

2025 Activity Report

RESEARCH CENTRE: Inria Lyon Centre

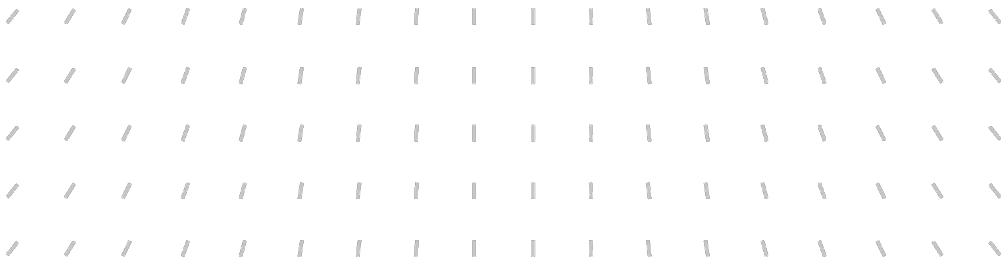
IN PARTNERSHIP WITH: Université Claude Bernard (Lyon 1), Ecole normale supérieure de Lyon, CNRS

Project-Team

CASH

Compilation and Analyses for Software and Hardware

In collaboration with Laboratoire de l'Informatique du Parallélisme (LIP)



Project-Team CASH

Creation of the Project-Team: 2019 June 01

Each year, Inria research teams publish an Activity Report presenting their work and results over the reporting period. These reports follow a common structure, with some optional sections depending on the specific team. They typically begin by outlining the overall objectives and research programme, including the main research themes, goals, and methodological approaches. They also describe the application domains targeted by the team, highlighting the scientific or societal contexts in which their work is situated. The reports then present the highlights of the year, covering major scientific achievements, software developments, or teaching contributions. When relevant, they include sections on software, platforms, and open data, detailing the tools developed and how they are shared. A substantial part is dedicated to new results, where scientific contributions are described in detail, often with subsections specifying participants and associated keywords. Finally, the Activity Report addresses funding, contracts, partnerships, and collaborations at various levels, from industrial agreements to international cooperations. It also covers dissemination and teaching activities, such as participation in scientific events, outreach, and supervision. The document concludes with a presentation of scientific production, including major publications and those produced during the year.

Keywords

Computer sciences and digital sciences

- A1.6. – Green Computing
- A2.1. – Programming Languages
 - A2.1.1. – Semantics of programming languages
 - A2.1.2. – Imperative programming
 - A2.1.4. – Functional programming
 - A2.1.6. – Concurrent programming
 - A2.1.7. – Distributed programming
 - A2.1.9. – Synchronous languages
 - A2.1.10. – Domain-specific languages
 - A2.1.11. – Proof languages
- A2.2. – Compilation
 - A2.2.1. – Static analysis
 - A2.2.2. – Memory models
 - A2.2.3. – Memory management
 - A2.2.4. – Parallel architectures
 - A2.2.5. – Run-time systems
 - A2.2.6. – GPGPU, FPGA...
 - A2.2.8. – Code generation
- A2.3.1. – Embedded systems
- A2.5.1. – Software Architecture & Design
- A2.5.3. – Empirical Software Engineering
- A2.5.4. – Software Maintenance & Evolution
- A4.5. – Formal method for verification, reliability, certification
 - A4.5.1. – Static analysis
 - A4.5.2. – Model-checking
 - A4.5.3. – Program proof
- A7.2. – Logic in Computer Science
 - A7.2.1. – Decision procedures
 - A7.2.3. – Interactive Theorem Proving
 - A7.2.4. – Mechanized Formalization of Mathematics
- A8.3. – Geometry, Topology
- A8.4. – Computer Algebra
- A8.5. – Number theory
- A9.11. – Generative AI

Other research topics and application domains

B3.1. – Sustainable development

B5.4. – Microelectronics

B9.5.1. – Computer science

B9.5.2. – Mathematics

B9.6.10. – Digital humanities

Contents

Project-Team CASH	1
1 Team members, visitors, external collaborators	6
2 Overall objectives	7
3 Research program	8
3.1 Programming Language Design	8
3.2 Semantics and Proofs	8
3.3 Program Analysis and Verification	9
3.4 Optimizations and Program Transformations	9
4 Application domains	9
5 Social and environmental responsibility	9
6 Latest software developments, platforms, open data	10
6.1 Latest software developments	10
6.1.1 DCC	10
6.1.2 PoCo	10
6.1.3 fkcc	11
6.1.4 Vellvm	11
6.1.5 ribbit	11
6.1.6 adtr	12
6.1.7 dowsing	12
6.1.8 odoc	12
6.1.9 PoLA	12
6.1.10 Actors-OCaml	13
6.1.11 ctrees	13
6.1.12 ERCtool	13
6.1.13 Math-Components	14
6.1.14 Math-comp-analysis	14
6.1.15 Hierarchy Builder	14
6.1.16 Trocq	15
7 New results	15
7.1 Research direction 1: Programming Language Design	15
7.1.1 Actors, futures, and algebraic effects	15
7.1.2 SxC: From Data-processing to SIMD	16
7.1.3 Software Ecosystems for Digital Frugality	16
7.2 Research direction 2: Semantics and Proofs	17
7.2.1 Tooling for the Rocq prover: HB and Trocq	17
7.2.2 Formalisation concentration inequalities in the Rocq proof assistant	17
7.2.3 Semantics of Probabilistic Programs Using s-Finite Kernels in Dependent Type Theory	17
7.2.4 Applications of LLM to Formal Verification	17
7.2.5 Deterministic parallel programs	18
7.2.6 Verified Compilation Infrastructure for Concurrent Programs	18
7.2.7 A new module system for OCaml	19
7.3 Research direction 3: Program Analysis and Verification	19
7.3.1 Formal Verification of Electric Properties on Transistor-Level Descriptions of Circuits	19
7.4 Research direction 4: Optimizations and Program Transformations	19
7.4.1 Automatic Specialization of Dense Code on Sparse Structures	19
7.4.2 Code Cartography	20
7.4.3 Towards Optimising Programs with Sketch-Guided Polyhedral Compilation	20

7.4.4	New Insights on Scalar Promotion with the Polyhedral Model	20
7.4.5	Memory optimizations for Algebraic Data Types	20
8	Bilateral contracts and grants with industry	21
8.1	Partnership with the Aniah startup on circuit verification	21
8.2	CAVOC Project with Inria/Nomadic Labs	21
9	Partnerships and cooperations	22
9.1	International initiatives	22
9.1.1	Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program	22
9.1.2	Participation in other International Programs	22
9.2	National initiatives	22
10	Dissemination	23
10.1	Promoting scientific activities	23
10.1.1	Scientific events: organisation	23
10.1.2	Scientific events: selection	23
10.1.3	Journal	23
10.1.4	Invited talks	24
10.1.5	Leadership within the scientific community	24
10.1.6	Scientific expertise	24
10.1.7	Research administration	24
10.2	Teaching - Supervision - Juries - Educational and pedagogical outreach	24
10.3	Popularization	25
10.3.1	Participation in Live events	25
11	Scientific production	25
11.1	Major publications	25
11.2	Publications of the year	26
11.3	Cited publications	28

1 Team members, visitors, external collaborators

Research Scientists

- Christophe Alias [INRIA, Researcher, HDR]
- Cyril Cohen [INRIA, Researcher]
- Ludovic Henrio [CNRS, Researcher, HDR]
- Filippo Nuccio Mortarino Majno Di Capriglio [UNIV Jean Monnet, Senior Researcher, from Sep 2025]
- Gabriel Radanne [INRIA, ISFP]
- Yannick Zakowski [INRIA, Researcher, until Mar 2025]

Faculty Member

- Matthieu Moy [Team leader, UNIV LYON I, Associate Professor, HDR]

Post-Doctoral Fellow

- Theo Stoskopf [INRIA, Post-Doctoral Fellow]

PhD Students

- Gregoire Aubertin [INRIA, from Sep 2025]
- Samy Avrillon [LIP, from Sep 2025]
- Vivien Gachet [LIP, from Sep 2025]
- Vincent Mastain [INRIA, from Sep 2025]
- Emma Nardino [ENS Lyon]
- Oussama Oulkaid [ANIAH, CIFRE, until Sep 2025]
- Etienne Parent [INRIA, from Sep 2025]
- Arthur Pons [Commown, CIFRE, from Feb 2025]
- Alec Sadler [INRIA]

Interns and Apprentices

- Samy Avrillon [ENS Lyon, Intern, from Apr 2025 until Jul 2025]
- Nathan Chandanson [INRIA, Intern, from Mar 2025 until Jul 2025]
- Marius Goyet [INRIA, Intern, from Feb 2025 until Jun 2025]
- Théa Hervier [ENS Lyon, Intern, from Feb 2025 until Jul 2025]
- Loup Paul-Dauphin [ENS Lyon, Intern, from Feb 2025 until Jul 2025]

Administrative Assistant

- Elise Denoyelle [Inria]

External Collaborator

- Laure Gonnord [GRENOBLE INP, HDR]

2 Overall objectives

The overall objective of the CASH team is to design and develop ways to improve the quality of software. We work both on tools that help programmers write better programs, and compilers that turn these programs into efficient executables.

By *improving the quality*, we mean both the safety of programs and the efficiency of their execution. We notably provide programmers with better *programming language* constructs to express their intent: constructions that give them guarantees by-construction such as memory-safety, determinism, etc. When guarantees can't be obtained by construction, we also develop *static analyses* to detect bugs or to prove preconditions needed to apply program transformations. We use such guarantees to develop new *optimizations* to generate efficient code. All these contributions find their foundation and justification in the *semantics* of programs.

When it comes to high level programming constructs for parallelism, we develop a specific expertise in asynchronous computations and ruling out race-conditions in concurrent programs. In this realm, we propose new paradigms, but also contribute to the semantics of such programs. For instance, we design ways to specify the semantics using monadic interpreters, and use them to study the correctness of compilers.

We ensure safety guarantees both through type-systems and analyses, in vastly different contexts: from the verification of electrical circuits to the design of a new module systems for OCaml. As is recurrent in our work, we pragmatically adapt the approach to the practical application at hand.

We design code transformation for the efficient execution of programs, in particular targeting HPC. Our contributions in this realm extend the polyhedral model to make it applicable to a wider range of programs, and to bring its potential for optimisation to new kind of applications (e.g. parametric tiling, sparse structures, etc.). We also design optimisations for structured data such as trees, or more generally algebraic data types.

Our Approach and methodology. We target a balance between theory and practice: problems extracted from industrial requirements often yield theoretical problems.

On the practical side, the CASH team targets applied research, in the sense that most research topics are driven by actual needs, discussed either through industrial partnership or extracted from available benchmarks.

The theoretical aspects ensure the coherency and the correctness of our approach, they are mostly based on formally defined semantics. The formalization of the different representations of the programs and of the analyses allow us to show that these different tasks are performed with the same understanding of the program semantics. Formalization is done either with pen-and-paper definitions and proofs, or through mechanized proofs using the Rocq prover¹.

Our approach is to cross-fertilize between several communities. For example, the abstract interpretation community provides a sound theoretical framework and very powerful analysis, but these are rarely applied in the context of optimizing compilation. Similarly, the formal verification community is very active on software, and RTL hardware verification, but very few researchers applied it to transistor-level verification (lower-level than RTL).

Many of our contributions are “meta-contributions”, in the sense that we do not only provide final results, but also tools to help on the foundational aspects of these results. We provide theoretical and practical tools that are both used internally in the team to achieve our final results, and provided to researchers outside the team. For example, we provide tools to express and manipulate program semantics formally in novel ways instead of focusing only on certified compilers themselves. Also, our contributions to the polyhedral model not only propose new optimizations based on the model, but also general purpose tools to help other researchers to prototype new optimizations. This makes it a bit easier to target general purpose conferences where meta-results are slightly more valued provided they are proven to be applicable but makes our development efforts spread out into several unrelated projects.

¹“The Rocq prover” is the new name of “the Coq proof assistant”, starting from end of 2024. In this document we will refer to Coq as “The Rocq prover”, except in paper names or conference titles which were not (yet) updated at the time of the writing.

3 Research program

The structure of the team is illustrated by Fig. 1 with 4 research directions, detailed below. Vertical shapes represent more fundamental aspects, while horizontal ones represent the applications. Note that three of these directions are an evolution of the first three research focuses in our past activities. The team is recognized and contributes mostly in the four identified axes. Obviously, every direction potentially intersects all the others, but we emphasize the intersections between the fundamental aspects and the applications. For example, in the work on static strong typing programming language constructs are designed to provide analyses with strong guarantees. The purpose of the direction semantics and proofs is to provide tools to certify correctness of optimizations and analyses. This complementarity is manifest in the ANR “Shannon Meet Cray” (sxc.inria.fr) started in 2025 aiming to design new computational models and languages for efficient data-processing that will require cooperation between language design, optimisations and analysis.

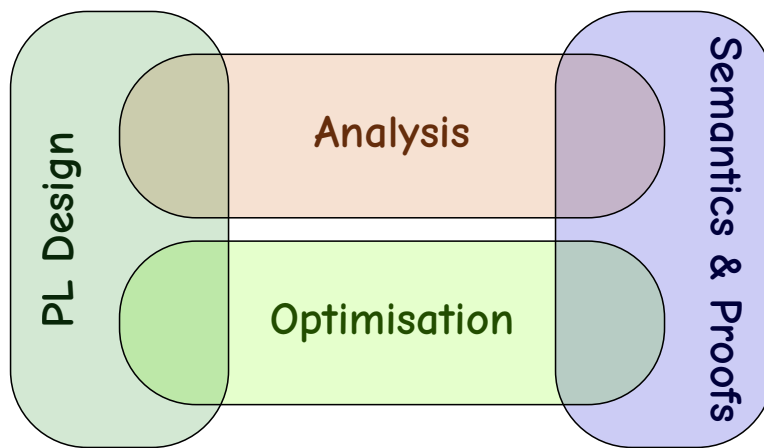


Figure 1: The CASH Research Topics

More importantly than intersection between axes, we highlight that many of the contributions we envision are shared between members of the team, and/or with external collaborators.

3.1 Programming Language Design

While parallelism is one major field of research concerning programming languages, we still have a significant research agenda in sequential programming languages. The interplay between programming languages and static guarantees is still a major concern that drives most of our language designs, in particular our expertise on type systems drives many of our design choices as it is a very effective way to ensure static properties of programs.

Participants Cyril Cohen, Ludovic Henrio, Matthieu Moy, Gabriel Radanne.

3.2 Semantics and Proofs

The team is, overall, interested in formalization and proofs. In this research direction, we push the degree of formalization as far as it can be pushed with machine-checked proofs, typically using the Rock prover (the new name of the “Coq proof assistant”). The following challenges in the organization of formal proofs arise: defining or reusing hierarchies of mathematical structures (adjunctions, monads, (co-)algebras, etc), transferring proofs between structures, or turning abstract algorithms in executable and efficient code.

Participants Cyril Cohen, Ludovic Henrio, Gabriel Radanne.

3.3 Program Analysis and Verification

Our activities on program analysis serve two purposes: large scale verification, intended to find bugs in programs and circuits and help fixing them; and analysis to improve optimizing compilers (e.g. infer invariants that can be used by the optimizer) and language tools (e.g. use the type system to improve the tooling available for developers).

Participants Christophe Alias, Ludovic Henrio, Matthieu Moy, Gabriel Radanne.

3.4 Optimizations and Program Transformations

We have a special focus on *data transformations* both in the context of functional compilation and imperative compilation; while ensuring the *scalability* of our optimizations. These are *transverse* topics that we address through the following several research directions, like memory representation of datatypes, automatic or semi-automatic parallelization, etc.

Participants Christophe Alias, Cyril Cohen, Ludovic Henrio, Gabriel Radanne.

4 Application domains

The scope of targeted applications is wide, from specialized HPC applications on supercomputers to general purpose computing, from massively parallel applications to sequential code, from functional languages to imperative code, etc. Our main focus is software, but we also consider hardware, both as an execution platform for software, and as a research topic (hardware generation and hardware circuit analysis). What links all our activities is that our object of study is *computer programs*.

While a global approach links CASH activities and members, we do not have a single unified toolchain where all contributions would be implemented. Instead we try to attach ourself to existing frameworks. This implies that different activities of CASH target different application domains and potential end-users: OCaml ecosystem; Vellvm project for certified compilation; standalone prototype for memory layout for algebraic data types but this may evolve to a Rust implementation. In other words, we chose for each of our contributions its most relevant toolchain and/or application. This allows us to both use the best technical framework and to pick the best way to eventually reach our final users.

5 Social and environmental responsibility

Footprint of research activities

Although we do not have a precise measure of our carbon (and other environmental) footprint, the two main sources of impact of computer-science research activities are usually travel (plane) and digital equipment (lifecycle of computers and other electronic devices).

Many members of the CASH team are already in an approach of reducing their international travel, and hopefully the new solutions we had to set up to continue our activities during the COVID crisis will allow us to continue our research with a sustainable amount of travel, and using other forms of remote collaborations when possible.

As far as digital equipment is concerned, we try to extend the lifetime of our machines as much as possible.

Impact of research results

Many aspects of our research are meant to provide tools to make better programs, in particular avoid bugs and improve power-efficiency. It is very hard, however, to assess the actual impact of such research. For example, our work on avoiding bugs may either avoid waste of resource due to buggy software or help developers write code for larger, resource-consuming execution platforms. In many cases, improvements in power-efficiency

lead to a rebound effect which may weaken the benefit of the improvement, or even lead to an increase in total consumption (backfire).

CASH provides tools for developers, but does not develop end-user applications. We believe the social impact of our research depends more on the way developers will use our tools than on the way we conduct our research.

Ludovic Henrio followed the “Atelier Sciences Environnements Sociétés Inria 2021” (atelier Sens) organized by Eric Tannier in June 2021. Then, for the voluntary Cash members, he has animated an atelier Sens during the Cash seminar in October 2021.

One line of research explicitly targets energy and environmental transition. We study the possibility of a degrowth in computing power, trying to tackle the issue both from the technical point of view (e.g. our memory-consumption measurements as part of the Fruganum project) and human sciences point of view. We started a CIFRE Ph.D on low-tech infrastructure with the company Commown, and plan to start a Ph.D co-supervised with a sociologist on the evolution of the complexity of programming languages in 2026.

6 Latest software developments, platforms, open data

6.1 Latest software developments

6.1.1 DCC

Name: DPN C Compiler

Keywords: Polyhedral compilation, Automatic parallelization, High-level synthesis

Functional Description: Dcc (Data-aware process network C Compiler) compiles a regular C kernel to a data-aware process network (DPN), a dataflow intermediate representation suitable for high-level synthesis in the context of high-performance computing. Dcc has been registered at the APP (“Agence de protection des programmes”) and transferred to the XtremLogic start-up under an Inria license.

Publication: [hal-03143777](https://hal.archives-ouvertes.fr/hal-03143777)

Contact: Christophe Alias

Participant: 2 anonymous participants

6.1.2 PoCo

Name: Polyhedral Compilation Library

Keywords: Polyhedral compilation, Automatic parallelization

Functional Description: PoCo (Polyhedral Compilation Library) is framework to develop program analysis and optimizations in the polyhedral model. PoCo features polyhedral building blocks as well as state-of-the-art polyhedral program analysis. PoCo has been registered at the APP (“agence de protection des programmes”) and transferred to the XtremLogic start-up under an Inria licence.

News of the Year: This year, PoCo was enhanced with: - a program equivalence procedure - a scalarization/data systolization procedure PoCo kernel was enhanced with: - an interface to ISL affine relations - a generic solver for systems of regular language equations - handling of programs with reductions - generation of systems of affine recurrence equations with reductions

URL: <https://gitlab.inria.fr/alias/poco-light>

Contact: Christophe Alias

Participant: an anonymous participant

6.1.3 fkcc

Name: The Farkas Calculator

Keywords: DSL, Farkas Lemma, Polyhedral compilation

Scientific Description: fkcc is a scripting tool to prototype program analyses and transformations exploiting the affine form of Farkas lemma. Our language is general enough to prototype in a few lines sophisticated termination and scheduling algorithms. The tool is freely available and may be tried online via a web interface. We believe that fkcc is the missing chain to accelerate the development of program analyses and transformations exploiting the affine form of Farkas lemma.

Functional Description: fkcc is a scripting tool to prototype program analyses and transformations exploiting the affine form of Farkas lemma. Our language is general enough to prototype in a few lines sophisticated termination and scheduling algorithms. The tool is freely available and may be tried online via a web interface. We believe that fkcc is the missing chain to accelerate the development of program analyses and transformations exploiting the affine form of Farkas lemma.

Release Contributions: - Script language - Polyhedral constructors - Farkas summation solver

URL: <http://foobar.ens-lyon.fr/fkcc/>

Publication: hal-03106000

Contact: Christophe Alias

Participant: an anonymous participant

6.1.4 Vellvm

Keywords: Coq, Semantic, Compilation, Proof assistant, Proof

Scientific Description: A modern formalization in the Coq proof assistant of the sequential fragment of LLVM IR. The semantics, based on the Interaction Trees library, presents several rare properties for mechanized development of this scale: it is compositional, modular, and extracts to a certified executable interpreter. A rich equational theory of the language is provided, and several verified tools based on this semantics are in development.

Functional Description: Formalization in the Coq proof assistant of a subset of the LLVM compilation infrastructure.

URL: <https://github.com/vellvm/vellvm>

Contact: Yannick Zakowski

Participant: 4 anonymous participants

Partner: University of Pennsylvania

6.1.5 ribbit

Keywords: Compilation, Pattern matching, Algebraic Data Types

Functional Description: Ribbit is a compiler for pattern languages with algebraic data types which is parameterized by the memory representation of types. Given a memory representation, it generates efficient and correct code for pattern matching clauses.

URL: <https://gitlab.inria.fr/cash/ribbit>

Contact: Gabriel Radanne

6.1.6 adtr

Name: ADT Rewriting language

Keywords: Compilation, Static typing, Algebraic Data Types, Term Rewriting Systems

Functional Description: ADTs are generally represented by nested pointers, for each constructors of the algebraic data type. Furthermore, they are generally manipulated persistently, by allocating new constructors.

ADTr allow representing ADTs in a flat way while compiling a pattern match-like construction as a rewrite on the memory representation. The goal is to then use this representation to optimize the rewriting and exploit parallelism.

URL: <https://github.com/Drup/adtr>

Publication: hal-03355377

Contact: Gabriel Radanne

Participant: 3 anonymous participants

6.1.7 dowsing

Keywords: Static typing, Ocaml

Functional Description: Dowsing is a tool to search function by types. Given a simple OCaml type, it will quickly find all functions whose types are compatible.

Dowsing works by building a database containing all the specified libraries. New libraries can be added to the database. It then builds an index which allow to quickly answer to requests.

URL: <https://github.com/Drup/dowsing/>

Publication: hal-03355381

Contact: Gabriel Radanne

Participant: 2 anonymous participants

6.1.8 odoc

Keyword: Ocaml

Functional Description: OCaml is a statically typed programming language with wide-spread use in both academia and industry. Odoc is a tool to generate documentation of OCaml libraries, either as HTML websites for online distribution or to create PDF manuals and man pages.

URL: <https://github.com/ocaml/odoc/>

Contact: Gabriel Radanne

Participant: 4 anonymous participants

6.1.9 PoLA

Name: PoLA: a Polyhedral Liveness Analyser

Keywords: Polyhedral compilation, Array contraction

Functional Description: PoLA is a C++ tool that optimizes the footprint of C(++) programs of the polyhedral model by applying efficient memory mappings deduced from dynamic analysis of the program. More precisely, we apply a fine-grain liveness analysis on execution traces of a program, obtained either by execution or interpretation, and infer parametrized mappings for the arrays used for intermediate computations.

URL: <https://hthieven.gitlabpages.inria.fr/pola/>

Publications: [thievenaz:hal-03862219](#), [thievenaz:hal-03862218](#)

Contact: Christophe Alias

Participant: 3 anonymous participants

Partner: Waseda University

6.1.10 Actors-OCaml

Keywords: Concurrency, Ocaml

Functional Description: An actor library for OCaml

URL: <https://gitlab.inria.fr/mandrieu/actors-ocaml>

Contact: Gabriel Radanne

6.1.11 ctrees

Name: Choice Trees

Keywords: Coq, Concurrency, Formalisation, Semantics, Proof assistant

Functional Description: We develop so-called "ctrees", a data-structure in Coq suitable for modelling and reasoning about non-deterministic programming languages as an executable monadic interpreter. We link this new library to the Interaction Trees project: ctrees offer a valid target for interpretation of non-deterministic events.

URL: <https://github.com/vellvm/ctrees/>

Contact: Yannick Zakowski

6.1.12 ERCtool

Name: Electrical Rule Checking Tool

Keywords: Verification, Model Checking, Electrical circuit, Transistor, Program verification, Formal methods

Functional Description: ERCtool is developed as part of a collaboration with the Aniah company, specialized in the verification of electric properties on circuits. A specificity of ERCtool is that it allows the analysis of a circuit with multiple power-supplies, and allows getting formal guarantees on the absence of error.

Contact: Matthieu Moy

Partner: Aniah

6.1.13 Math-Components

Name: Mathematical Components library

Keywords: Proof assistant, Coq, Formalisation

Functional Description: The Mathematical Components library is a set of Coq libraries that cover the prerequisites for the mechanization of the proof of the Odd Order Theorem.

URL: <https://math-comp.github.io/>

Contact: Assia Mahboubi

Participant: 15 anonymous participants

6.1.14 Math-comp-analysis

Name: Mathematical Components Analysis

Keywords: Proof assistant, Coq, Formalisation

Functional Description: This library adds definitions and theorems to the Math-components library for real numbers and their mathematical structures.

Release Contributions: First major release, several results in topology, integration, and measure theory have been added.

URL: <https://github.com/math-comp/analysis>

Publications: [hal-02463336](#), [hal-03917948](#), [hal-01719918](#)

Contact: Cyril Cohen

Participant: 12 anonymous participants

Partners: Ecole Polytechnique, AIST Tsukuba, Onera

6.1.15 Hierarchy Builder

Keywords: Metaprogramming, Rocq

Scientific Description: It is nowadays customary to organize libraries of machine-checked proofs around hierarchies of algebraic structures. One influential example is the Mathematical Components library, on top of which the long and intricate proof of the Odd Order Theorem could be fully formalized. Still, building algebraic hierarchies in a proof assistant such as Rocq requires a lot of manual labor and often deep expertise in the prover's internals. Moreover, according to our experience, making a hierarchy evolve without breaking client code is equally tricky: even a simple refactoring such as splitting a structure into two simpler ones is hard to get right.

Hierarchy Builder is a high-level language to build hierarchies of algebraic structures and to evolve these hierarchies without breaking user code. The key concepts are factory, builder, and abbreviation, which let the hierarchy developer describe an actual interface for their library. Behind that interface, the developer can provide appropriate code to ensure backward compatibility.

We implement the Hierarchy Builder language in the hierarchy-builder add-on for the Rocq system using the Elpi extension language.

Functional Description: Hierarchy Builder is a high-level language for Rocq to build hierarchies of algebraic structures and to evolve these hierarchies without breaking user code. The key concepts are factory, builder, and abbreviation, which let the hierarchy developer describe an actual interface for their library. Behind that interface, the developer can provide appropriate code to ensure backward compatibility.

Release Contributions: Compatible with Coq 8.18 to Coq 8.20. Added support for instance saturation

URL: <https://github.com/math-comp/hierarchy-builder>

Publication: [hal-02478907](#)

Contact: Enrico Tassi

Participants: Kazuhiko Sakaguchi, Enrico Tassi, Cyril Cohen

Partners: University of Tsukuba, Onera

6.1.16 Trocq

Keywords: Proof synthesis, Proof transfer, Coq, Elpi, Logic programming, Parametricity, Univalence

Functional Description: Trocq is a modular parametricity plugin for Rocq (prototype), aimed at proof transfer. It translates a user goal into a related variant using the target data structures, along with a rich parametricity witness from which a justifying substitution function can be extracted.

The plugin features a hierarchy of parametricity witness types, ranging from structure-less relations to a novel formulation of type equivalence. This gathers several pre-existing parametricity translations (including univalent parametricity and CoqEAL) in a unified framework.

This modular translation performs fine-grained analysis and generates sufficiently rich witnesses to preprocess the goal — without always requiring full type equivalence. It enables proof transfer with the power of univalent parametricity while avoiding the univalence axiom when unnecessary.

The translation is implemented in Rocq-Elpi and features transparent, readable code aligned with a sequent-style theoretical presentation.

Release Contributions: Support for the Prop sort

URL: <https://github.com/coq-community/trocq>

Publication: [hal-04177913](#)

Contact: Cyril Cohen

Participants: Cyril Cohen, Enzo Crance, Assia Mahboubi

Partner: Mitsubishi Electric R&D Centre Europe, France

7 New results

This section presents the scientific results obtained in the evaluation period. They are grouped according to the directions of our research program.

7.1 Research direction 1: Programming Language Design

7.1.1 Actors, futures, and algebraic effects

Participants: Ludovic Henrio, Emma Nardino, Gabriel Radanne, Yannick Zakowski.

This work aims to provide high level language constructors for concurrency and parallelism like actors and futures using modern functional language constructs like algebraic effects. Actors have been implemented and successfully used in several industry-grade frameworks, such as Akka. Algebraic effects allow the precise modelling of operation with effects, while providing excellent composition properties. They have been used both as a fundamental primitive for theoretical study, but also used as effective building blocks to create new complex control and effectful operators. The new version of OCaml with multicore support promotes the use of algebraic effects to implement new concurrency primitives. We implemented actors using

algebraic effects, and obtain a practical, efficient implementation of Actors for OCaml. We also optimize tail asynchronous calls, avoiding the creation of extra futures for representing the results and limiting the number of context-switch between threads.

This year we have continued our efforts on tail-modulo-async calls

- We proved the correction of our optimisation on tail asynchronous calls, and worked on the implementation and its practical evaluation. A research paper is under submission on the subject [25].
- in the context of the PhD thesis of Emma Nardino we also worked on the mechanized formalisation of the transformation, based on the Iris framework. Several technical difficulties and limitations of the framework make the approach difficult, but on the other side, such a proof would increase the expressiveness of the Iris framework itself, which sounds promising.

7.1.2 SxC: From Data-processing to SIMD

Participants: Etienne Parent, Loup Lobet, Gabriel Radanne, Matthieu Moy, Laure Gonnord, Charles Paperman.

The “Shannon meet Cray” (**SxC**) project aims to investigate the use of SIMD instructions, which provide fine-grained parallelism in CPUs, for data-processing such as text search or DNA processing. In particular, this project aims to leverage the theory of circuit complexity (introduced by Shannon) to leverage vectorized instruction (as pioneered in the Cray-1 supercomputer).

This year, we kickstarted the ANR project accepted in 2024 with a workshop in Lille. Etienne Parent started his PhD advised by Charles Paperman, Matthieu Moy and Gabriel Radanne on the design of a DSL for text and DNA processing that compiles to SIMD. Another PhD (Loup Lobet) started simultaneously in Lille advised by Laure Gonnord and Charles Paperman on the compilation aspects.

7.1.3 Software Ecosystems for Digital Frugality

Participants: Matthieu Moy, Guillaume Salagnac, Arthur Pons.

The environmental impact of digital equipment is a concern of growing importance. The carbon footprint of the domain is estimated to 2-4% and most scenario for the next decades anticipate a growth. Manufacturing digital devices requires a wide range of rare elements, that become rarer and rarer as we extract them and that are very hard to recycle. The current trajectory of the digital domain is not sustainable in the long term, and we will probably need to replace the current “infinite growth” paradigm with some form of degrowth in the future. Technological progress on efficiency may be part of the solution, but they are usually cancelled by rebound effect. We claim that a reflection on a potential reduction of the overall computational power is needed. If not well anticipated, a degrowth may be imposed to us rather than chosen. We started working on anticipating such digital degrowth scenarios.

We currently tackle the question of digital ecosystems (programming language, build tool chain, execution environments, editing tools) for frugal systems. Assuming the computational power of computers is reduced in the future, some tools stop being usable by lack of resources. We consider that the main bottleneck would be physical RAM. We developed a tool called mprobe that measures the memory consumption of a program or a set of programs, and a set of scripts to launch this tool on a variety of programs, written in different programming language. We measure both the resource needed to execute the programs, to build it (compile, link, etc.), and also the resource needed to build the tool chain itself. We are working on the analysis of the results. Preliminary results show that even “old” languages like C and C++ require relatively large amounts of RAM to build simple programs, and that interpreted languages like Python may be viable candidates, although their resource consumption in terms of CPU is higher. We also work on integrating mprobe in the Nix package manager, to make it easy to profile the build of any packaged program.

We also started a CIFRE Ph.D with the society **Commown**, that starts an activity of Managed Service Provider. The Ph.D studies the feasibility of low-tech systems to provide services to its customers. We target a long-term sustainable approach with the smallest possible environmental impact.

7.2 Research direction 2: Semantics and Proofs

7.2.1 Tooling for the Rocq prover: HB and Trocq

Participants: Cyril Cohen, Lucie Lahaye, Quentin Vermande, Reynald Affeldt (*AIST, Japan*), Matteo Calosci (*University of Florence*), Marie Kerjean (*LIPN, Paris*), Pierre Roux (*Onera, Toulouse*), Assia Mahboubi (*Inria, Nantes*), Kazuhiko Sakaguchi (*LIP, ENS de Lyon*), Vojtěch Štěpančík (*Inria, Nantes*), Enrico Tassi (*STAMP, Inria, Sophia Antipolis*).

This axis deals with the contribution to the Rocq prover libraries to either automate the structuring (using Hierarchy Builder) and transfer (using Trocq) of properties, or provide general mathematical results (using the mathematical components library).

Trocq is both a new calculus performing a variant of a parametricity translation to achieve transfer of theorems for the calculus of constructions [13], and a prototype plugin for the Rocq prover, which has been ported to the latest version of Rocq by [27].

Hierarchy Builder (HB) is a domain specific language integrated to Rocq and designed to formally describe hierarchies of mathematical structures (e.g. Groups, Rings, Vector Spaces, etc), their various equivalent axiomatisations, their generic theory and their instances. As the impact of this project grows, several axes have emerged. We also extend the Math-Components and math-comp-analysis libraries with several structures (including N-modules, semi-normed spaces, etc).

As part of his PhD, Quentin Vermande, together with Cyril Cohen, worked on extending the capabilities of HB to deal with both bundled and unbundled structures.

Cyril Cohen together with Matteo Calosci and Enrico Tassi worked on extending HB to support changing subject in structure, thus switch from inferring structure on a type to inferring structure on its operations and vice versa.

As part of Vojtěch Štěpančík's PhD, we also worked towards the automated reconstruction of structure from partial descriptions, e.g.: reconstructing the definition of morphisms from the description of objects, the morphism part of a functor from the object part, and naturality properties.

7.2.2 Formalisation concentration inequalities in the Rocq proof assistant

Participants: Cyril Cohen, Reynald Affeldt (*AIST, Japan*), Alessandro Bruni (*ITU Copenhagen*), Pierre Roux (*Onera, Toulouse*), Takafumi Saikawa (*Nagoya University*).

We formalize a proof of concentration inequalities in Rocq, involving a theory of L_p spaces and bounding inequalities in probability theory, contributing to mathcomp analysis in passing.

This led to a publication [17].

7.2.3 Semantics of Probabilistic Programs Using s-Finite Kernels in Dependent Type Theory

Participants: Cyril Cohen, Reynald Affeldt (*AIST, Japan*), Ayumu Saito (*Japan*).

We extend previously published work with more examples and theorems for reasoning on probabilistic programs in the s-finite kernel semantics [11].

7.2.4 Applications of LLM to Formal Verification

Participants: Cyril Cohen, Théo Stoskopf, Guillaume Baudart (*Inria Paris*), Marc Lelarge (*Inria Paris*), Assia Mahboubi (*Inria Nantes*), Nicolas Tabureau (*Inria Nantes*), Jules Viennot (*Inria Paris*).

We explore the use of LLM for proof assistants in the LLM4Code Inria Challenge (« défi Inria »), around essentially three projects:

- Babel Formal [26], a prototype translation tool between CoC based proof assistants, using kernel translations and proof script decompilation,
- LLM4Docq [28], a documentation generating framework for computer formalized mathematical statements,
- **CRRRocq** a chain of thoughts agent with RAG and tool calling to generate proof in interaction with the Rocq prover.

7.2.5 Deterministic parallel programs

Participants: Ludovic Henrio, Yannick Zakowski, Violet Ka I Pun, Einar Broch Johnsen, Asmund Kløvstad.

This research direction is a collaboration between Ludovic Henrio, Yannick Zakowski and our Norwegian colleagues. First results were published in 2021 on a simple static criteria for deterministic behaviour of active objects. We are now extending this work to be able to ensure deterministic behaviour in more cases and to lay a theoretical background that will make our results more general and easier to adapt to different settings.

We have formalized in the Rocq prover a result by DeBruijn dating back from the 70th on proving confluence of a system. In the process, we have solved some mistakes in the existing proof, and generalised it in a way that will make it even more useful in the context of programming language semantics. We applied this theorem to prove the confluence of systems where many threads interact with a set of stateless servers organised as layers. The proof has revealed to be more difficult than expected but we finished it this year and published the results in CPP' 2026.

We plan to extend the application language to futures and more asynchronous behaviour.

7.2.6 Verified Compilation Infrastructure for Concurrent Programs

Participants: Nicolas Chappe, Ludovic Henrio, Yannick Zakowski.

The objective of this research direction is to provide semantic and reasoning tools for the formalization of concurrent programs and the verification of compilers for concurrent languages. In particular, we want to apply these results to the design of verified optimizing compilers for parallel high-level languages. We wish to proceed in the spirit of the approach advocated in Vellvm [33]: compositional, modular, executable monadic interpreters based on Interaction Trees [32] are used to specify the semantics of the language, in contrast with more traditional transition systems. Proving correct optimizations for such concurrent languages naturally requires new proof techniques that we need to design as well. Last year had seen the successful publication of the ctrees project. This year we published the results of the previous year [19]² and investigated the generalisation of the approach to a more expressive notion of (non-deterministic) composition nodes [24].

²An extended version of the POPL paper has been accepted to the JFP journal [23]

7.2.7 A new module system for OCaml

Participants: Clement Blaudeau, Didier Remy, Gabriel Radanne.

ML modules offer large-scale notions of composition and modularity. Provided as an additional layer on top of the core language, they have proven to be both vital to the working OCaml and SML programmers, and inspiring to other use-cases and languages. Unfortunately, their meta-theory remains difficult to comprehend, requiring heavy machinery to prove their soundness.

We study a translation from ML modules to F_ω to provide a new comprehensive description of OCaml modules, embarking on a journey right from the source OCaml module system, up to F_ω , and back. This year, we published a novel model systems that applies directly to OCaml, without using a translation to F_ω but still preserving its expressivity [12].

7.3 Research direction 3: Program Analysis and Verification

7.3.1 Formal Verification of Electric Properties on Transistor-Level Descriptions of Circuits

Participants: Oussama Oulkaid, Bruno Ferres, Ludovic Henrio, Matthieu Moy, Gabriel Radanne, Mehdi Khosravian.

We started discussions with the **Aniah** start-up in 2019, and started a formal partnership in 2022, with the recruitment of Bruno Ferres as a post-doc, and Oussama Oulkaid as a CIFRE Ph.D (co-supervised by Aniah, Verimag, and LIP). We developed a prototype verification tool. The tool compiles transistor-level circuit descriptions (CDL file format) to logical formula expressing the semantics of the circuit plus a property to verify, and uses an SMT solver (Z3) to check the validity of the property. The tool was successfully used on a real-life case study, and we showed that our approach can reduce the number of false-alarms significantly compared to traditional approaches, with a reasonable computational cost (under a second for most sub-circuits analyzed). To the best of our knowledge, formal methods like SAT/SMT-solving were never applied to multi-supplies electronic circuits before. The technique experimented in the prototype was successfully re-implemented in the production tool commercialized by Aniah and is now available in the released version.

In 2025, we finished the validation of a richer semantics able to take into account more quantitative aspects on the circuits under analysis, and applied it to circuit reliability analysis (find the worst-case configuration in terms of circuit aging) and electrical over-stress (check that the voltage difference applied to each transistor is never larger than a maximum allowed value). The quantitative semantics was published in the TCAD journal [16], and the application to worst-case aging was submitted in the same journal.

In parallel with the technical work, we conducted a thorough review of existing work on the domain which was published as a survey in the TODAES journal [15].

The thesis of Oussama Oulkaid was defended on 21st of November 2025.

7.4 Research direction 4: Optimizations and Program Transformations

7.4.1 Automatic Specialization of Dense Code on Sparse Structures

Participants: Alec Sadler, Christophe Alias.

This work is concerned with the *automatic parallelization of sparse code*. Our approach is to *specialize at runtime* a canonical dense kernel on a sparse structure and to extract the runtime kernels to be distributed on the target architecture. This year, we focused on the *specialization* part. We proposed an algorithm able to propagate the sparsity across a dense computation, which may includes *reductions*. Our approach leverages

an *abstraction* using *regular expressions* and particularly *Kleene iterations* to summarize the effect of loop nests. Experimental evaluation shows the precision of our approach.

These results are part of the PhD thesis of Alec Sadler and were presented at IMPACT'25 [20].

7.4.2 Code Cartography

Participants: Alec Sadler, Christophe Alias.

This contribution is the next step of our sparse parallelization framework. Once the dense code have been specialized on a sparse input data, we want to recognize parts which might be implemented with a BLAS sparse kernel. We address this problem with a novel approach, able to *cartography* the code i.e. decide for a parametrized slice of code which algorithm it implements, provided a database of reference algorithms. We propose a preliminary algorithm to solve this problem on general polyhedral programs. Our procedure relies on a parametric program slicing, followed by an template matching method able to compare two parametrized polyhedral programs.

This on-going work is part of the PhD thesis of Alec Sadler. It is still under implementation.

7.4.3 Towards Optimising Programs with Sketch-Guided Polyhedral Compilation

Participants: Valeran Maytie, Reuben Carolan, Christophe Alias, Cedric Bastoul, Thomas Kœhler.

When programmers use semi-automatic compilers, they typically write optimisation scripts, to guide the compiler towards key optimisations. Instead of writing scripts, it may be preferable to write *sketches* that focus on the desired structure of the optimized code, without worrying about individual transformations. In this work, we propose a new semi-automatic, sketch-guided compilation approach. We introduce a sketch language that enables expressing the result of imperative loop transformations and a new polyhedral algorithm capable of generating code constrained by both a sketch and a computation specification.

This is a joint work with the Inria CAMUS team, these results are part of the PhD thesis of Valeran Maytie and will be presented at IMPACT'26.

7.4.4 New Insights on Scalar Promotion with the Polyhedral Model

Participants: Alec Sadler, Nathan Chandanson, Hugo Thievenaz, Christophe Alias.

Memory accesses are a well known bottleneck whose impact might be mitigated by using properly the memory hierarchy until registers. In this paper, we address scalar promotion, a technique to turn temporary arrays into a collection of scalar variables to be allocated to registers. We revisit array scalarization in the light of the recent advances of the polyhedral model. We propose a general algorithm for array scalarization and we show a scalarization of stencil computations thanks to a preliminary preprocessing. Our scalarization algorithm operates on the polyhedral intermediate representation and could be plugged in a polyhedral compiler among other passes. In particular, our scalarization algorithm is parametrized by the program schedule, possibly computed by a previous compilation pass. Experimental results confirm the effectiveness and the efficiency of our approach.

These results will be presented at IMPACT'26.

7.4.5 Memory optimizations for Algebraic Data Types

Participants: Thaïs Baudon, Vivien Gachet, Marius Goyet, Gabriel Radanne, Ludovic Henrio, Laure Gonnord.

Algebraic Data Types (ADT) have emerged as an incredibly effective tool to model and manipulate data in programs. ADTs also provide numerous advantages for optimizing compilers, as the rich declarative description allows choosing the memory representation of the types. Initially found in functional programming languages such as OCaml and Haskell, ADT have found their ways in languages such as Rust, which allows aggressive optimisations of their memory representation.

Ribbit ([2], 6.1.5) is a prototype language and compiler which allow programmers to annotate ADTs with complex memory representation, and compile the rest of the code accordingly.

This year, we redesigned Ribbit’s compilation algorithm to be more general and easier to implement, in preparation for trying to implement it in broader contexts. As part of the internship of Marius Goyet, we designed a new verification algorithm for memory layout, and implemented it in Ribbit. Finally, Vivien Gachet started his PhD supervised by Ludovic Henrio and Gabriel Radanne on the extension of Ribbit to mutable and concurrent data-structures.

8 Bilateral contracts and grants with industry

8.1 Partnership with the Aniah startup on circuit verification

Participants: Bruno Ferres, Matthieu Moy, Ludovic Henrio, Gabriel Radanne, Ousama Oulkaid.

The CASH team started discussion with the [Aniah](#) startup in 2019, to work on verification of electrical properties of circuits at transistor level. We recruited a post-doc (Bruno Ferres) in March 2022, and formalized the collaboration with a bilateral contract (Réf. Inria : 2021-1144), in parallel with a joint internship with LIP, Verimag laboratory and Aniah (Oussama Oulkaid), which led to a CIFRE Ph.D (LIP/Verimag/Aniah) started in October 1st 2022. The collaboration led to the development of a prototype tool, which served as the basis for the re-implementation of the approach in the production tool, and to 4 articles in major conference and journals, plus one ongoing submission.

8.2 CAVOC Project with Inria/Nomadic Labs

Participants: Guilhem Jaber, Gabriel Radanne, Laure Gonnord.

This project aims to develop a *sound and precise static analyzer* for OCaml, that can catch large classes of bugs represented by uncaught exceptions. It will deal with both user-defined exceptions, and built-in ones used to represent *error behaviors*, like the ones triggered by `failwith`, `assert`, or a match failure. Via “assert-failure” detection, it will thus be able to check that invariants annotated by users hold. The analyzer will reason *compositionally* on programs, in order to analyze them at the granularity of a function or of a module. It will be *sound* in a strong way: if an OCaml module is considered to be correct by the analyzer, then one will have the guarantee that no OCaml code interacting with this module can trigger uncaught exceptions coming from the code of this module. In order to be *precise*, it will take into account the abstraction properties provided by the type system and the module system of the language: local values, abstracted definition of types, parametric polymorphism. The goal being that most of the interactions taken into account correspond to typeable OCaml code.

This project is part of the partnership between Inria and Nomadic Labs, and lead by Guilhem Jaber, from the Inria Team Galinette.

9 Partnerships and cooperations

9.1 International initiatives

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

Inria Associate Team FormaSys

Participants: Cohen Cyril, Vermande Quentin.

Title: FormaSys

Partner Institution(s):

- AIST, Japan
- University of Nagoya, Japan
- University of Kyoto, Japan

Date/Duration: 3 years

Additional info/keywords: Formalization of mathematics, Rocq, mathcomp, cyberphysical systems

9.1.2 Participation in other International Programs

ANR PRCI MLOpt

Participants: Christophe Alias, Ali Jannesari, Sid Touati.

Title: MLOpt – Machine Learning for Code Optimization

Partner Institution(s):

- Inria Lyon
- Université Côte d’Azur/CNRS
- Iowa State University

Date/Duration: 3 years

Additional info/keywords: Code Optimization, Machine Learning, Translation Validation, Program Repair

9.2 National initiatives

PEPR NumPex, ExaSoft Project (WP2, Task 2.3)

Participants: Alec Sadler, Christophe Alias, Thierry Gautier, Xavier Rival, Philippe Clauss.

Title: Polysparse – Compiling Sparse Kernels by Specialization

Partner Institution(s): Inria Paris (X. Rival), Inria Nancy (P. Clauss)

Date/Duration: 6 years, started on September 2023.

Additional info/keywords: Compilers, HPC, Polyhedral Model, Sparse Computation

Action exploratoire ProgReco

Participants: Christophe Alias, Sid Touati.

Title: PolyReco – Program Recognition with Machine Learning

Partner Institution(s): Université Côte d’Azur, Inria Lyon

Date/Duration: 4 years

Additional info/keywords: Compilers, Program Analysis, Machine Learning

ANR CoREACT

Participants: Cohen Cyril.

Title: CoREACT

Partner Institution(s): IRIF, école polytechnique, Inria d’Université côte d’Azur

Date/Duration: 4 years

Additional info/keywords: graph rewriting, category theory, Rocq

10 Dissemination

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

Member of the organizing committees

- Cyril Cohen: Steering committee of ITP
- Ludovic Henrio: Steering committee of the ICE workshop

10.1.2 Scientific events: selection

Member of the conference program committees

- Cyril Cohen: FroCoS 2025
- Gabriel Radanne: OCaml Workshop, FProper
- Ludovic Henrio: “Coordination” conference

10.1.3 Journal

Reviewer - reviewing activities All permanent members of the team work as referees for various journals in their respective fields.

10.1.4 Invited talks

- Cyril Cohen gave an invited talk at Chalmers Workshop "Deep-Learning Models for Mathematics and Type Theory", in Göteborg (Sweden),
- Cyril Cohen gave an invited talk at Institut Pascal E-COST Action: EuroProofNet in Orsay,
- Cyril Cohen gave an invited talk at the Malinca ERC synergy grant kick-off meeting at IHP, Paris,
- Cyril Cohen gave an invited seminar at LaBRI in Bordeaux.

10.1.5 Leadership within the scientific community

- Cyril Cohen is
 - 2025 - 2027: co-PI of the Leaning and Rocq'ing project, funded by the [AI initiative AI for Math via a donation of Renaissance Philantropy](#) (together with Guillaume Baudart, Marc Lelarge, Nicolas Tabareau). Total 890kUSD, including 2 budgeted post-doct.
 - 2025 - 2028: french PI of the [FormaSys Associate team](#) (Japan PI: Reynald Affeldt) : 7 Inria members, 5 Japan members, 2 external members, 10800 EUR for travels in 2025,
 - 2025 - 2026: co-PI of Inria Défi Liberabaci (with Martin Bodin), 7 EPI, 1 external partner,
 - [Rocq Zulip server](#) administrator and code of conduct team member,
 - Co-organizer of the monthly Proof Assistant Seminar at ENS de Lyon.

10.1.6 Scientific expertise

- Christophe Alias has been an examiner for the PhD defense of Clément Rossetti at Université de Strasbourg. PhD Advisor: Philippe Clauss, Inria CAMUS.
- Christophe Alias has been an external expert for the PhD evaluation committee (CSI) of Ugo Battiston, Raphaël Colin and Tom Hammer (Université de Strasbourg, Inria CAMUS).
- Cyril Cohen has been an examiner for the PhD defense of Théo Laurent at Université de Montpellier. PhD Advisors: Kenji Maillard and David Delahaye.
- Cyril Cohen has been an examiner for the PhD defense of Cécile Marcon at ISAE, Toulouse. PhD Advisors: Xavier THIRIOUX, Célia PICARD and Cyril Allignol.

10.1.7 Research administration

- Ludovic Henrio: Membre de l'équipe de direction du LIP, correspondant local PIQ, codirection de l'équipe CASH, membre du conseil de laboratoire du LIP, responsable du GT CLAP du GDR GPL.

10.2 Teaching - Supervision - Juries - Educational and pedagogical outreach

Licence

- Christophe Alias: "Compilation", INSA CVL 3A, cours+TD, 27h ETD. Oral examiner for the "second concours de l'ENS de Lyon". Grading and exam designing for X/ENS competitive exam.
- Matthieu Moy: "Réfèrent pédagogique", supervising 100 students/year; "Programmation web", L3 UCBL, 19h; "Projet Informatique", L3 UCBL, 9.5h.
- Etienne Parent: "Algorithmique, Programmation et Structures de données", L2 UCBL, 24h TP ; "Architecture", L2 UCBL, 18h TP.
- Yannick Zakowski: organisation du "Séminaire du département d'informatique" L3 ENS

Master 1

- Christophe Alias: "Optimisation des applications embarquées", INSA CVL 4A, cours+TD, 27h ETD.
- Christophe Alias and Emma Nardino: "Compilation", Préparation à l'agrégation d'informatique, ENS-Lyon, 10h cours + 10h TD.
- Cyril Cohen: "Proofs and Programs" (PP), M1 ENS de Lyon, Master Informatique Fondamentale, 4h CM + 14h TD.
- Matthieu Moy: "Compilation et traduction des programmes", M1 UCBL, responsable, 31h; "Gestion de projet et génie logiciel", M1 UCBL, responsable, 32h; "Projet pour l'Orientation en Master", M1 UCBL, 2 students supervised.
- Gabriel Radanne, Ludovic Henrio and Emma Nardino: "Compilation and Analysis" (CAP), ENS-Lyon, Master d'Informatique Fondamentale, cours, 48h CM + 28h TD, 64h ETD.

10.3 Popularization

10.3.1 Participation in Live events

- Like every year, Matthieu Moy participated in a **Declics** meeting to present the research-related jobs to high-school pupils.
- Christophe Alias has been involved in the organization of "Journée Filles, Maths et Informatique" at ENS de Lyon.

11 Scientific production

11.1 Major publications

- [1] C. Alias and A. Plesco. 'Data-Aware Process Networks'. In: CC 2021 - 30th ACM SIGPLAN International Conference on Compiler Construction. Virtual, South Korea: ACM, 2nd Mar. 2021, pp. 1–11. DOI: [10.1145/3446804.3446847](https://hal.inria.fr/hal-03143777). URL: <https://hal.inria.fr/hal-03143777>.
- [2] T. Baudon, G. Radanne and L. Gonnord. 'Bit-Stealing Made Legal: Compilation for Custom Memory Representations Of Algebraic Data Types'. In: *Proceedings of the ACM on Programming Languages*. ICFP 2023. ICFP. Seattle (USA), United States, 4th Sept. 2023. DOI: [10.1145/3607858](https://inria.hal.science/hal-04165615). URL: <https://inria.hal.science/hal-04165615> (cit. on p. 21).
- [3] C. Blaudeau, D. Rémy and G. Radanne. 'Avoiding Signature Avoidance in ML Modules with Zippers'. In: *Proceedings of the ACM on Programming Languages* POPL.9 (22nd Jan. 2025). DOI: [10.1145/3704902](https://inria.hal.science/hal-04801582). URL: <https://inria.hal.science/hal-04801582>.
- [4] N. Chappe, P. He, L. Henrio, Y. Zakowski and S. Zdancewic. 'Choice Trees: Representing Nondeterministic, Recursive, and Impure Programs in Coq'. In: *Proceedings of the ACM on Programming Languages* (15th Jan. 2023), pp. 1–31. DOI: [10.1145/3571254](https://hal.science/hal-03886910). URL: <https://hal.science/hal-03886910>.
- [5] N. Chappe, L. Henrio, A. Maillé, M. Moy and H. Renaud. 'An Optimised Flow for Futures: From Theory to Practice'. In: *The Art, Science, and Engineering of Programming* 6.1 (15th July 2021), pp. 1–41. DOI: [10.22152/programming-journal.org/2022/6/3](https://hal.inria.fr/hal-03440766). URL: <https://hal.inria.fr/hal-03440766>.
- [6] C. C. Din, R. Hähnle, L. Henrio, E. B. Johnsen, V. K. I. Pun and S. L. T. Tarifa. 'Locally Abstract, Globally Concrete Semantics of Concurrent Programming Languages'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 46.1 (29th Mar. 2024), pp. 1–58. DOI: [10.1145/3648439](https://hal.science/hal-04732946). URL: <https://hal.science/hal-04732946>.
- [7] K. Fernandez-Reyes, D. Clarke, L. Henrio, E. Broch Johnsen and T. Wrigstad. 'Godot: All the Benefits of Implicit and Explicit Futures'. In: ECOOP 2019 - 33rd European Conference on Object-Oriented Programming. Leibniz International Proceedings in Informatics (LIPIcs). London, United Kingdom, 2019, pp. 1–28. URL: <https://hal.archives-ouvertes.fr/hal-02302214>.

- [8] L. Gonnord, L. Henrio, L. Morel and G. Radanne. ‘A Survey on Parallelism and Determinism’. In: *ACM Computing Surveys* (27th Sept. 2022). DOI: [10.1145/3564529](https://doi.org/10.1145/3564529). URL: <https://hal.inria.fr/hal-03828497>.
- [9] R. Hähnle and L. Henrio. ‘Provably Fair Cooperative Scheduling’. In: *The Art, Science, and Engineering of Programming* 8.2 (15th Oct. 2023). DOI: [10.22152/programming-journal.org/2024/8/6](https://doi.org/10.22152/programming-journal.org/2024/8/6). URL: <https://hal.science/hal-04372450>.
- [10] O. Oulkaid, B. Ferres, M. Moy, P. Raymond, M. Khosravian, L. Henrio and G. Radanne. ‘A Transistor Level Relational Semantics for Electrical Rule Checking by SMT Solving’. In: <https://ieeexplore.ieee.org/document/10546537>. Design, Automation and Test in Europe Conference. Valencia, Spain: IEEE, 2024, pp. 1–6. DOI: [10.23919/DATe58400.2024.10546537](https://doi.org/10.23919/DATe58400.2024.10546537). URL: <https://hal.science/hal-04527225>.

11.2 Publications of the year

International journals

- [11] R. Affeldt, C. Cohen and A. Saito. ‘Semantics of Probabilistic Programs Using s-Finite Kernels in Dependent Type Theory’. In: *ACM Transactions on Probabilistic Machine Learning* 1.3 (29th Aug. 2025), pp. 1–34. DOI: [10.1145/3732291](https://doi.org/10.1145/3732291). URL: <https://hal.science/hal-05318682> (cit. on p. 17).
- [12] C. Blaudeau, D. Rémy and G. Radanne. ‘Avoiding Signature Avoidance in ML Modules with Zippers’. In: *Proceedings of the ACM on Programming Languages* POPL.9 (22nd Jan. 2025). DOI: [10.1145/3704902](https://doi.org/10.1145/3704902). URL: <https://inria.hal.science/hal-04801582> (cit. on p. 19).
- [13] C. Cohen, E. Crance and A. Mahboubi. ‘Troq: Proof Transfer for Free, Beyond Equivalence and Univalence’. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* (26th May 2025), pp. 1–40. DOI: [10.1145/3737283](https://doi.org/10.1145/3737283). URL: <https://inria.hal.science/hal-05192017> (cit. on p. 17).
- [14] C. Cohen and K. Sakaguchi. ‘A bargain for mergesorts — How to prove your mergesort correct and stable, almost for free’. In: *Proceedings of the ACM on Programming Languages* 9.ICFP (5th Aug. 2025), pp. 1–29. DOI: [10.1145/3747505](https://doi.org/10.1145/3747505). URL: <https://hal.science/hal-05111866>.
- [15] B. Ferres, O. Oulkaid, M. Moy, G. Radanne, L. Henrio, P. Raymond and M. Khosravian. ‘A Survey on Transistor-Level Electrical Rule Checking of Integrated Circuits’. In: *ACM Transactions on Design Automation of Electronic Systems* (12th July 2025). DOI: [10.1145/3748327](https://doi.org/10.1145/3748327). URL: <https://hal.science/hal-05185119> (cit. on p. 19).
- [16] O. Oulkaid, B. Ferres, M. Moy, P. Raymond and M. Khosravian. ‘Modeling Techniques for the Formal Verification of Integrated Circuits at Transistor-Level: Performance vs. Precision Trade-offs’. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (10th Sept. 2025). DOI: [10.1109/TCAD.2025.3608647](https://doi.org/10.1109/TCAD.2025.3608647). URL: <https://hal.science/hal-05290989> (cit. on p. 19).

International peer-reviewed conferences

- [17] R. Affeldt, A. Bruni, C. Cohen, P. Roux and T. Saikawa. ‘Formalizing Concentration Inequalities in Rocq: Infrastructure and Automation’. In: 16th International Conference on Interactive Theorem Proving (ITP 2025). Vol. 16th International Conference on Interactive Theorem Proving (ITP 2025). Reykjavik, Iceland: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 22nd Sept. 2025. DOI: [10.4230/LIPIcs.ITP.2025.21](https://doi.org/10.4230/LIPIcs.ITP.2025.21). URL: <https://hal.science/hal-05318685> (cit. on p. 17).
- [18] P. Borthelle, T. Hirschowitz, G. Jaber and Y. Zakowski. ‘An abstract, certified account of operational game semantics’. In: European Symposium on Programming. Vol. 15694. Hamilton, Ontario, Canada: Springer, 2025, pp. 172–199. DOI: [10.1007/978-3-031-91118-7_7](https://doi.org/10.1007/978-3-031-91118-7_7). URL: <https://hal.science/hal-04583895>.

- [19] N. Chappe, L. Henrio and Y. Zakowski. ‘Monadic Interpreters for Concurrent Memory Models: Executable Semantics of a Concurrent Subset of LLVM IR’. In: *Proceedings of the 14th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP ’25: 14th ACM SIGPLAN International Conference on Certified Programs and Proofs. Denver CO USA, France: ACM, 13th Jan. 2025, pp. 283–298. DOI: [10.1145/3703595.3705890](https://doi.org/10.1145/3703595.3705890). URL: <https://hal.science/hal-04594073> (cit. on p. 18).
- [20] A. Sadler and C. Alias. ‘Automatic Specialization of Polyhedral Programs on Sparse Structures’. In: *Online proceedings*. 15th International Workshop on Polyhedral Compilation Techniques (IMPACT’25). Barcelona (ES), Spain, 22nd Jan. 2025. URL: <https://inria.hal.science/hal-05005245> (cit. on p. 20).
- [21] Q. Vermande. ‘Optimizing Canonical Structures’. In: *UNIF 2025 - Informal Proceedings of the 39th International Workshop on Unification*. UNIF 2025 - 39th International Workshop on Unification. Birmingham, United Kingdom, 14th July 2025. URL: <https://inria.hal.science/hal-05148851>.

Doctoral dissertations and habilitation theses

- [22] O. Oulkaid. ‘Formal Models of Integrated Circuits for Transistor Level Electrical Verification’. Université Claude Beranrd Lyon 1, 21st Nov. 2025. URL: <https://hal.science/tel-05414117>.

Reports & preprints

- [23] N. Chappe, P. He, L. Henrio, E. Ioannidis, Y. Zakowski and S. Zdancewic. *Choice Trees: Representing and Reasoning About Nondeterministic, Recursive, and Impure Programs in Rocq*. 9th July 2025. URL: <https://hal.science/hal-05154458> (cit. on p. 18).
- [24] T. Hervier. *MTrees : Mixing monadic effects and Labelled Transition Systems in Rocq*. ENS de Lyon, 26th June 2025. URL: <https://hal.science/hal-05140642> (cit. on p. 18).
- [25] E. Nardino, L. Henrio, G. Radanne and Y. Zakowski. *Tail Modulo Async-Await*. 2025. URL: <https://hal.science/hal-05006570> (cit. on p. 16).
- [26] T. Stoskopf, C. Cohen and N. Tabareau. *Babel-formal: Translation of Proofs between Lean and Rocq*. 2025. URL: <https://hal.science/hal-05342510> (cit. on p. 18).

Other scientific publications

- [27] L. Lahaye. ‘Troq: a deeper study of the lattice of relations’. LIP : Laboratoire de l’Informatique du Parallélisme, ENS Lyon, 10th July 2025. URL: <https://inria.hal.science/hal-05342516> (cit. on p. 17).
- [28] T. Stoskopf, J. Vienne and C. Cohen. *LLM4Docq: Bootstrapping Documentation for MathComp with LLMs and Expert Feedback*. 27th Sept. 2025. URL: <https://inria.hal.science/hal-05342426> (cit. on p. 18).

Software

- [29] [SW] R. Affeldt, Y. Bertot, A. Bruni, C. Cohen, M. Kerjean, A. Mahboubi, D. Rouhling, P. Roux, K. Sakaguchi, Z. Stone, P.-Y. Strub and L. Théry, *Mathematical Components Analysis* 18th Apr. 2025. LIC: CeCILL-C Free Software License Agreement. HAL: [hal-05038721](https://hal.science/hal-05038721), URL: <https://inria.hal.science/hal-05038721>, SWHID: [swh:1:dir:5fe4b61acc38fea6066881d466ac64524e90a4cd](https://zenodo.org/record/10492403).
- [30] [SW] C. Cohen, E. Crance and A. Mahboubi, *Troq* 18th Apr. 2025. LIC: GNU Lesser General Public License v3.0 or later. DOI: [10.5281/zenodo.10492403](https://doi.org/10.5281/zenodo.10492403), HAL: [hal-05038716](https://hal.science/hal-05038716), URL: <https://inria.hal.science/hal-05038716>, SWHID: [swh:1:dir:21f2e18a8e117e0615eb19d5554ad4a5f828f6b1](https://zenodo.org/record/19d5554ad4a5f828f6b1).

- [31] [SW] C. Cohen, P. Roux, K. Sakaguchi and E. Tassi, *Hierarchy Builder* 18th Apr. 2025. LIC: MIT License. HAL: [hal-05038713](https://hal.archives-ouvertes.fr/hal-05038713), URL: <https://inria.hal.science/hal-05038713>, SWHID: [swh:1:dir:50c5087fc42a2f95913dada99ef4ac1cdba66d7b](https://sw.hal.science/swh1:dir:50c5087fc42a2f95913dada99ef4ac1cdba66d7b)).

11.3 Cited publications

- [32] L. Xia, Y. Zakowski, P. He, C. Hur, G. Malecha, B. C. Pierce and S. Zdancewic. ‘Interaction Trees’. In: *Proceedings of the ACM on Programming Languages* 4.POPL (2020). DOI: [10.1145/3371119](https://doi.org/10.1145/3371119) (cit. on p. 18).
- [33] Y. Zakowski, C. Beck, I. Yoon, I. Zaichuk, V. Zaliva and S. Zdancewic. ‘Modular, compositional, and executable formal semantics for LLVM IR’. In: *Proceedings of the ACM on Programming Languages* 5.ICFP (Aug. 2021), pp. 1–30. DOI: [10.1145/3473572](https://doi.org/10.1145/3473572). URL: <https://hal.science/hal-03525711> (cit. on p. 18).