RESEARCH CENTRE

**Inria Saclay Center at Institut Polytechnique de Paris**

**IN PARTNERSHIP WITH:**

**CNRS, Institut Polytechnique de Paris**

2022

# ACTIVITY REPORT

## Project-Team

# PARTOUT

## Proof Automation and RepresenTation: a fOundation of compUtation and deducTion

**IN COLLABORATION WITH: Laboratoire d'informatique de l'école polytechnique (LIX)**

**DOMAIN**

**Algorithmics, Programming, Software and Architecture**

**THEME**

**Proofs and Verification**

Inria

# Contents

# Project-Team PARTOUT

*Creation of the Project-Team: 2019 December 01*

## Keywords

**Computer sciences and digital sciences**

A2.1. – Programming Languages

A2.2. – Compilation

A2.4. – Formal method for verification, reliability, certification

A4.5. – Formal methods for security

A7.2. – Logic in Computer Science

A7.2.1. – Decision procedures

A7.2.2. – Automated Theorem Proving

A7.2.3. – Interactive Theorem Proving

A7.2.4. – Mechanized Formalization of Mathematics

A7.3.1. – Computational models and calculability

A8.1. – Discrete mathematics, combinatorics

A8.11. – Game Theory

**Other research topics and application domains**

B6.1. – Software industry

# 1 Team members, visitors, external collaborators

## Research Scientists

- Lutz Strassburger [Team leader, INRIA, Senior Researcher, HDR]
- Beniamino Accattoli [INRIA, Researcher]
- Kaustuv Chaudhuri [INRIA, Researcher]
- Ian Mackie [CNRS, Researcher]
- Dale Miller [INRIA, Senior Researcher]
- Gabriel Scherer [INRIA, Researcher]

## Faculty Members

- Benjamin Werner [LIX, Professor, HDR]
- Noam Zeilberger [LIX, Associate Professor]

## Post-Doctoral Fellows

- Bryce Clarke [Inria, from Mar 2022]
- Peter Faul [Ecole Polytechnique, from Oct 2022]
- Lê Thành Dung Nguyên [Ecole Polytechnique, from Mar 2022 until Sep 2022]

## PhD Students

- Farah Al Wardani [Inria]
- Nicolas Blanco [University of Birmingham]
- Pablo Donato [LIX]
- Adrienne Lancelot [Inria, from Oct 2022]
- Olivier Martinot [INRIA]
- Marianela Evelyn Morales Elena [IP Paris]
- Giti Omidvar [INRIA]
- Wendlasida Ouedraogo [SIEMENS MOBILITY]
- Antoine Sere [LIX]
- Alexandros Singh [Université Paris Nord, until Nov 2022]
- Jui-Hsuan Wu [IP PARIS]

## Interns and Apprentices

- Luigi Massacci [Ecole polytechnique, Intern, until Sep 2022]
- Remy Sasseau [Inria, Intern, until Mar 2022, Bachelor Student]

## Administrative Assistant

- Michael Barbosa [Inria, from May 2022]

### Visiting Scientist

- Victoria Barrett [University of Bath, from Nov 2022 until Nov 2022]

## 2 Overall objectives

There is an emerging consensus that formal methods must be used as a matter of course in software development. Most software is too complex to be fully understood by one programmer or even a team of programmers, and requires the help of computerized techniques such as testing and model checking to analyze and eliminate entire classes of bugs. Moreover, in order for the software to be maintainable and reusable, it not only needs to be bug-free but also needs to have fully specified behavior, ideally accompanied with formal and machine-checkable proofs of correctness with respect to the specification. Indeed, formal specification and machine verification is the only way to achieve the highest level of assurance (EAL7) according to the ISO/IEC Common Criteria.[1]

Historically, achieving such a high degree of certainty in the operation of software has required significant investment of manpower, and hence of money. As a consequence, only software that is of critical importance (and relatively unchanging), such as monitoring software for nuclear reactors or fly-by-wire controllers in airplanes, has been subjected to such intense scrutiny. However, we are entering an age where we need trustworthy software in more mundane situations, with rapid development cycles, and without huge costs. For example: modern cars are essentially mobile computing platforms, smart-devices manage our intensely personal details, elections (and election campaigns) are increasingly fully computerized, and networks of drones monitor air pollution, traffic, military arenas, etc. Bugs in such systems can certainly lead to unpleasant, dangerous, or even life-threatening incidents.

The field of formal methods has stepped up to meet this growing need for trustworthy general purpose software in recent decades. Techniques such as computational type systems and explicit program annotations/contracts, and tools such as model checkers and interactive theorem provers, are starting to become standard in the computing industry. Indeed, many of these tools and techniques are now a part of undergraduate computer science curricula. In order to be usable by ordinary programmers (without PhDs in logic), such tools and techniques have to be high level and rely heavily on automation. Furthermore, multiple tools and techniques often need to marshaled to achieve a verification task, so theorem provers, solvers, model checkers, property testers, etc. need to be able to communicate with—and, ideally, trust—each other.

With all this sophistication in formal tools, there is an obvious question: what should we trust? Sophisticated formal reasoning tools are, generally speaking, complex software artifacts themselves; if we want complex software to undergo rigorous formal analysis we must be prepared to formally analyze the tools and techniques used in formal reasoning itself. Historically, the issue of trust has been addressed by means of relativizing it to *small* and *simple* cores. This is the basis of industrially successful formal reasoning systems such as Coq, Isabelle, HOL4, and ACL2. However, the relativization of trust has led to a balkanization of the formal reasoning community, since the Coq kernel, for example, is incompatible with the Isabelle kernel, and neither can directly cross-validate formal developments built with the other. Thus, there is now a burgeoning cottage industry of translations and adaptations of different formal proof languages for bridging the gap. A number of proposals have also been made for universal or retargetable proof languages (e.g., Dedukti, ProofCert) so that the cross-platform trust issues can be factorized into single trusted checkers.

Beyond mutual incompatibility caused by relativized trust, there is a bigger problem that the proof evidence that is accepted by small kernels is generally far too detailed to be useful. Formal developments usually occurs at a much higher level, relying on algorithmic techniques such as unification, simplification, rewriting, and controlled proof search to fill in details. Indeed, the most reusable products of formal developments tend to be these algorithmic techniques and associated collections of hand-crafted rules. Unfortunately, these techniques are even less portable than the fully detailed proofs themselves, since the techniques are often implemented in terms of the behaviors of the trusted kernels. We can broadly say that the problem with relativized trust is that it is based on the *operational* interpretation of implementations of trusted kernels. There still remains the question of *meta-theoretic correctness*. Most formal reasoning

[1] http://www.commoncriteriaportal.org/cc/

systems implement a variant of a well known mathematical formalism (e.g., Martin-Löf type theory, set theory, higher-order logic), but it is surprising that hardly any mainstream system has a formalized meta-theory.[2] Furthermore, formal reasoning systems are usually associated with complicated checkers for side-conditions that often have unclear mathematical status. For example, the Coq kernel has a built-in syntactic termination checker for recursive fixed-point expressions that is required to work correctly for the kernel to be sound. This termination checker evolves and improves with each version of Coq, and therefore the most accurate documentation of its behavior is its own source code. Coq is not special in this regard: similar trusted features exist in nearly every mainstream formal reasoning system.

The PARTOUT project is interested in the principles of deductive and computational formalisms. In the broadest sense, we are interested in the question of *trustworthy and verifiable meta-theory*. At one end, this includes the well studied foundational questions of the meta-theory of logical systems and type systems: cut-elimination and focusing in proof theory, type soundness and normalization theorems in type theory, etc. The focus of our research here is on the fundamental relationships behind the the notions of *computation* and *deduction*. We are particularly interested in relationships that go beyond the well known correspondences between proofs and programs.[3] Indeed, interpreting *computation in terms of deduction* (as in logic programming) or *deduction in terms of computation* (as in rewrite systems or in model checking) can often lead to fruitful and enlightening research questions, both theoretical and practical.

From another end, PARTOUT works on the question of the *essential nature* of deductive or computational formalisms. For instance, we are interested in the question of *proof identity* that attempts to answer the following question: when are two proofs of the same theorem the same? Surprisingly, this very basic question is left unanswered in *proof theory*, the branch of mathematics that supposedly treats proofs as algebraic objects of interest. We also pay particular attention to the combinatorial and complexity-theoretic properties of the formalisms. Indeed, it is surprising that until very recently the $\lambda$-calculus, which is the de facto basis of every functional programming language, lacked a good complexity-theoretic foundation, i.e., a cost model that would allow us to use the $\lambda$-calculus directly to define complexity classes.

To put trustworthy meta-theory to use, the PARTOUT project also works on the design and implementations of formal reasoning tools and techniques. We study the mathematical principles behind the representations of formal concepts ($\lambda$-terms, proofs, abstract machines, etc.), with the goal of identifying the relationships and trade-offs. We also study computational formalisms such as higher-order relational programming that is well suited to the specification and analysis of systems defined in the *structural operational semantics* (SOS) style. We also work on foundational questions about induction and co-induction, which are used in intricate combinations in metamathematics.

# 3 Research program

Software and hardware systems perform *computation* (systems that process, compute and perform) and *deduction* (systems that search, check or prove). The makers of those systems express their intent using various frameworks such as programming languages, specification languages, and logics. The PARTOUT project aims at developing and using mathematical principles to design better frameworks for computation and reasoning. Principles of expression are researched from two directions, in tandem:

- Foundational approaches, from theories to applications: studying fundamental problems of programming and proof theory.

  Examples include studying the complexity of reduction strategies in lambda-calculi with sharing, or studying proof representations that quotient over rule permutations and can be adapted to many different logics.

- Empirical approaches, from applications to theories: studying systems currently in use to build a theoretical understanding of the practical choices made by their designers.

[2] A prominent exception is HOL-Light, whose implementation has been self-certified—in HOL-Light itself—up to a strong assumption necessary to side-step incompleteness.

[3] The *Curry-Howard* correspondence.

Examples include studying realistic implementations of programming languages and proof assistants, which differ in interesting ways from their usual high-level formal description (regarding of sharing of code and data, for example), or studying new approaches to efficient automated proof search, relating them to existing approaches of proof theory, for example to design proof certificates or to generalize them to non-classical logics.

One of the strengths of PARTOUT is the co-existence of a number of different expertise and points of view. Many dichotomies exist in the study of computation and deduction: functional programming *vs* logic programming, operational semantics *vs* denotational semantics, constructive logic *vs* classical logic, proof terms *vs* proof nets, etc. We do not identify with any one of them in particular, rather with them as a whole, believing in the value of interaction and cross-fertilization between different approaches. PARTOUT defines its scope through the following core tenets:

- An interest in both computation and logic.
- The use of mathematical formalism as our core scientific method, paired with practical implementations of the systems we study.
- A shared belief in the importance of good *design* when creating new means of expression, iterating towards simplicity and elegance.

More concretely, the research in PARTOUT will be centered around the following four themes:

1. **Foundations of proof theory as a theory of proofs.** Current proof theory is not a theory of proofs but a theory of proof systems. This has many practical consequences, as a proof produced by modern theorem provers cannot be considered independent from the tool that produced it. A central research topic here is the quest for proof representations that are independent from the proof system, so that proof theory becomes a proper theory of proofs.
2. **Program Equivalence** We intend to use our proof theoretical insights to deepen our understanding of the structure of computer programs by discovering canonical representations for functional programming languages, and to apply these to the problems of program equivalence checking and program synthesis.
3. **Reasoning with relational specifications of formal systems.** Formal systems play a central role for proof checkers and proof assistants that are used for software verification. But there is usually a large gap between the specification of those formal systems in concise informal mathematical language and their implementation in ML or C code. Our research goal is to close that gap.
4. **Foundations of complexity analysis for functional programs.** One of the great merits of the functional programming paradigm is the natural availability of high-level abstractions. However, these abstractions jeopardize the programmer's predictive control on the performance of the code, since many low-level steps are abstracted away by higher-order functions. Our research goal is to regain that control by developing models of space and time costs for functional programs.

# 4 Application domains

## 4.1 Automated Theorem Proving

The Partout team studies the structure of mathematical proofs, in ways that often makes them more amenable to automated theorem proving – automatically searching the space of proof candidates for a statement to find an actual proof – or a counter-example.

(Due to fundamental computability limits, fully-automatic proving is only possible for simple statements, but this field has been making a lot of progress in recent years, and is in particular interested with the idea of generating verifiable evidence for the proofs that are found, which fits squarely within the expertise of Partout.)

### 4.2 Proof-assistants

Our work on the structure of proofs also suggests ways how they could be presented to a user, edited and maintained, in particular in “proof assistants”, automated tool to assist the writing of mathematical proofs with automatic checking of their correctness.

### 4.3 Programming language design

Our work also gives insight on the structure and properties of programming languages. We can improve the design or implementation of programming languages, help programmers or language implementors reason about the correctness of the programs in a given language, or reason about the cost of execution of a program.

## 5 Highlights of the year

Dale Miller was named an ACM Fellow for contributions to proof theory and computational logic.

Dale Miller was named a Fellow of the Asia-Pacific Artificial Intelligence Association (AAIA).

Accattoli has been co-chair of the international conference PPDP 2022.

The version 5.0 of the OCaml programming language implementation was released, with participation from Gabriel Scherer.

### 5.1 Awards

Paper [9] by Accattoli, Dal Lago, and Vanoni received the *distinguished paper award* of the international conference ICFP 2022.

### 5.2 Results

- We solved a longstanding open problem by showing that the logic BV and pomset logic are not the same (see [22] and [34]).

- Accattoli has two papers [15, 14] in the international conference LICS 2022, one of the top conferences of the area, and both papers have been selected for the special issue of the conference. Moreover, [15] solved a long-standing problem in the theory of $\lambda$-calculus, namely how to measure logarithmic space in a way equivalent to Turing machines.

### 5.3 Grants

- Zeilberger is the scientific coordinator of LambdaComb, a four-year project financed by the ANR from the beginning of 2022. Broadly, the project aims to deepen connections between lambda calculus and logic on the one hand and combinatorics on the other. One important motivation for the project is the discovery over recent years of a host of surprising links between subsystems of lambda calculus and enumeration of graphs on surfaces, or "maps", the latter being an active subfield of combinatorics with roots in W. T. Tutte's work in the 1960s. Currently, the project involves over 20 researchers distributed across four partner laboratories in France (LIX, LIPN, LIS, and LIGM) and one partner in Poland (Jagiellonian University).

- Accattoli is the PI of the Inria Action Exploratoire *CANofGAS* (Cost ANalyses of GAme Semantics), led together with Guilhem Jaber (Université de Nantes & Inria Rennes, Gallinette team), which started in 2022 and will continue until 2025. The aim of the project is to merge two lines of research, the one about reasonable cost models for the $\lambda$-calculus and the one about game semantics for functional languages.

# 6 New software and platforms

## 6.1 New software

### 6.1.1 MOIN

**Name:** MOdal Intuitionistic Nested sequents

**Keywords:** Logic programming, Modal logic

**Functional Description:** MOIN is a SWI Prolog theorem prover for classical and intuitionstic modal logics. The modal and intuitionistic modal logics considered are all the 15 systems occurring in the modal S5-cube, and all the decidable intuitionistic modal logics in the IS5-cube. MOIN also provides a protptype implementation for the intuitionistic logics for which decidability is not known (IK4,ID5 and IS4). MOIN is consists of a set of Prolog clauses, each clause representing a rule in one of the three proof systems. The clauses are recursively applied to a given formula, constructing a proof-search tree. The user selects the nested proof system, the logic, and the formula to be tested. In the case of classic nested sequent and Maehara-style nested sequents, MOIN yields a derivation, in case of success of the proof search, or a countermodel, in case of proof search failure. The countermodel for classical modal logics is a Kripke model, while for intuitionistic modal logic is a bi-relational model. In case of Gentzen-style nested sequents, the prover does not perform a countermodel extraction.

A system description of MOIN is available at https://hal.inria.fr/hal-02457240

**URL:** http://www.lix.polytechnique.fr/Labo/Lutz.Strassburger/Software/Moin/MoinProver.html

**Publication:** hal-02457240

**Contact:** Lutz Strassburger

### 6.1.2 OCaml

**Keywords:** Functional programming, Static typing, Compilation

**Functional Description:** The OCaml language is a functional programming language that combines safety with expressiveness through the use of a precise and flexible type system with automatic type inference. The OCaml system is a comprehensive implementation of this language, featuring two compilers (a bytecode compiler, for fast prototyping and interactive use, and a native-code compiler producing efficient machine code for x86, ARM, PowerPC, RISC-V and System Z), a debugger, and a documentation generator. Many other tools and libraries are contributed by the user community and organized around the OPAM package manager.

**URL:** https://ocaml.org/

**Publications:** hal-03146495, hal-03510931, hal-03145030, hal-01929508, hal-03125031, hal-00772993, hal-00914493, hal-00914560, inria-00074804, hal-01499973, hal-01499946

**Contact:** Damien Doligez

**Participants:** Florian Angeletti, Damien Doligez, Xavier Leroy, Luc Maranget, Gabriel Scherer, Alain Frisch, Jacques Garrigue, Marc Shinwell, Jeremy Yallop, Leo White

### 6.1.3 Abella

**Keyword:** Proof assistant

**Functional Description:** Abella is an interactive theorem prover for reasoning about computations given as relational specifications. Abella is particuarly well suited for reasoning about binding constructs.

**URL:** http://abella-prover.org/

**Contact:** Kaustuv Chaudhuri

**Participants:** Dale Miller, Gopalan Nadathur, Kaustuv Chaudhuri, Mary Southern, Matteo Cimini, Olivier Savary-Bélanger, Yuting Wang

**Partner:** Department of Computer Science and Engineering, University of Minnesota

#### 6.1.4 ocaml-boxroot

**Keywords:** Interoperability, Library, Ocaml, Rust

**Scientific Description:** Boxroot is an implementation of roots for the OCaml GC based on concurrent allocation techniques. These roots are designed to support a calling convention to interface between Rust and OCaml code that reconciles the latter's foreign function interface with the idioms from the former.

**Functional Description:** Boxroot implements fast movable roots for OCaml in C. A root is a data type which contains an OCaml value, and interfaces with the OCaml GC to ensure that this value and its transitive children are kept alive while the root exists. This can be used to write programs in other languages that interface with programs written in OCaml.

**URL:** https://gitlab.com/ocaml-rust/ocaml-boxroot

**Publication:** hal-03910313

**Contact:** Guillaume Munch

**Participants:** Guillaume Munch, Gabriel Scherer

#### 6.1.5 Actema

**Name:** Actema

**Keywords:** Higher-order logic, First-order logic, Proof assistant, GUI (Graphical User Interface), Man-machine interfaces, User Interfaces

**Functional Description:** This is a new approach, aiming at making the building of formal proofs more intuitive and convenient. The system is currently at a prototype stage. An interfacing with the Coq proof-system is under study. The system runs through an html/JS serve.

**Release Contributions:** This version can be used online at actema.xyz and comes with explanation videos.

**News of the Year:** The logical mechanism at work in Actema have been described in the article https://hal.archives-ouvertes.fr/hal-03823357

**URL:** http://actema.xyz

**Publication:** 03823357

**Contact:** Benjamin Werner

**Participants:** Benjamin Werner, Pablo Donato, Pierre-Yves Strub

**Partner:** Ecole Polytechnique

#### 6.1.6 Profound-Intuitionistic

**Name:** Interactive theorem proving by direct manipulation for Intuitionistic Logic

**Keywords:** Interactive Theorem Proving, First-order logic

**Functional Description:** Profount-Intuitionistic (Profint) is a tool for building formal proofs in intuitionistic logic using an interactive direct manipulation based web-interface. The tool can transform the interactive proof into formal proof objects in a variety of backend provers including: Coq, Lean 3, Lean 4, Isabelle/HOL, HOL4, and Abella.

**Release Contributions:** This release adds support for proof output in a variety of backend provers including: Coq, Lean 3, Lean 4, Isabelle/HOL, HOL 4, and Abella.

**URL:** https://github.com/direct-manipulation/profint

**Contact:** Kaustuv Chaudhuri

## 7 New results

### 7.1 Combinatorial Proofs for Constructive Modal Logic

**Participants:** Lutz Straßburger.

**External Collaborators:** Matteo Acclavio (Univ. Luxembourg)

Combinatorial proofs form a syntax-independent presentation of proofs, originally proposed by Hughes for classical propositional logic. In this paper we present a notion of combinatorial proofs for the constructive modal logics CK and CD, we show soundness and completeness of combinatorial proofs by translation from and to sequent calculus proofs, and we discuss the notion of proof equivalence enforced by these translations.

This work has been published at the AiML 2022 conference [18]

### 7.2 Normalization Without Syntax

**Participants:** Lutz Straßburger.

**External Collaborators:** Willem Heijltjes (University of Bath), Dominic Hughes (UC Berkeley)

We present normalization for intuitionistic combinatorial proofs (ICPs) and relate it to the simply-typed lambda-calculus. We prove confluence and strong normalization. Combinatorial proofs, or "proofs without syntax", form a graphical semantics of proof in various logics that is canonical yet complexity-aware: they are a polynomial-sized representation of sequent proofs that factors out exactly the non-duplicating permutations. Our approach to normalization aligns with these characteristics: it is canonical (free of permutations) and generic (readily applied to other logics). Our reduction mechanism is a canonical representation of reduction in sequent calculus with closed cuts (no abstraction is allowed below a cut), and relates to closed reduction in lambda-calculus and supercombinators. While we will use ICPs concretely, the notion of reduction is completely abstract, and can be specialized to give a reduction mechanism for any representation of typed normal forms.

This work was published at the FSCD 2022 conference [20].

### 7.3 Combinatorial Flows as Bicolored Atomic Flows

**Participants:** Giti Omidvar, Lutz Straßburger.

Combinatorial flows are a graphical representation of proofs. They can be seen as a generalization of atomic flows on one side and of combinatorial proofs on the other side. From atomic flows, introduced by Guglielemi and Gundersen, they inherit the close correspondence with open deduction and the possibility of tracing the occurrences of atoms in a derivation. From combinatorial proofs, introduced by Hughes, they inherit the correctness criterion that allows to reconstruct the derivation from the flow. In fact, combinatorial flows form a proof system in the sense of Cook and Reckhow. We show how to translate between open deduction derivations and combinatorial flows, and we show how they are related to combinatorial proofs with cuts.

This work has been published in [23]

### 7.4 BV and Pomset Logic Are Not the Same

**Participants:** Lutz Straßburger.

**External Collaborators:** Nguyễn, Lê Thành Dũng (ENS Lyon)

BV and pomset logic are two logics that both conservatively extend unit-free multiplicative linear logic by a third binary connective, which (i) is non-commutative, (ii) is self-dual, and (iii) lies between the "par" and the "tensor". It was conjectured early on (more than 20 years ago), that these two logics, that share the same language, that both admit cut elimination, and whose connectives have essentially the same properties, are in fact the same. In this paper we show that this is not the case. We present a formula that is provable in pomset logic but not in BV.

We also studied the complexity of the two logics. These results are presented in [22] and [34].

### 7.5 Coqlex, an approach to generate verified lexers

**Participants:** Lutz Straßburger, Wendlasida Ouedraogo, Gabriel Scherer.

**External Collaborators:** Danko Ilik (Siemens)

A compiler consists of a sequence of phases going from lexical analysis to code generation. Ideally, the formal verification of a compiler should include the formal verification of every component of the tool-chain. In order to contribute to the end-to-end verification of compilers, we implemented a verified lexer generator with usage similar to OCamllex. This software-Coqlex-reads a lexer specification and generates a lexer equipped with Coq proofs of its correctness. Although the performance of the generated lexers does not measure up to the performance of a standard lexer generator such as OCamllex, the safety guarantees it comes with make it an interesting alternative to use when implementing totally verified compilers or other language processing tools.

More details on this work can be found here [35]

### 7.6 An Analytic Propositional Proof System On Graphs

**Participants:** Lutz Straßburger.

**External Collaborators:** Matteo Acclavio (Univ. Luxembourg), Ross Horne (Univ. Luxembourg)

In this work, published in [11] we present a proof system that operates on graphs instead of formulas. Starting from the well-known relationship between formulas and cographs, we drop the cograph-conditions and look at arbitrary (undirected) graphs. This means that we lose the tree structure of the formulas corresponding to the cographs, and we can no longer use standard proof theoretical methods that depend on that tree structure. In order to overcome this difficulty, we use a modular decomposition of graphs and some techniques from deep inference where inference rules do not rely on the main connective of a formula. For our proof system we show the admissibility of cut and a generalization of the splitting property. Finally, we show that our system is a conservative extension of multiplicative linear logic with mix, and we argue that our graphs form a notion of generalized connective.

## 7.7 A Graphical Proof Theory of Logical Time

**Participants:** Lutz Straßburger.

**External Collaborators:** Matteo Acclavio (Univ. Luxembourg), Ross Horne (Univ. Luxembourg), Sjouke Mauw (Univ. Luxembourg)

Logical time is a partial order over events in distributed systems, constraining which events precede others. Special interest has been given to series-parallel orders since they correspond to formulas constructed via the two operations for "series" and "parallel" composition. For this reason, seriesparallel orders have received attention from proof theory, leading to pomset logic, the logic BV, and their extensions. However, logical time does not always form a series-parallel order; indeed, ubiquitous structures in distributed systems are beyond current proof theoretic methods. In this paper, we explore how this restriction can be lifted. We design new logics that work directly on graphs instead of formulas, we develop their proof theory, and we show that our logics are conservative extensions of the logic BV.

This work was published at the FSCD 2022 conference [17].

## 7.8 Taming Bounded Depth with Nested Sequents

**Participants:** Lutz Straßburger.

**External Collaborators:** Agata Ciabattoni (TU Wien), Matteo Tesi (SNS Pisa)

Bounded depth refers to a property of Kripke frames that serve as semantics for intuitionistic logic. We introduce nested sequent calculi for the intermediate logics of bounded depth. Our calculi are obtained in a modular way by adding suitable structural rules to a variant of Fitting's calculus for intuitionistic propositional logic, for which we present the first syntactic cut elimination proof. This proof modularly extends to the new nested sequent calculi introduced in this paper, which has been published at the the AiML 2022 conference [19]

## 7.9 From axioms to synthetic inference rules via focusing

**Participants:** Dale Miller.

**External Collaborators:** Sonia Marin (University of Birmingham), Elaine Pimentel (University College of London), and Marco Volpe (Osnabrueck University).

We examine the synthetic inference rules that arise when using theories composed of bipolars in both classical and intuitionistic logics. A key step in transforming a formula into synthetic inference rules involves attaching a polarity to atomic formulas and some logical connectives. Since there are different choices in how polarity is assigned, it is possible to produce different synthetic inference rules

for the same formula. We show that this flexibility allows for the generalization of different approaches for transforming axioms into sequent rules present in the literature. We also show how to apply these results to organize the proof theory of labeled sequent systems for several propositional modal logics. This work was published in the Annals of Pure and Applied Logic [13].

## 7.10 Computational logic based on linear logic and fixed points

**Participants:** Matteo Manighetti, Dale Miller.

We use $\mu$MALL, the logic that results from adding least and greatest fixed points to first-order multiplicative-additive linear logic, as a framework for presenting several topics in computational logic. In particular, we present various levels of restrictions on the roles of fixed points in proofs and show that these levels capture different topics. For example, level 0 of $\mu$MALL captures (generalized) unification problems, level 1 captures Horn-clause logic programming, level 2 captures various model checking problems, and level 3 introduces a linearized form of arithmetic. We also show how the proof search interpretation of $\mu$MALL can be used to compute general recursive functions. Finally, we identify several situations where provability in Peano Arithmetic can be replaced by provability in $\mu$MALL. In such situations, the proof theory of $\mu$MALL can be used to study and implement proof search procedures for fragments of Peano Arithmetic.

This work was presented at TLLA-Linearity 2022 and appears in [33].

## 7.11 Distributing and trusting proof checking

**Participants:** Kaustuv Chaudhuri, Dale Miller, Farah Al Wardani.

When a proof-checking kernel completes the checking of a formal proof, that kernel asserts that a specific formula follows from a collection of lemmas within a given logic. We describe a framework in which such an assertion can be made globally so that any other proof assistant willing to trust that kernel can use that assertion without rechecking (or even understanding) the formal proof associated with that assertion. In this framework, we propose to move beyond autarkic proof checkers-i.e., self-sufficient provers that trust proofs only when they are checked by their kernel-to an explicitly non-autarkic setting. This framework must, of course, explicitly track which agents (proof checkers and their operators) are being trusted when a trusting proof checker makes its assertions. We describe how we have integrated this framework into a particular theorem prover while making minor changes to how the prover inputs and outputs text files. This framework has been implemented using off-the-shelf web-based technologies, such as JSON, IPFS, IPLD, and public key cryptography. A preliminary report on this work appears in [31].

## 7.12 A positive perspective on term representation

**Participants:** Dale Miller, Jui-Hsuan Wu.

The focused proof system LJF can be used as a framework for describing term structures and substitution. Since the proof theory of LJF does not pick a canonical polarization for primitive types, two different approaches to term representation arise. When primitive types are given the negative polarity, LJF proofs encode terms as tree-like structures in a familiar fashion. In this situation, cut elimination also yields the familiar notion of substitution. On the other hand, when primitive types are given the positive polarity, LJF proofs yield a structure in which explicit sharing of term structures is possible. Such a representation of terms provides an explicit method for sharing term structures. In this setting, cut elimination yields a different notion of substitution. We illustrate these two approaches to term representation by applying

them to the encoding of untyped $\lambda$-terms. We also exploit concurrency theory techniques—namely traces and simulation—to compare untyped $\lambda$-terms using such different structuring disciplines.

This work will be presented as an invited paper at CSL 2023 [21].

## 7.13 Graphical User Interfaces for Formal Proof Construction

**Participants:** Kaustuv Chaudhuri, Pablo Donato, Benjamin Werner.

Deep Inference can ways to construct proofs steps by pointing to two different locations in the goal and/or hypotheses. This had been described in [36]. We have built on this to provide a novel proof interface, Actema, which to construct proofs without textual commands, in a way which is, we beleive, intuitive and user-friendly. A version of the system can be used on the system's web page.

This work was published at the CPP 2022 conference [27].

A new version of Actema is under way which allows to use it as a front-end for the Coq proof system.

## 7.14 Certified Cryptography

**Participants:** Antoine Séré.

Antoine Séré is conducting his PhD research, with Pierre-Yves Strub as external main adviser, on formally certified cryptography. This year he participated to the formal correctness proof of the Kyber crpytographic primitive (which won the NIST post-quantum competition). An article is to be submitted.

## 7.15 Useful Open Call-by-Need

**Participants:** Beniamino Accattoli.

**External Collaborators:** Maico Leberle (ex PhD student in PARTOUT).

This work studies *useful sharing*, which is a sophisticated optimization for $\lambda$-calculi, in the context of call-by-need evaluation (an efficient evaluation strategy) in presence of open terms (a feature needed in the implementation of proof assistants such as Coq). Useful sharing turns out to be harder in call-by-need than for other strategies (such as call-by-name or call-by-value), because call-by-need evaluates inside so-called environments, making it harder to specify when a substitution step is useful. We isolate the key involved concepts and prove the correctness and the completeness of useful sharing in this setting.

This work belongs to the *foundations of complexity analysis for functional programs* theme of the research program of PARTOUT. It has been published in [16].

## 7.16 Reasonable Space for the $\lambda$-Calculus, Logarithmically

**Participants:** Beniamino Accattoli.

**External Collaborators:** Ugo Dal Lago (University of Bologna & Inria), Gabriele Vanoni (University of Bologna & Inria).

Can the $\lambda$-calculus be considered a reasonable computational model? Can we use it for measuring the time and space consumption of algorithms? While the literature contains positive answers about time, much less is known about space.

This work presents a new reasonable space cost model for the $\lambda$-calculus, based on a variant over the Krivine abstract machine, that we call Space KAM. For the first time, this cost model is able to accommodate logarithmic space, thus our result is the answer to a long-standing open problem in the theory of $\lambda$-calculus. Moreover, we study the time behavior of our machine and show how to transport our results to the call-by-value $\lambda$-calculus.

This work belongs to the *foundations of complexity analysis for functional programs* theme of the research program of PARTOUT. It has been published in [15], and it has been selected for the special issue of the conference.

## 7.17 Multi Types and Reasonable Space

**Participants:** Beniamino Accattoli.

**External Collaborators:** Ugo Dal Lago (University of Bologna & Inria), Gabriele Vanoni (University of Bologna & Inria).

This work continues the study of the previous sub-section, providing a new system of multi types (a variant of intersection types) and we show how to extract from multi type derivations the space used by the Space KAM, capturing into a type system the space complexity of the abstract machine. Additionally, we show how to capture also the time of the Space KAM, which is a reasonable time cost model, via minor changes to the type system.

This work belongs to the *foundations of complexity analysis for functional programs* theme of the research program of PARTOUT. It has been published in [9], and it received the *distinguished paper award* of the conference.

## 7.18 Exponentials as Substitutions and the Cost of Cut Elimination in Linear Logic

**Participants:** Beniamino Accattoli.

This work introduces the exponential substitution calculus (ESC), a new presentation of cut elimination for intuitionistic multiplicative and exponential linear logic (IMELL), based on proof terms and building on the idea that exponentials can be seen as explicit substitutions (a formalism for representing sharing in the $\lambda$-calculus). The idea in itself is not new, but here it is pushed to a new level, inspired by Accattoli and Kesner's linear substitution calculus (LSC).

One of the key properties of the LSC is that it naturally models the sub-term property of abstract machines, which is the key ingre- dient for the study of reasonable time cost models for the $\lambda$-calculus. The new ESC is then used to design a cut elimination strategy with the sub-term property, providing the first polynomial cost model for cut elimination with unconstrained exponentials.

For the ESC, we also prove untyped confluence and typed strong normalization, showing that it is an alternative to proof nets for an advanced study of cut elimination.

This work belongs to the *foundations of complexity analysis for functional programs* theme of the research program of PARTOUT. It has been published in [14], and it has been selected for the special issue of the conference.

## 7.19 The Theory of Call-by-Value Solvability

**Participants:** Beniamino Accattoli.

**External Collaborators:** Giulio Guerrieri(Edinbugh Research Centre, Huawei, UK).

The denotational semantics of the untyped $\lambda$-calculus is a well developed field built around the concept of *solvable terms*, which are elegantly characterized in many different ways. In particular, unsolvable terms provide a consistent notion of meaningless term. The semantics of the untyped call-by-value $\lambda$-calculus (CbV), the variant of the $\lambda$-calculus used to model most applications, is instead still in its infancy, because of some inherent difficulties but also because CbV solvable terms are less studied and understood than in call-by-name. On the one hand, we show that a carefully crafted presentation of CbV allows us to recover many of the properties that solvability has in call-by-name, in particular qualitative and quantitative characterizations via multi types. On the other hand, we stress that, in CbV, solvability plays a different role: identifying unsolvable terms as meaningless induces an inconsistent theory.

This work belongs to the *foundations of complexity analysis for functional programs* theme of the research program of PARTOUT. It has been published in [10].

## 7.20 Parsing as a lifting problem and the Chomsky-Schützenberger Representation Theorem

**Participants:** Noam Zeilberger.

We begin by explaining how any context-free grammar encodes a functor of operads from a freely generated operad into a certain "operad of spliced words". This motivates a more general notion of CFG over any category $C$, defined as a finite species $S$ equipped with a color denoting the start symbol and a functor of operads $p : Free[S] \to W[C]$ into the operad of spliced arrows in $C$. We show that many standard properties of CFGs can be formulated within this framework, and that usual closure properties of CF languages generalize to CF languages of arrows. We also discuss a dual fibrational perspective on the functor $p$ via the notion of "displayed" operad, corresponding to a lax functor of operads $W[C] \to Span(Set)$.

We then turn to the Chomsky-Schützenberger Representation Theorem. We describe how a non-deterministic finite state automaton can be seen as a category $Q$ equipped with a pair of objects denoting initial and accepting states and a functor of categories $Q \to C$ satisfying the unique lifting of factorizations property and the finite fiber property. Then, we explain how to extend this notion of automaton to functors of operads, which generalize tree automata, allowing us to lift an automaton over a category to an automaton over its operad of spliced arrows. We show that every CFG over a category can be pulled back along a ND finite state automaton over the same category, and hence that CF languages are closed under intersection with regular languages. The last important ingredient is the identification of a left adjoint $C[-] : Operad \to Cat$ to the operad of spliced arrows functor, building the "contour category" of an operad. Using this, we generalize the C-S representation theorem, proving that any context-free language of arrows over a category $C$ is the functorial image of the intersection of a $C$-chromatic tree contour language and a regular language.

This work has been published in [28].

**External Collaborators:** Paul-André Melliès (CNRS)

## 7.21 An introduction to enriched cofunctors

**Participants:** Bryce Clarke.

Cofunctors are a kind of map between categories which lift morphisms along an object assignment. In this paper, we introduce cofunctors between categories enriched in a distributive monoidal category. We define a double category of enriched categories, enriched functors, and enriched cofunctors, whose horizontal and vertical 2-categories have 2-cells given by enriched natural transformations between functors and cofunctors, respectively. Enriched lenses are defined as a compatible enriched functor and enriched cofunctor pair; weighted lenses, which were introduced by Perrone, are precisely lenses enriched in weighted sets. Several other examples are also studied in detail.

A preliminary version of this article was released as the preprint [32]. A revised version is in preparation, for journal submission.

**External Collaborators:** Matthew di Meglio

## 7.22 Debootstrapping without archeology

**Participants:** Gabriel Scherer.

**External Collaborators:** Nathanëlle Courant (INRIA Paris), Julien Lepiller (Yale University)

It is common for programming languages that their reference implementation is implemented in the language itself. This requires a "bootstrap binary": the executable form of a previous version of the implementation is provided along with the sources, to be able to run the implementation itself. Those bootstrap binaries are opaque; they could contain bugs, or even malicious changes that could reproduce themselves when running the source version of the language implementation-this is called the "trusting trust attack". A collective project called Bootstrappable was launched in to remove bootstrap binaries, providing alternative build paths that do not rely on opaque binaries.

Camlboot is our project to debootstrap the OCaml compiler, version 4.07. Using diverse double-compilation, we were able to prove the absence of trusting trust attack in the existing bootstrap binaries of the standard OCaml implementation.

To our knowledge, our publication [12] is the first scholarly discussion of "tailored" debootstrapping for high-level programming languages. Debootstrapping recently grew an active community of free software contributors, but so far the interactions with the programming language research community have been minimal. We share our experience on Camlboot, trying to highlight aspects that are of interest to other language designers and implementors; we hope to foster stronger ties between the Bootstrappable project and relevant academic communities. In particular, the debootstrapping experience has been an interesting reflection on language design and implementation..

## 7.23 Déboîter les constructeurs

**Participants:** Gabriel Scherer.

**External Collaborators:** Nicolas Chataing (INRIA Paris), Camille Noûs (laboratoire Cogitamus)

Nous proposons dans [24] une implémentation d'une nouvelle fonctionnalité pour OCaml, l'unboxing de constructeur. Elle permet d'élimineer certains constructeurs de la représentation dynamique des valeurs quand cela ne crée pas d'ambiguité entre différentes valeurs au même type. Nous décrivons:

- l'analyse statique nécessaire pour accepter ou rejeter l'unboxing d'un constructeur,
- l'impact sur la compilation du filtrage de motif, et
- un cas d'usage préliminaire sur les grands entiers où la fonctionnalité améliore les performances de code OCaml idiomatique, éliminant le besoin d'écrire du code non-sûr.

Pour notre analyse statique, nous devons normaliser certaines expressions de type, avec une relation de normalisation qui ne termine pas nécessairement en présence de types mutuellement récursifs; nous décrivons une analyse de terminaison qui garantit la normalisation sans rejeter les déclarations de types qui nous intéressent.

## 7.24 Backtracking reference stores

**Participants:** Gabriel Scherer.

**External Collaborators:** Camille Noûs (laboratoire Cogitamus)

François Pottier's `union-find` library is parameterized over an underlying store of mutable references, and provides the usual references, transactional reference stores (for rolling back some changes in case of higher-level errors), and persistent reference stores. In [26], we extend this library with a new implementation of backtracking reference stores, to get a Union-Find implementation that efficiently supports arbitrary backtracking and also subsumes the transactional interface.

Our backtracking reference stores are not specific to union-find, they can be used to build arbitrary backtracking data structures. The natural implementation, using a journal to record all writes, provides amortized-constant-time operations with a space overhead linear in the number of store updates. A refined implementation reduces the memory overhead to be linear in the number of store cells updated, and gives performance that match non-backtracking references in practice.

### 7.25 Boxroot, fast movable GC roots for a better FFI

**Participants:** Gabriel Scherer.

**External Collaborators:** Guillaume Munch-Maccagnoni (INRIA Rennes/Nantes)

In [29] we study the safe manipulation of GC-managed values inside non-managed (foreign) code. Focusing on the problem of implementing a safe and convenient FFI for OCaml in Rust, we propose a new interface and implementation for storing roots for the OCaml GC inside foreign (C and Rust) data structures, along with a typing discipline in Rust's ownership type system, which offer:

- better performance than existing root-registration interfaces;
- efficient support for multicore OCaml, thanks to a multicore-friendly design;
- a reasoning based on resource-management idioms, enabling an easier OCaml FFI for Rust.

### 7.26 A OCaml use-case for strong call-by-need reduction

**Participants:** Gabriel Scherer.

**External Collaborators:** Nathanëlle Courant (INRIA Paris)

In [30] we detail a use-case for strong call-by-need reduction in the OCaml compiler. Strong call-by-need reduction is a sophisticated reduction strategy for programming languages, and to our knowledge all its practical applications known today are in the field of proof assistants. Our use-case is the first sighting of a use for strong call-by-need reduction outside this specific domain.

## 8 Bilateral contracts and grants with industry

### 8.1 Bilateral contracts with industry

#### 8.1.1 CIFRE Thesis Inria - Siemens

**Participants:** Lutz Straßburger, Wendlasida Ouedraogo.

**Title:** Optimization of source code for safety-critical systems

**Duration:** 2020 – 2022

**Scientific Responsible:** Lutz Straßburger

**Industrial Partner:** Siemens Mobility, Chatillon

**Summary:** The goal of the thesis is to develop ways to optimize the performance of software, while not sacrificing the guarantees of safety already provided for non-optimized code. The software that Siemens is using for their self-driving trains (e.g. Metro 14 in Paris) is programmed in Ada. Due to the high safety requirements for the software, the used Ada compiler has to be certified. At the current state of the art, only non-optimized code fulfils all necessary requirements. Because of higher performance needs, we are interested in producing optimized code that also fulfils these reqirements.

Stated most generally, the aim of the thesis is to assure, *at the same time*:

- optimization of execution-time of safety-critical software — safety-critical software is more prone to bad execution-time performance, because most of its actions involve performing checks (i.e., CPU branch instructions), and
- maintaining the safety guarantees from the input source code to the produced binary code — in general, as soon as we decide to use a compiler optimization, the qualification of the compiler no longer applies.

## 8.2 Bilateral grants with industry

### 8.2.1 OCaml Software Foundation

**Participants:** Gabriel Scherer.

The OCaml Software Foundation (OCSF),[4] established in 2018 under the umbrella of the Inria Foundation, aims to promote, protect, and advance the OCaml programming language and its ecosystem, and to support and facilitate the growth of a diverse and international community of OCaml users.

Since 2019, Gabriel Scherer serves as the director of the foundation.

### 8.2.2 General OCaml funding from Nomadic Labs

**Participants:** Gabriel Scherer, Olivier Martinot.

Nomadic Labs, a Paris-based company, has implemented the Tezos blockchain and cryptocurrency entirely in OCaml. In 2019, Nomadic Labs and Inria have signed a framework agreement (“contrat-cadre”) that allows Nomadic Labs to fund multiple research efforts carried out by Inria groups. Within this framework, we participate to the following grants, in collaboration with the project-team Cambium at INRIA Paris:

#### Évolution d’OCaml

This grant is intended to fund a number of improvements to OCaml, including the addition of new features and a possible re-design of the OCaml type-checker. This grant funds the PhD thesis of Olivier Martinot on this topic.

[4] http://ocaml-sf.org/

**Maintenance d'OCaml**

This grant is intended to fund the day-to-day maintenance of OCaml as well as the considerable work involved in managing the release cycle.

**OCaml-Rust**

**Title:** OCaml/Rust bindings

**Duration:** 2021-2023

**Coordinator:** Gabriel Scherer (INRIA Saclay, EPI Partout)

**Participants:** Guillaume Munch-Maccagnoni (INRIA Rennes, EPI Galinette), Jacques-Henri Jourdan (CNRS, LRI)

**Partners:** Inria, Nomadic Labs

**Inria contact:** Gabriel Scherer

**Summary:** We often want to write hybrid programs with components in several different programming languages. Interfacing two languages typically goes through low-level, unsafe interfaces. The OCaml/Rust project studies safer interfaces between OCaml and Rust.

**Expected Impact:** We investigated safe low-level representations of OCaml values on the Rust side, representing GC ownership, and developed a calling convention that reconciles the OCaml FFI idioms with Rust idioms. We also developed Boxroot, a new API to register values with the OCaml GC, for used when interfacing with Rust (and other programming languages) and possibly when writing concurrent programs. This resulted in novel techniques which can benefit other pairs of languages in the future. These works are now integrated in the ocaml-rs interface between OCaml and Rust used in the industry.

# 9 Partnerships and cooperations

## 9.1 International initiatives

### 9.1.1 Inria associate team not involved in an IIL or an international program

**COMPRONOM**

**Title:** Combinatorial Proof Normalization

**Duration:** 2020 ->

**Coordinator:** Lutz Strassburger, Willem Heijltjes

**Partners:**

- Inria Saclay (France)
- Université de Bath (Royaume-Uni)
- University College London (Royaume-Uni)

**Inria contact:** Lutz Strassburger

**Summary:** This project teams up three research groups, one at Inria Saclay, one at the University of Bath, and one at University College London, who are driven by their joint interest in the development of a combinatorial proof theory which is able to treat formal proofs independently from syntactic proof systems. We plan to focus our research in two major directions: First, study the normalization of combinatorial proofs, with possible applications for the implementation of functional programming languages, and second, study combinatorial proofs for the logic of bunched implications, with the possible application for separation logic and its use in the verification of imperative programs.

#### 9.1.2 STIC/MATH/CLIMAT AmSud projects

**DyLo-MPC**

**Title:** Dynamic Logics: Model Theory, Proof Theory and Computational Complexity

**Duration:** 2020–2022

**Coordinator:** Carlos Eduardo Areces

**Partners:**

- Facultad de Matemática, Astronomía, Física y Computación, Universidad Institution Nacional de Córdoba, Argentina [UNC-AR]
- Departamento de Ciência da Computação, Universidade Federal do Rio de Janeiro, Brazil [UFRJ-BR]
- Equipe Partout, Laboratoire d'Informatique de l'Ecole Polytechnique and Inria Saclay, France [LIX-FR]
- Laboratoire Spécification et Vérification, ENS Paris-Saclay and CNRS, France [LSV-FR]

**Inria contact:** Lutz Strassburger

**Summary:** During the project we will advance our understanding of a novel family of modal logics called dynamic logics. Dynamic logics are characterized by the inclusion of modal operators that can modify the model in which they are being evaluated. This characteristic made them especially well suited for the description of evolving scenarios like, for example, the temporal evolution of a communication network, where connections are dynamically created and eliminated, constantly changing the actual topology. A number of different dynamic logics have been investigated by members of the project, but a general perspective is still missing, and a number of important open questions remains, ranging from adequate model theoretic characterizations, to a proper understanding of how to define proof calculi for this logics, which are usually not closed under uniform substitutions. The project aims to pull together the strengths of the four international research teams, to unify existing results and attempt to answer these open problems .

### 9.2 International research visitors

#### 9.2.1 Visits to international teams

**Research stays abroad**

**Beniamino Accattoli**

**Visited institution:** Huawei Research Center, Edinburgh;

**Country:** UK;

**Dates:** 1-31 July 2022;

**Context of the visit:** collaboration with Dan Ghica on the complexity analyses of program transformations at work in compilers for functional languages;

**Mobility program/type of mobility:** research stay.

## 9.3 National initiatives

**LambdaComb**

**Title:** LambdaComb: a cartographic quest between lambda-calculus, logic, and combinatorics

**Duration:** 2022 – 2026 (4 years)

**Coordinator:** Noam Zeilberger

**Partners:**

- LIX (Ecole Polytechnique), LIPN (Paris Nord), LIS (Marseille), LIGM (Marne-la-Vallée)
- Jagiellonian University (Poland)

**Summary:** LambdaComb is an interdisciplinary project financed by the Agence Nationale de la Recherche (PRC grant ANR-21-CE48-0017). Broadly, the project aims to deepen connections between lambda calculus and logic on the one hand and combinatorics on the other. One important motivation for the project is the discovery over recent years of a host of surprising links between subsystems of lambda calculus and enumeration of graphs on surfaces, or "maps", the latter being an active subfield of combinatorics with roots in W. T. Tutte's work in the 1960s. Using these new links and other ideas and tools, the LambdaComb project aims to:

- develop rigorous logical perspectives on maps and related combinatorial objects; and
- develop precise quantitative perspectives on lambda calculus and related systems.

The project also intersects with and aims to shed new light on other established connections between logic and geometry, notably Joyal and Street's categorical framework of string diagrams as well as Girard's proof nets for linear logic.

**REPRO**

**Title:** REPRO: searching for canonical REpresentations of PROgrams.

**Duration:** 2021 – 2025 (4 years)

**Coordinator:** Gabriel Scherer

**Summary:** The REPRO project aims to

1. deepen our understanding of the structure of computer programs by discovering canonical representations for fundamental programming languages, and to
2. explore the application of canonical representations to the problems of program equivalence checking and program synthesis.

**CoREACT**

**Title:** CoREACT: Coq-based Rewriting: towards Executable Applied Category Theory

**Duration:** 2023 – 2027 (4 years)

**Coordinator:** Nicolas Behr

**Partners:** IRIF (Université Paris Cité), LIP (ENS-Lyon), LIX (Ecole Polytechnique), Sophia-Antipolis (Inria)

**Local participants:** Benjamin Werner, Noam Zeilberger

**Summary:** The main objectives of the CoREACT project include:

1. Development of a methodology for diagrammatic reasoning in Coq

2. Formalization and certification of a representative collection of axioms and theorems for compositional categorical rewriting theory
3. Development of a Coq-enabled interactive database and wiki system
4. Development of a CoREACT wiki-based "proof-by-pointing" engine
5. Executable reference prototype algorithms from categorical structures in Coq (via the use of SMT solvers/theorem provers such as Z3)

# 10 Dissemination

## 10.1 Promoting scientific activities

### 10.1.1 Scientific events: organisation

**Member of the organizing committees**

- Miller is a member of the Steering Committees of LICS, LFMTP, CPP, FLOPS.
- Accattoli is a member of the Steering Committee of PPDP.
- Zeilberger is a member of the Steering Committee of CLA (Computational Logic and Applications).
- Zeilberger organized the kick-off meeting for the ANR LambdaComb, as a hybrid event at Ecole Polytechnique on April 11, 2022.
- Chaudhuri is a member of the steering committee of the International Joint Conference on Automated Reasoning (IJCAR)

**Participating in the organization**

- Marianela Morales was student volunteer for the ETAPS 2022 conference

### 10.1.2 Scientific events: selection

**Member of the conference program committees**

- Miller was on the program committee of CiE 2022: Computability in Europe 2022 (Swansea, UK, July)
- Miller was on the program committee of HCVS-2022: Horn Clauses for Verification and Synthesis (Munich, Germany, April).
- Strassburger was on the program committee of IJCAR 2022: International Joint Conference on Automated Reasoning 2022, August 7-12, Haifa, Israel (part of FLoC 2022)
- Strassburger was on the program committee of CSL'23: Computer Science Logic 2023, Annual conference of the European Association for Computer Science Logic (EACSL), February 13-16, 2023, University of Warsaw, Poland
- Accattoli was co-chair of PPDP 2022: International Symposium on Principles and Practice of Declarative Programming 2022, 20-22 September 2022, Tbilisi, Georgia.
- Accattoli was on the program committee of APLAS 2022: Asian Symposium on Programming Languages and System, 5-10 December 2022 Auckland, New Zealand.
- Accattoli was on the program committee of LSFA 2022: International Workshop on Logical and Semantic Frameworks, with Applications, 23-24 September 2022, Belo Horizonte, Brazil.
- Chaudhuri was on the program committee of IJCAR 2022
- Scherer was on the program committees of ICFP 2022, LFMTP 2022 and the TyDe workshop 2022

**Reviewer**

- Strassburger was reviewer for the CSL 2022, IJCAR 2022, AiML 2022, and CSL 2023 conferences
- Marianela Morales was reviewer for the CSL 2023 and FoSSaCS 2023 conferences
- Accattoli reviewed papers for CSL 2023 and LICS 2022 (plus APLAS 2022, PPDP 2022, and LSFA 2022, for which he served on the PC).

### 10.1.3 Journal

**Member of the editorial boards**

- Miller is a member of the Advisory Board of the new Diamond Open Access electronic journal TheoretiCS, which publishes research work in all areas of Theoretical Computer Science.
- Miller is a member of the Journal of Automated Reasoning, published by Springer (since May 2011).
- Miller is an area editor for “Type Theory for Theorem Proving Systems” of the Journal of Applied Logic, published by Elsevier (since 2003).

**Reviewer - reviewing activities**

- Strassburger was reviewer for the journal “Mathematical Structures in Computer Science”, the journal “Logical Methods in Computer Science”, and the “Notre Dame Journal of Formal Logic”
- Accattoli reviewed papers for MSCS, Ann. pure and applied logic, LMCS, TOCL.
- Zeilberger was a reviewer for LMCS.

### 10.1.4 Invited talks

- Miller was an invited speaker at the StrIP Kick-Off Workshop, University of Birmingham, June 7-10, 2022.
- Strassburger was an invited speaker at the workshop on Logic and transdiciplinarity: Mathematics/Computer Science/Philosophy/Linguistics: 7-11 February 2022 CIRM, Marseille, France. Part of thematic month "Logic and Interactions"
- Strassburger was an invited speaker at the StrIP Kick-Off Workshop, University of Birmingham, June 7-10, 2022.
- Zeilberger gave an invited talk for the Topos Institute Colloquium on June 30, 2022.
- Zeilberger was an invited speaker at the 3rd Workshop on Proofs, Computation, and Meaning, held online December 7, 2022.
- Clarke gave an invited talk for the Topos Institute Colloquium on November 3, 2022.
- Chaudhuri was an invited speaker at the ENS Saclay for the seminar organized by the LMF/UPSCaLe (Digicosme) project

### 10.1.5 Scientific expertise

- Miller reviewed research projects for the Austrian Science Fund (FWF), the Icelandic Research Fund (RANNIS), and the Swiss National Science Foundation.

#### 10.1.6 Involvement in partner institutions

- Benjamin Werner is member of the executive boards (*conseils d'administration*) of both Ecole polytechnique and Institut Polytechnique de Paris. He is also member of the executive committee of the CS department of Ecole polytechnique.

- Kaustuv Chaudhuri is an elected member of the Conseil de Laboratoire of LIX

- Giti Omidvar is an elected member of the Conseil de Laboratoire of LIX

### 10.2 Teaching - Supervision - Juries

#### 10.2.1 Teaching

- Strassburger was teaching a course "From Axioms to Rules — The Factory of Modal Proof Systemsrial Proofs" at ESSLLI 2022.

- Wendlasida Ouedraogo was teaching assistant for the courses

  – INF442 - Algorithmes pour l'analyse de données en C++ (10h)

  – INF411 - Les bases de la programmation et de l'algorithmique (12h)

  – INF361 - Introduction à l' informatique (40h)

  – INF592 - Internship in Data Science (2h)

  at Ecole Polytechnique

- Miller was an instructor for MPRI (Master Parisien de Recherche en Informatique) in the Course 2-1: Logique linéaire et paradigmes logiques du calcul. He taught 15 hours during Fall 2022.

- Noam Zeilberger taught the third year undergraduate course "Functional Programming" in the Bachelors program at Ecole Polytechnique, and was a teaching assistant for the second year polytechnicien course INF412 "Fondements de l'informatique".

- Accattoli was an instructor for MPRI (Master Parisien de Recherche en Informatique) in the Course 2-1: Logique linéaire et paradigmes logiques du calcul. He taught 15 hours.

- Chaudhuri taught the third year undergraduate course "CSE 302: Compiler Design" at the Ecole polytechnique.

- Marianela Morales was teaching assistant at the course Computer Programming at École Polytechnique (CSE101), second semester of Bachelor of Science 1

- Benjamin Werner is responsible for the course INF371 "Mécanismes de la Programation Orientée-Objet" for first year students of the engineering program of Ecole polytechnique (190 students, 8 TAs).

- Benjamin Werner is teaching the CSE203 course "Proofs and Programs" of the Bachelor Program of Ecole polytechnique.

- Gabriel Scherer taught the first year course "Introduction à la Programmation Fonctionnelle" at Université Vincennes-Saint-Denis (Paris 8).

- Gabriel Scherer taught for MPRI in the course 2-4 (functional programming and type systems).

#### 10.2.2 Supervision

- Miller is supervising three Ph.D. students: Farah Al Wardani, Matteo Manighetti, and Jui-Hsuan Wu.
- Strassburger is supervising three PhD students: Marianela Morales, Giti Omidvar, Wendlasida Ouedraogo
- Strassburger supervised one Bachelor student: Remy Seassau
- Accattoli is supervising one PhD student, Adrienne Lancelot, since October 2022
- Accattoli supervised the internship of Adrienne Lancelot, May-September 2022.
- Zeilberger has been supervising two PhD students Nicolas Blanco and Alexandros Singh. Singh successfully defended his thesis on November 15, 2022, and Blanco is expected to defend his thesis in February 2023.
- Chaudhuri is co-supervising the PhD student Farah Al Wardani. He is also supervising the Bachelor student Luigi Massacci at the Ecole Polytechnique.
- Scherer is supervising a PhD student, Olivier Martinot.

#### 10.2.3 Juries

- Miller was the present of the jury for Gabriel Hondet, University of Paris-Saclay, 27 September 2022.

### 10.3 Popularization

#### 10.3.1 Internal or external Inria responsibilities

- Strassburger is member of the BCEP at Inria Saclay
- Scherer is a member of the CLHSCT at Inria Saclay

#### 10.3.2 Interventions

- Scherer participated to the "Fête de la science" as an INRIA researcher, introducing computer science concepts to students aged from 10 to 16.

## 11 Scientific production

### 11.1 Major publications

[1] B. Accattoli. 'Exponentials as Substitutions and the Cost of Cut Elimination in Linear Logic'. In: LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science. Haifa Israel, France: ACM, 2nd Aug. 2022, pp. 1–15. DOI: 10.1145/3531130.3532445. URL: https://hal.inria.fr/hal-03912448.

[2] B. Accattoli, U. Dal Lago and G. Vanoni. 'Multi types and reasonable space'. In: *Proceedings of the ACM on Programming Languages* 6.ICFP (29th Aug. 2022), pp. 799–825. DOI: 10.1145/3547650. URL: https://hal.inria.fr/hal-03912436.

[3] B. Accattoli, U. Dal Lago and G. Vanoni. 'Reasonable Space for the $\lambda$-Calculus, Logarithmically'. In: LICS 2022 - 37th Annual ACM/IEEE Symposium on Logic in Computer Science. Haifa, Israel: ACM, 2nd Aug. 2022, pp. 1–13. DOI: 10.1145/3531130.3533362. URL: https://hal.inria.fr/hal-03912449.

[4] M. Acclavio, R. Horne and L. Strassburger. 'An Analytic Propositional Proof System On Graphs'. In: *Logical Methods in Computer Science* 18.4 (21st Oct. 2022). DOI: 10.46298/LMCS-18(4:1)2022. URL: https://hal.inria.fr/hal-03087392.

[5] W. Heijltjes, D. Hughes and L. Strassburger. 'Normalization Without Syntax'. In: FSCD 2022. Haifa, Israel, 2nd Aug. 2022. URL: https://hal.inria.fr/hal-03654060.

[6] S. Marin, D. Miller, E. Pimentel and M. Volpe. 'From axioms to synthetic inference rules via focusing'. In: *Annals of Pure and Applied Logic* 173.5 (May 2022), p. 103091. DOI: 10.1016/j.apal.2022.103091. URL: https://hal.inria.fr/hal-03792129.

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

[8] L. T. D. Nguyên and L. Straßburger. 'BV and Pomset Logic Are Not the Same'. In: 30th EACSL Annual Conference on Computer Science Logic, CSL 2022. Göttingen, Germany, 14th Feb. 2022. DOI: 10.4230/LIPIcs.CSL.2022.32. URL: https://hal.inria.fr/hal-03909463.

## 11.2 Publications of the year

**International journals**

[9] B. Accattoli, U. Dal Lago and G. Vanoni. 'Multi types and reasonable space'. In: *Proceedings of the ACM on Programming Languages* 6.ICFP (29th Aug. 2022), pp. 799–825. DOI: 10.1145/3547650. URL: https://hal.inria.fr/hal-03912436.

[10] B. Accattoli and G. Guerrieri. 'The theory of call-by-value solvability'. In: *Proceedings of the ACM on Programming Languages* 6.ICFP (29th Aug. 2022), pp. 855–885. DOI: 10.1145/3547652. URL: https://hal.inria.fr/hal-03912446.

[11] M. Acclavio, R. Horne and L. Strassburger. 'An Analytic Propositional Proof System On Graphs'. In: *Logical Methods in Computer Science* 18.4 (21st Oct. 2022). DOI: 10.46298/LMCS-18(4:1)2022. URL: https://hal.inria.fr/hal-03087392.

[12] N. Courant, J. Lepiller and G. Scherer. 'Debootstrapping without Archeology'. In: *The Art, Science, and Engineering of Programming* 6.3 (18th Feb. 2022). DOI: 10.22152/programming-journal.org/2022/6/13. URL: https://hal.archives-ouvertes.fr/hal-03917754.

[13] S. Marin, D. Miller, E. Pimentel and M. Volpe. 'From axioms to synthetic inference rules via focusing'. In: *Annals of Pure and Applied Logic* 173.5 (May 2022), p. 103091. DOI: 10.1016/j.apal.2022.103091. URL: https://hal.inria.fr/hal-03792129.

**International peer-reviewed conferences**

[14] B. Accattoli. 'Exponentials as Substitutions and the Cost of Cut Elimination in Linear Logic'. In: LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science. Haifa Israel, France: ACM, 2nd Aug. 2022, pp. 1–15. DOI: 10.1145/3531130.3532445. URL: https://hal.inria.fr/hal-03912448.

[15] B. Accattoli, U. Dal Lago and G. Vanoni. 'Reasonable Space for the $\lambda$-Calculus, Logarithmically'. In: LICS 2022 - 37th Annual ACM/IEEE Symposium on Logic in Computer Science. Haifa, Israel: ACM, 2nd Aug. 2022, pp. 1–13. DOI: 10.1145/3531130.3533362. URL: https://hal.inria.fr/hal-03912449.

[16] B. Accattoli and M. Leberle. 'Useful Open Call-By-Need'. In: CSL 2022 - 30th EACSL Annual Conference on Computer Science Logic. Vol. 2016. 30th EACSL Annual Conference on Computer Science Logic (CSL 2022). Gottingen, Germany, 14th Feb. 2022. DOI: 10.4230/LIPIcs.CSL.2022.4. URL: https://hal.inria.fr/hal-03912452.

[17] M. Acclavio, R. Horne, S. Mauw and L. Straßburger. 'A Graphical Proof Theory of Logical Time'. In: 7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022). Haifa, Israel, 2nd Aug. 2022. DOI: 10.4230/LIPIcs.FSCD.2022.22. URL: https://hal.inria.fr/hal-03909486.

[18] M. Acclavio and L. Straßburger. 'Combinatorial Proofs for Constructive Modal Logic'. In: Advances in Modal Logic 2022. Rennes, France, 22nd Aug. 2022. URL: https://hal.inria.fr/hal-03909538.

[19] A. Ciabattoni, L. Straßburger and M. Tesi. 'Taming Bounded Depth with Nested Sequents'. In: Advances in Modal Logic 2022. Rennes, France, 22nd Aug. 2022. URL: https://hal.inria.fr/hal-03909534.

[20] W. Heijltjes, D. Hughes and L. Strassburger. 'Normalization Without Syntax'. In: FSCD 2022. Haifa, Israel, 2nd Aug. 2022. URL: https://hal.inria.fr/hal-03654060.

[21] D. Miller and J.-H. Wu. 'A positive perspective on term representation: Extended paper'. In: CSL 2023: Computer Science Logic. Warsaw, Poland, 13th Feb. 2023. URL: https://hal.inria.fr/hal-03843587.

[22] L. T. D. Nguyên and L. Straßburger. 'BV and Pomset Logic Are Not the Same'. In: 30th EACSL Annual Conference on Computer Science Logic, CSL 2022. Göttingen, Germany, 14th Feb. 2022. DOI: 10.4230/LIPIcs.CSL.2022.32. URL: https://hal.inria.fr/hal-03909463.

[23] G. Omidvar and L. Straßburger. 'Combinatorial Flows as Bicolored Atomic Flows'. In: Logic, Language, Information, and Computation - 28th International Workshop, WoLLIC 2022. Vol. 13468. Lecture Notes in Computer Science. Iaşi, Romania: Springer International Publishing, 9th Sept. 2022, pp. 141–157. DOI: 10.1007/978-3-031-15298-6_9. URL: https://hal.inria.fr/hal-03909530.

**National peer-reviewed Conferences**

[24] N. Chataing, C. Noûs and G. Scherer. 'Déboîter les constructeurs'. In: Journées Francophones des Langages Applicatifs. Saint-Médard-d'Excideuil, France, 2nd Feb. 2022. URL: https://hal.inria.fr/hal-03510931.

[25] P. Donato, P.-Y. Strub and B. Werner. 'Actema : une interface graphique et gestuelle pour preuves formelles (démonstration)'. In: *Journées Francophones des Langages Applicatifs*. 33èmes Journées Francophones des Langages Applicatifs. Saint-Médard-d'Excideuil, France, 2nd Feb. 2022, pp. 267–268. URL: https://hal.inria.fr/hal-03626854.

[26] C. Noûs and G. Scherer. 'Backtracking reference stores'. In: *Journées Francophones des Langages Applicatifs*. JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs. Praz-sur-Arly, France, 16th Jan. 2023, pp. 190–210. URL: https://hal.inria.fr/hal-03936704.

**Conferences without proceedings**

[27] P. Donato, P.-Y. Strub and B. Werner. 'A drag-and-drop proof tactic'. In: CPP 2022: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs. Philadelphia, United States: ACM; ACM, 17th Jan. 2022, pp. 197–209. DOI: 10.1145/3497775.3503692. URL: https://hal.archives-ouvertes.fr/hal-03823357.

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

[29] G. Munch-Maccagnoni and G. Scherer. 'Boxroot, fast movable GC roots for a better FFI'. In: ML Family Workshop. Ljubljana, Slovenia, 13th Sept. 2022. URL: https://hal.inria.fr/hal-03910313.

[30] G. Scherer and N. Courant. 'An OCaml use case for strong call-by-need reduction'. In: ACM SIGPLAN Workshop on ML 2022 - ML Family Workshop. Ljubljana, Slovenia, 15th Sept. 2022. URL: https://hal.inria.fr/hal-03947986.

**Reports & preprints**

[31] K. Chaudhuri, D. Miller and F. Al Wardani. *Distributing and trusting proof checking: a preliminary report*. 21st Dec. 2022. URL: https://hal.inria.fr/hal-03909741.

[32] B. Clarke and M. Di Meglio. *An introduction to enriched cofunctors*. 2nd Sept. 2022. URL: https://hal.archives-ouvertes.fr/hal-03805364.

[33] M. Manighetti and D. Miller. *Computational logic based on linear logic and fixed points*. 18th Feb. 2022. URL: https://hal.inria.fr/hal-03579451.

[34] L. T. D. Nguyễn and L. Straßburger. *A System of Interaction and Structure III: The Complexity of BV and Pomset Logic*. 2022. URL: https://hal.inria.fr/hal-03909547.

[35] W. Ouedraogo, L. Strassburger and G. Scherer. *Coqlex: Generating Formally Verified Lexers*. INRIA Saclay - Ile-de-France, 2022. URL: https://hal.science/hal-03912170.

## 11.3 Cited publications

[36] K. Chaudhuri. 'Subformula Linking as an Interaction Method'. In: *4th Conference on Interactive Theorem Proving*. Vol. 7998. Lecture Notes in Computer Science. Rennes, France: Springer, July 2013, pp. 386–401. DOI: 10.1007/978-3-642-39634-2_28. URL: https://hal.inria.fr/hal-00937009.