Vol. LNCS. Reversible Computation 12227. Oslo / Virtual, Norway, 9th July 2020, pp. 144–152. DOI: 10.1007/978-3-030-52482-1_8. URL: https://hal.archives-ouvertes.fr/hal-03103455.

[27] H. Herbelin and É. Miquey. ‘A calculus of expandable stores: Continuation-and-environment-passing style translations’. In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS ’20), July 8–11, 2020, Saarbrücken, Germany*. LICS 2020 - 35th ACM/IEEE Symposium on Logic in Computer Science. Saarbrücken / Virtual, Germany, 8th July 2020, pp. 564–577. DOI: 10.1145/3373718.3394792. URL: https://hal.archives-ouvertes.fr/hal-02557823.

[28] T. Letan and Y. Régis-Gianas. ‘FreeSpec: Specifying, Verifying and Executing Impure Computations in Coq’. In: CPP 2020 - 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. Nouvelle-Orléans, United States, 20th Jan. 2020, pp. 1–15. DOI: 10.1145/3372885.3373812. URL: https://hal.inria.fr/hal-02422273.

[29] T. Zimmermann. ‘A first look at an emerging model of community organizations for the long-term maintenance of ecosystems’ packages’. In: SoHeal 2020 - 3rd International Workshop on Software Health. Seoul / Virtual, South Korea, 24th May 2020. DOI: 10.1145/3387940.3392209. URL: https://hal.inria.fr/hal-02534965.

Edition (books, proceedings, special issue of a journal)

[30] Z. L. Dargaye and Y. Regis-Gianas. *31ème Journées Francophones des Langages Applicatifs*. 31st Jan. 2020. URL: https://hal.inria.fr/hal-02427360.

Doctoral dissertations and habilitation theses

[31] C. Ho Thanh. ‘Opetopes: Syntactic and Algebraic Aspects’. Université de Paris, 15th Oct. 2020. URL: https://hal.archives-ouvertes.fr/tel-02968939.

Reports & preprints

[32] N. Brede and H. Herbelin. *On the logical structure of choice and bar induction principles*. 15th Mar. 2021. URL: https://hal.inria.fr/hal-03144849.

[33] C. Ho Thanh and C. Leena Subramaniam. *Opetopic algebras III: Presheaf models of homotopy-coherent opetopic algebras*. Jan. 2020. URL: https://hal.archives-ouvertes.fr/hal-02448208.

12.3 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, D. Kuperberg and A. Saurin. *Bouncing threads for infinitary and circular proofs*. 2020. arXiv: 2005.08257 [cs.LO].

[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] B. Becker, N. Jeannerod, C. Marché, Y. Régis-Gianas, M. Sighireanu and R. Treinen. 'Analysing installation scenarios of Debian packages'. In: *TACAS 2020 - 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Vol. 12079. Lecture Notes in Computer Science. Dublin, Ireland, Apr. 2020, pp. 235–253. DOI: 10.1007/978-3-030-45237-7_14. URL: https://hal.archives-ouvertes.fr/hal-02355602.

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

[43] L. Birkedal, R. E. Møgelberg, J. Schwinghammer and K. Støvring. 'First steps in synthetic guarded domain theory: step-indexing in the topos of trees'. In: *Logical Methods in Computer Science* 8.4 (2012). DOI: 10.2168/LMCS-8(4:1)2012. URL: https://doi.org/10.2168/LMCS-8(4:1)2012.

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

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

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

[47] D. Brumley and J. Newsome. *Alias Analysis for Assembly*. Tech. rep. Carnegie Mellon University, 2006.

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

[49] 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.

[50] C. Cheng, Y. Xiong, W. Huang and L. Ma. 'Context-aware Generation of Proof Scripts for Theorem Proving'. In: *2020 6th International Conference on Big Data Computing and Communications (BIGCOM)*. 2020, pp. 81–85. DOI: 10.1109/BigCom51056.2020.00018.

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

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

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

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

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

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

[57] 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.

[58] P. Dehornoy and Y. Lafont. 'Homology of Gaussian groups'. In: *Ann. Inst. Fourier (Grenoble)* 53.2 (2003), pp. 489–540. URL: http://aif.cedram.org/item?id=AIF_2003__53_2_489_0.

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

[60] 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.

[61] 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.

[62] E. J. Gallego Arias and J. Lipton. 'Logic Programming in Tabular Allegories'. In: *Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012, September 4-8, 2012, Budapest, Hungary*. Ed. by A. Dovier and V. Costa. Vol. 17. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012, pp. 334–347. DOI: 10.4230/LIPIcs.ICLP.2012.334.

[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. Kock, A. Joyal, M. Batanin and J.-F. Mascari. 'Polynomial functors and opetopes'. In: *Advances in Mathematics* 224.6 (2010), pp. 2690–2737. DOI: `https://doi.org/10.1016/j.aim.2010.02.012`. URL: `http://www.sciencedirect.com/science/article/pii/S0001870810000769`.

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

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

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

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

[82] 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.

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

[84] P. L. Lumsdaine. 'Weak ω-categories from intensional type theory'. In: *International Conference on Typed Lambda Calculi and Applications*. Springer. 2009, pp. 172–187.

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

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

[87] P. Nie, K. Palmskog, J. J. Li and M. Gligoric. 'Deep Generation of Coq Lemma Names Using Elaborated Terms'. In: *International Joint Conference on Automated Reasoning*. 2020, pp. 97–118. DOI: `10.1007/978-3-030-51054-1_6`.

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

[89] 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`.

[90] Y. Régis-Gianas, N. Jeannerod and R. Treinen. 'Morbig: A Static parser for POSIX shell'. In: *Journal of Computer Languages* 57 (2020), p. 100944. DOI: `https://doi.org/10.1016/j.cola.2020.100944`. URL: `https://www.sciencedirect.com/science/article/pii/S2590118420300046`.

[91] 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.

[92] 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.

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

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

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

[96] 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.

[97] B. Van Den Berg and R. Garner. 'Types are weak ω-groupoids'. In: *Proceedings of the london mathematical society* 102.2 (2011), pp. 370–394.

[98] 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.