RESEARCH CENTRE

Inria Saclay Centre at Institut Polytechnique de Paris

IN PARTNERSHIP WITH:

CNRS, Institut Polytechnique de Paris

2023

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

- Ambroise Lafont [ECOLE POLY PALAISEAU, from Oct 2023]
- Benjamin Werner [ECOLE POLY PALAISEAU]
- Noam Zeilberger [ECOLE POLY PALAISEAU]

Post-Doctoral Fellow

- Bryce Clarke [INRIA, Post-Doctoral Fellow]

PhD Students

- Farah Al Wardani [INRIA]
- Pablo Donato [INRIA, from Nov 2023]
- Pablo Donato [ECOLE POLY PALAISEAU, until Sep 2023]
- Adrienne Lancelot [INRIA]
- Olivier Martinot [INRIA]
- Marianela Evelyn Morales Elena [Inria, until Sep 2023]
- Giti Omidvar [INRIA, until Sep 2023]
- Adonis Rima [INRIA, from Dec 2023]
- Antoine Sere [ECOLE POLY PALAISEAU, until Sep 2023]
- Jui-Hsuan Wu [IP PARIS]

Interns and Apprentices

- Ruben Bueno [ECOLE POLY PALAISEAU, Intern, from Jul 2023 until Sep 2023]
- Thea Li [ECOLE POLY PALAISEAU, Intern, from Jun 2023 until Jul 2023]

Administrative Assistant

- Michael Barbosa [INRIA]

External Collaborator

- Wendlasida Ouedraogo [SIEMENS MOBILITY, until Sep 2023]

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 λ-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 λ-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 (λ-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

5.1 Results

- We proved that intuitionistic modal logic S4 is decidable, which was an open problem for 3 decades

5.2 Awards

- Miller received the *Dov Gabbay Prize for Logic and Foundations* in 2023, for “pioneering and agenda-setting research bringing together and advancing logical proof theory and computational logic in the areas of higher-order logic programming and higher-order theorem proving.”
- The OCaml programming language received the ACM SIGPLAN Programming Languages Software Award, with Gabriel Scherer listed among the 14 co-recipients.

6 New software, platforms, open data

6.1 New software

6.1.1 Abella

Keyword: Proof assistant

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

Release Contributions: This release includes a major refactoring of the Abella documentation generator. Abella developments can now be easily converted into interactive web-based presentations that can be used without having to run Abella by the reader.

This release also fixes a number of outstanding issues with the 2.0.7 and earlier releases. At least two of these fixes involve soundness issues with regard to higher-order arguments.

Abella is now also independently packaged for MacOS (homebrew), FreeBSD, and OpenBSD.

URL: https://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

6.1.2 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.

URL: http://actema.xyz

Publication: 03823357

Contact: Benjamin Werner

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

Partner: Ecole Polytechnique

6.1.3 DAMF Dispatch

Keywords: Interactive Theorem Proving, Distributed systems, Verification

Scientific Description: The Distributed Assertion Management Framework (DAMF) is a proposed collection of formats and techniques to enable heterogeneous formal reasoning systems and users to communicate assertions in a decentralized, reliable, and egalitarian manner. An assertion is a unit of mathematical knowledge—think lemmas, theorems, corollaries, etc.—that is cryptographically signed by its originator.

DAMF is based on content-addressable storage using the InterPlanetary File System (IPFS) network, and uses the InterPlanetary Linked Data (IPLD) data model to represent assertions and all their components.

Functional Description: Dispatch is an intermediary tool for publishing, retrieval, and trust analysis in the Distributed Assertion Management Framework (DAMF). Dispatch specifies a family of JSON-based formats for DAMF objects and implements the main DAMF processes. It is intended to be usable by both human users and tools.

Dispatch is being developed as part of the exploratory action W3Proof.

Release Contributions: This initial version has a demonstration proof of a theorem using a combination of Coq, LambdaProlog, and Abella.

URL: https://distributed-assertions.github.io

Publication: hal-04167922

Contact: Kaustuv Chaudhuri

Participants: Farah Al Wardani, Kaustuv Chaudhuri, Dale Miller

6.1.4 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

6.1.5 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, David Allsopp, Stephen Dolan, Alain Frisch, Jacques Garrigue, Anil Madhavapeddy, Kc Sivaramakrishnan, Nicolas Ojeda Bar, Leo White

6.1.6 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

6.1.7 Profound-Intuitionistic

Name: Interactive theorem proving by direct manipulation for Intuitionistic Logic

Keywords: Interactive Theorem Proving, First-order logic

Functional Description: Profound-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 inductive theorem proving using sized relations in the style of Abella.

This release also adds preliminary support for three-dimensional representations of the proof state (using WebGL and the Three.js library).

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

Contact: Kaustuv Chaudhuri

6.1.8 YADE

Name: Yet Another Diagram Editor

Keyword: Diagram

Functional Description: This diagram editor can help mechanising diagrammatic categorical proofs by generating Coq proof scripts from a drawn diagram. This is part of the Coreact ANR Project (started in March 2023), which aims at developing a methodology for diagrammatic reasoning in Coq.

URL: https://amblafont.github.io/graph-editor/index.html

Contact: Ambroise Lafont

Participant: Ambroise Lafont

7 New results

7.1 The Complexity of BV and Pomset Logic

Participants: Lutz Straßburger.

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

Following our result from last year, where we have shown that BV and pomset logic are different [27], we have continued our efforts and studied the complexity of the two logics. Our results are published in [10].

7.2 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 [11]

7.3 Intuitionistic S4 is decidable

Participants: Lutz Straßburger, Marianela Evelyn Morales Elena.

External Collaborators: Marianna Girlando (Amsterdam), Sonia Marin (Birmingham), Roman Kuznets (Vienna)

In this work, published in [20], we demonstrate decidability for the intuitionistic modal logic S4 first formulated by Fischer Servi. This solves a problem that has been open for almost thirty years since it had been posed in Simpson's PhD thesis in 1994. We obtain this result by performing proof search in a labelled deductive system that, instead of using only one binary relation on the labels, employs two: one corresponding to the accessibility relation of modal logic and the other corresponding to the order relation of intuitionistic Kripke frames. Our search algorithm outputs either a proof or a finite counter-model, thus, additionally establishing the finite model property for intuitionistic S4, which has been another long-standing open problem in the area.

7.4 Term representation and proof theory

Participants: Beniamino Accattoli, Dale Miller, Jui-Hsuan Wu.

Structural proof theory has been used to provide a principled approach to designing term representations and operations (such as substitutions) on them. In the past, the negative polarity approach has been the only one used. In our recent work, we have considered, instead, using the positive polarity approach. This has lead us to a principled description of sharing within terms. While sharing has been described previously in various setting, the one based on proof theory has certain advantages since it is less ad hoc and comes with its own notion of substitution (derived immediately from using cut elimination). This approach also provides a natural setting for treating binding structures within programs. Details of this approach are available from our CSL 2023 paper [13]. Wu has developed this positive perspective further in his APLAS 2023 paper [21].

7.5 Formal Reasoning using Distributed Assertions

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

When a proof system checks a formal proof, we can say that its kernel asserts that the formula is a theorem in a particular logic. We describe a general framework in which such assertions can be made globally available so that any proof assistant willing to trust the assertion's creator can use that assertion without rechecking any associated formal proof. This framework, called DAMF, is heterogeneous and allows each participant to decide which tools and operators they are willing to trust in order to accept external assertions. This framework can also be integrated into existing proof systems by making minor changes to the input and output subsystems of the prover. DAMF achieves a high level of distributivity using off-the-shelf technologies such as IPFS, IPLD, and public key cryptography. We illustrate the framework by describing an implemented tool, called DISPATCH, for validating and publishing assertion objects and a modified version of the Abella theorem prover that can use and publish such assertions. The details of the DAMF system can be found in the FroCos 2023 paper [17] and the earlier report [24].

7.6 A foundation for proof theory based on searching for proofs

Participants: Dale Miller.

In 1935, Gentzen captured the way human think about formal proofs using the natural deduction proof system. This system was based on describing complete proofs and is based on a kind of annotated tree structure. In practice, however, it is also natural to consider the process of extending partial proofs until they become complete. In his LICS 2023 paper [12], Miller describes a simply motivated way to describe proofs based on proof search based on multiset rewriting of various annotated formula objects. This proof framework, called PSF, incorporates many structural features of proof (many, comming from Linear Logic) that were not part of Gentzen's original natural deduction proof systems.

7.7 Convolution Products on Double Categories and Categorification of Rule Algebras

Participants: Noam Zeilberger.

External Collaborators: Nicolas Behr (IRIF), Paul-André Melliès (IRIF)

Motivated by compositional categorical rewriting theory, we introduce a convolution product over presheaves of double categories which generalizes the usual Day tensor product of presheaves of monoidal categories. One interesting aspect of the construction is that this convolution product is in general only oplax associative. For that reason, we identify several classes of double categories for which the convolution product is not just oplax associative, but fully associative. This includes in particular framed bicategories on the one hand, and double categories of compositional rewriting theories on the other. For the latter, we establish a formula which justifies the view that the convolution product categorifies the rule algebra product. These results were published at FSCD 2023 [18].

7.8 The Algebraic Weak Factorisation System for Delta Lenses

Participants: Bryce Clarke.

Delta lenses are functors equipped with a suitable choice of lifts, and are used to model bidirectional transformations between systems. In this paper, we construct an algebraic weak factorisation system whose R-algebras are delta lenses. Our approach extends a semi-monad for delta lenses previously introduced by Johnson and Rosebrugh, and generalises to any suitable category equipped with an orthogonal factorisation system and an idempotent comonad. We demonstrate how the framework of an algebraic weak factorisation system provides a natural setting for understanding the lifting operation of a delta lens, and also present an explicit description of the free delta lens on a functor. Published in [19]

7.9 Formalizing Functions as Processes

Participants: Beniamino Accattoli.

External Collaborators: Horace Blanc (independent researcher), Claudio Sacerdoti Coen (Bologna University).

This work develops the first formalization of Milner's classic translation of the λ-calculus into the π-calculus. It is a challenging result with respect to variables, names, and binders, as it requires one to relate variables and binders of the λ-calculus with names and binders in the π-calculus. We formalize it in Abella, merging the set of variables and the set of names, thus circumventing the challenge and obtaining a neat formalization. About the translation, we follow Accattoli's factoring of Milner's result via the linear substitution calculus, which is a λ-calculus with explicit substitutions and contextual rewriting rules, mediating between the λ-calculus and the π-calculus. Another aim of the formalization is to investigate to which extent the use of contexts in Accattoli's refinement can be formalized.

This work has been published in the proceedings of an international conference [15].

7.10 Sharing a Perspective on the λ-calculus

Participants: Beniamino Accattoli.

This is an essay about the lambda calculus, published in the proceedings of an international conference for essays on programming languages and related topics [26].

The essay discusses a gap between the theory of the lambda calculus and functional languages, namely the fact that the former does not give a status to sharing, the essential ingredient for efficiency in the latter. The essay provides an overview of the perspective of the author, who has been and still is studying sharing from various angles. In particular, it explains how sharing impacts the equational and denotational semantics of the lambda calculus, breaking some expected properties, and demanding the development of new, richer semantics of the lambda calculus.

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

Participants: Beniamino Accattoli.

This work is the extended journal version [9] of the conference paper with the same title from 2022. It was selected for the special issue of the conference LICS 2022.

This paper introduces the exponential substitution calculus (ESC), a new presentation of cut elimination for IMELL, based on proof terms and building on the idea that exponentials can be seen as explicit substitutions. 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, that is the key ingredient for the study of reasonable time cost models for the λ-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.

7.12 A Diamond Machine for Strong Evaluation

Participants: Beniamino Accattoli.

External Collaborators: Pablo Barenbaum (Universidad Nacional de Quilmes & 3 Universidad de Buenos Aires, Argentina).

Abstract machines for strong evaluation of the λ-calculus enter into arguments and have a set of transitions for backtracking out of an evaluated argument. We study a new abstract machine which avoids backtracking by splitting the run of the machine in smaller jobs, one for argument, and that jumps directly to the next job once one is finished. Usually, machines are also deterministic and implement deterministic strategies. Here we weaken this aspect and consider a light form of non- determinism, namely the diamond property, for both the machine and the strategy. For the machine, this introduces a modular management of jobs, parametric in a scheduling policy. We then show how to obtain various strategies, among which leftmost-outermost evaluation, by instantiating in different ways the scheduling policy.

This work has been published in the proceedings of an international conference [14].

7.13 Strong Call-by-Value and Multi Types

Participants: Beniamino Accattoli.

External Collaborators: Giulio Guerrieri (Aix Marseille University), Maico Leberle (independent researcher).

This paper provides foundations for strong (that is, possibly under abstraction) call-by-value evaluation for the λ-calculus. Recently, Accattoli et al. proposed a form of call-by-value strong evaluation for the λ-calculus, the external strategy, and proved it reasonable for time. Here, we study the external strategy using a semantical tool, namely Ehrhard's call-by-value multi types, a variant of intersection types. We show that the external strategy terminates exactly when a term is typable with so-called shrinking multi types, mimicking similar results for strong call-by-name. Additionally, the external strategy is normalizing in the untyped setting, that is, it reaches the normal form whenever it exists.

We also consider the call-by-extended-value approach to strong evalua- tion shown reasonable for time by Biernacka et al. The two approaches turn out to not be equivalent: terms may be externally divergent but terminating for call-by-extended-value.

This work has been published in the proceedings of an international conference [16].

7.14 Unboxed data constructors

Participants: Gabriel Scherer.

External Collaborators: Nicolas Chataing (ENS Paris), Stephen Dolan (Jane Street), Jeremy Yallop (Cambridge University)

This work proposes a new feature for the OCaml programming language that improves the programmer's control over the low-level representation of OCaml datatypes, by "unboxing" certain constructors of a variant type. This is useful in certain advanced situations to improve performance or reduce space usage.

Along the way we discovered an interesting theoretical problem related to the question of deciding normalization in lambda-calculi with recursive definitions. We propose an "on-the-fly" normalization algorithm in the first-order fragment of the simply-typed lambda-calculus with just functions. Its relation with existing higher-order algorithms is unclear and deserves further study.

This result was accepted for publication in 2024.

7.15 Backtracking reference store

Participants: Gabriel Scherer.

We are working on a library that provides mutable references with backtracking. That is, it is possible to take a snapshot of a group of references at once, and later go back to this snapshot, restoring the value of these references. This provides an easy way to equip imperative data-structures with backtracking; for example, we use this to implement a backtracking search algorithm that maintains an efficient Union-Find data structure.

Preliminary results in this direction were published at [22], and we are working on an improved version and possibly a formally verified implementation.

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.

Remark: The contract formally ended in December 2022, but we worked on it until September 2023.

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.

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.

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

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: Willem Heijltjes (w.b.heijltjes@bath.ac.uk)

Partners:

- Université de Bath (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.

9.2 International research visitors

9.2.1 Visits to international teams

Research stays abroad Miller made a 10-day research visit to the University of Birmingham to work with Prof. Anupam Das and his team during 29 November - 8 December 2023.

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 Committee for LFMTP: Logical Frameworks and Metatheory: Theory and Practice until June 2024.
- Zeilberger is a member of the Steering Committee for CLA: Computational Logic and Applications.
- Accattoli is a member of the Steering Committee for PPDP: Principle and Practice of Declarative Programming.

10.1.2 Scientific events: selection

Chair of conference program committees

- Straßburger was area chair for "Logic and Computation" at the 34th European Summer School in Logic, Language and Information (ESSLLI 2023)
- Gabriel Scherer was the chair of the OCaml Workshop in 2023, colocated with ICFP.

Member of the conference program committees

- Straßburger was member of the PC for the following conferences:
  - 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
  - TABLEAUX 2023: 32nd International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, September 18-21, 2023, Prague, Czech Republic.
- Miller was a member of the PC for LPAR-24: International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Manizales, Colombia, 4-9 June 2023.
- Accattoli was a member of the PC for PPDP 2023: 25th International Symposium on Principles and Practice of Declarative Programming, Cascais, Portugal, 22-23 October 2023.

Reviewer

- Lutz Straßburger was reviewer for LICS 2023 and FSCD 2023.
- Noam Zeilberger was reviewer for CSL 2023.
- Ambroise Lafont was reviewer for CSL 2024.
- Scherer was reviewer for FSCD 2023, and the ML Family Workshop at ICFP.

10.1.3 Journal

Member of the editorial boards

- Miller has been a member of Editorial Board Journal of Automated Reasoning, published by Springer, since May 2011.
- Miller is a member of the Advisory Board for the TheoretiCS online journal.

Reviewer - reviewing activities

- Straßburger was reviewer for the Journal of Symbolic Logic and the Journal of Logic and Computation
- Miller reviewed for the online journal *Logical Methods in Computer Science*.
- Zeilberger reviewed for the journals *Compositionality*, *Logical Methods in Computer Science*, and *Mathematical Structures in Computer Science*.
- Accattoli reviewed for the online journal *Logical Methods in Computer Science* (3 papers).

10.1.4 Invited talks

- Miller has been an invited speaker at LICS 2023 (Boston, 26-29 June 2023) and CSL 2023 (Warsaw, Poland, 13-16 February 2023).
- Accattoli has been invited speaker for the evening talks of the ESSLLI 2023 summer school.
- Scherer was an invited speaker at the SCALP meeting (a "groupe de travail" of the GDR IM.)

10.1.5 Leadership within the scientific community

- Werner is member of the Executive Boards (conseils d'administration) of Ecole polytechnique and Institut Polytechnique de Paris.
- Omidvar was elected member of the Conseil de Laboratoire of LIX

10.1.6 Scientific expertise

- Miller is a Member of ACM's Heidelberg Laureate Forum Young Researcher Selection Committee for three years starting 2023.

10.1.7 Research administration

- Straßburger reviewed for the Austrian funding agency FWF.
- Miller reviewed proposals for the Portuguese funding agency FCT.
- Zeilberger reviewed for the Agence Nationale de la Recherche (ANR).
- Scherer served on hiring committees for three MCF positions at Université Paris Cité, and one position at École Polytechnique.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Miller lectures for 12 hours in the Master level course MPRI 2-1 "Logique linéaire et paradigmes logiques du calcul" in October 2023.
- Werner lectures for 24 hours in the Master level course MPRI 2-7-1 "Fondements des Systèmes de Preuves" in the fall 2023.
- Zeilberger, Lafont and Werner teach within the Bachelors and Polytechnicien programs of Ecole Polytechnique.
- Accattoli lectures for 16 hours in the Master level course MPRI 2-1 "Logique linéaire et paradigmes logiques du calcul" in January-February 2023.
- Werner lectures and coordinates the course INF371, "Mécanismes d'un Langage Orienté Objet" for 200 *élèves ingénieurs polytechniciens*.
- Accattoli taught a course "Time ans Space for the lambda Calculus" at the 34th European Summer School In Logic, Language and Information (ESSLLI 2023).

10.2.2 Supervision

- Straßburger supervised the following PhD students: Wendlasida Ouedraogo (defended 15 September, 2023), Marianela Morales (defended 8 December, 2023), and Giti Omidvar (defended 11 December, 2023)
- Miller supervised the Ph.D. student Matteo Manighetti, who defended his dissertation on 9 February 2023.
- Chaudhuri and Miller supervised the following PhD student: Farah al Wardani.
- Accattoli and Miller supervised the following PhD student: Hsuan-Jui Wu.
- Zeilberger's PhD student Nicolas Blanco successfully defended his thesis in February 2023.
- Accattoli supervised the following PhD student: Adrienne Lancelot.
- Scherer supervised his PhD student Olivier Martinot.
- Werner supervised PhD students Pablo Donato and Antoine Séré

10.3 Popularization

10.3.1 Internal or external Inria responsibilities

- Straßburger is member of the BCEP at Inria Saclay.
- Accattoli is member of the Commission Scientifique at Inria Saclay
- Scherer is member of the CLHSCT at Inria Saclay

10.3.2 Articles and contents

- Accattoli wrote an essay presented and published in the conference Onward! Essays [26].

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 λ-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. 'Exponentials as Substitutions and the Cost of Cut Elimination in Linear Logic'. In: *Logical Methods in Computer Science* Volume 19, Issue 4 (14th Dec. 2023). DOI: 10.46298/lmcs-19(4:23)2023. URL: https://hal.science/hal-04374165.

[10] L. T. D. Nguyên and L. Straßburger. 'A System of Interaction and Structure III: The Complexity of BV and Pomset Logic'. In: *Logical Methods in Computer Science* 9.4 (18th Dec. 2023). DOI: 10.46298/LMCS-19(4:25)2023. URL: https://inria.hal.science/hal-03909547.

[11] W. Ouedraogo, L. Strassburger and G. Scherer. 'Coqlex: Generating Formally Verified Lexers'. In: *The Art, Science, and Engineering of Programming* 8.1 (15th June 2023). DOI: 10.22152/programming-journal.org/2024/8/3. URL: https://hal.science/hal-03912170.

Invited conferences

[12] D. Miller. 'A system of inference based on proof search: an extended abstract'. In: LICS 2023 - 38th Annual ACM/IEEE Symposium on Logic in Computer Science. Boston, United States: IEEE, 26th June 2023, pp. 1–11. DOI: 10.1109/LICS56636.2023.10175827. URL: https://inria.hal.science/hal-04169014.

[13] 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://inria.hal.science/hal-03843587.

International peer-reviewed conferences

[14] B. Accattoli and P. Barenbaum. 'A Diamond Machine for Strong Evaluation'. In: APLAS 2023 - The 21st Asian Symposium on Programming Languages and Systems. Taipei, Taiwan, 26th Nov. 2023. URL: https://hal.science/hal-04395635.

[15] B. Accattoli, H. Blanc and C. Sacerdoti Coen. 'Formalizing Functions as Processes'. In: ITP 2023 - 14th International Conference on Interactive Theorem Proving. Bialystok, Poland: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPICS.ITP.2023.5. URL: https://hal.science/hal-04280546.

[16] B. Accattoli, G. Guerrieri and M. Leberle. 'Strong Call-by-Value and Multi Types'. In: ICTAC 2023 - 20th International Colloquium on Theoretical Aspects of Computing. Lima, Peru, 4th Dec. 2023. URL: https://hal.science/hal-04395549.

[17] F. Al Wardani, K. Chaudhuri and D. Miller. 'Formal Reasoning using Distributed Assertions'. In: *Proceedings of the 14th International Conference on Frontiers of Combining Systems (FroCoS)*. FroCoS 2023 - 14th International Symposium on Frontiers of Combining Systems. Prague (Czech Republic), Czech Republic, 20th Sept. 2023. URL: https://inria.hal.science/hal-04167922.

[18] N. Behr, P.-A. Melliès and N. Zeilberger. 'Convolution Products on Double Categories and Categorification of Rule Algebras'. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. FSCD 2023 - 8th International Conference on Formal Structures for Computation and Deduction. Vol. 260. 8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023). Rome, Italy: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28th June 2023, 17:1–17:20. DOI: 10.4230/LIPIcs.FSCD.2023.17. URL: https://hal.science/hal-04222049.

[19] B. Clarke. 'The Algebraic Weak Factorisation System for Delta Lenses'. In: *Electronic Proceedings in Theoretical Computer Science*. Proceedings of the Sixth International Conference on Applied Category Theory 2023. Vol. 397. University of Maryland, College Park, MD, United States, 14th Dec. 2023, pp. 54–69. DOI: 10.4204/EPTCS.397.4. URL: https://inria.hal.science/hal-04089742.

[20] M. Girlando, R. Kuznets, S. Marin, M. Morales and L. Strassburger. 'Intuitionistic S4 is decidable'. In: LICS 2023- 38th Annual ACM/IEEE Symposium on Logic in Computer Science. 2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). Boston, United States: IEEE, 24th Apr. 2023, pp. 1–13. DOI: 10.1109/LICS56636.2023.10175684. URL: https://inria.hal.science/hal-04267899.

[21] J.-H. Wu. 'Proofs as Terms, Terms as Graphs'. In: APLAS 2023 - 21st Asian Symposium on Programming Languages and Systems. Taipei, Taiwan, 27th Nov. 2023. URL: https://inria.hal.science/hal-04222527.

National peer-reviewed Conferences

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

Doctoral dissertations and habilitation theses

[23] M. Manighetti. 'Developing proof theory for proof exchange'. Institut Polytechnique de Paris, 9th Feb. 2023. URL: https://theses.hal.science/tel-04289251.

Reports & preprints

[24] F. Al Wardani, K. Chaudhuri and D. Miller. *Distributing and trusting proof checking: a preliminary report*. 10th Mar. 2023. URL: https://inria.hal.science/hal-03909741.

[25] P.-A. Melliès and N. Zeilberger. *The categorical contours of the Chomsky-Schützenberger representation theorem*. 29th Dec. 2023. URL: https://hal.science/hal-04399404.

11.3 Other

Scientific popularization

[26] B. Accattoli. 'Sharing a Perspective on the lambda Calculus'. In: Onward! 2023 - ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. Cascais, Portugal: ACM, 18th Oct. 2023, pp. 179–190. DOI: 10.1145/3622758.3622884. URL: https://hal.science/hal-04280550.

11.4 Cited publications

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