RESEARCH CENTRE

Saclay - Île-de-France

IN PARTNERSHIP WITH:

CNRS, Ecole Polytechnique

2020

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

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, Researcher, HDR]
- Beniamino Accattoli [Inria, Researcher]
- Nicolas Blanco [Université de Birmingham, Researcher, from Sep 2020 until Oct 2020]
- Kaustuv Chaudhuri [Inria, Researcher]
- François Lamarche [Inria, Senior Researcher]
- Ian Mackie [CNRS, Researcher]
- Matteo Manighetti [Univ Paris Sciences et Lettres, Researcher, until May 2020]
- Dale Miller [Inria, Senior Researcher]
- Gabriel Scherer [Inria, Researcher]
- Noam Zeilberger [École polytechnique, Researcher]

Post-Doctoral Fellows

- Matteo Acclavio [Telecom Paris Sud, until May 2020]
- Marianna Girlando [Inria]
- Luc Pellissier [Inria, until Aug 2020]

PhD Students

- Maico Carlos Leberle [Inria]
- Matteo Manighetti [Inria, from Jun 2020]
- Olivier Martinot [Inria, from Sep 2020]
- Marianela Evelyn Morales Elena [École polytechnique]
- Giti Omidvar [Inria, from Oct 2020]
- Wendlasida Ouedraogo [Siemens Mobility]

Interns and Apprentices

- Olivier Martinot [Inria, from Apr 2020 until Sep 2020]
- Francesco Mecca [Inria, until Mar 2020]
- Giti Omidvar [Inria, from Mar 2020 until Aug 2020]
- Jui Hsuan Wu [Inria, from Apr 2020 until Aug 2020]

Administrative Assistant

- Bahar Carabetta [Inria, from Oct 2020 until Nov 2020]

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 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

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

[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.

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.

  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.

[3] The *Curry-Howard* correspondence.

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 Social and environmental responsibility

Partout participated to the discussion within INRIA Saclay and LIX of a new multi-year law on research (LPPR: Loi Pluriannuelle de Programamtion de la Recherche). In particular, Partout team members (Gabriel Scherer and Luc Pellissier) organized and ran the main physical meeting of the lab / research centre on this topic in March 2020.

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

Contacts: Lutz Straßburger, Marianna Girlando

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 and System Z), a debugger, a documentation generator, a compilation manager, a package manager, and many libraries contributed by the user community.

URL: https://ocaml.org/

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

Contacts: Xavier Leroy, Damien Doligez

Participants: Damien Doligez, Xavier Leroy, Fabrice Le Fessant, 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/

Contacts: Dale Miller, 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

7 New results

7.1 A Proof System on Graphs

Participants: Matteo Acclavio, Lutz Straßburger

External Collaborators: Ross Horne, University of Luxembourg

In this work we developed a proof system that does not have formulas as its basic object of reasoning but arbitrary graphs. More precisey, we start from the well-known relationship between formulas and cographs, which are undirected graphs that do not have P_4 (the four-vertex path) as vertex-induced subgraph. In this way, any standard proof system can be seen as a system manipulating cographs. In our work we drop the cograph condition and look at arbitrary (undirected) graphs. That means that instead of formulas/cographs, our basic objects of reasoning in a proof system are just graphs. The consequence is that we lose the tree structure of the formulas. Therefore we cannot 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 (MLL) with mix, meaning that if a graph is a cograph and provable in our system, then it is also provable in MLL+mix. We also invetigate the notion of generalized connective. This work has been published in [8] and [24]. Related work on generalized connectives is [9]

7.2 Proof Theory for Intuitionistic Modal Logics

Participants: Marianna Girlando, Marianela Morales, Lutz Straßburger

External Collaborators: Sonia Marin, UCL

We we continued our work on the development of labelled sequent systems and a nested sequent systems for intuitionistic modal logics. In particular, our labelled systems are equipped with two relation symbols, one for the accessibility relation associated with the Kripke semantics for modal logics and one for the preorder relation associated with the Kripke semantics for intuitionistic logic.

The nested sequent systems come in three flavours: First, standard Gentzen-style single-conclusion systems; second Maehara-style multiple conclusion systems; and third, fully structured with two kinds of brackets, one We present a labelled sequent system and a nested sequent system for intuitionisticmodal

logics equipped with two relation symbols, one for the accessibility relation associated with the Kripke semantics for modal logics and one for the preorder relation associated with the Kripke semantics for intuitionistic logic. For all systems we established a close correspondence with the bi-relational Kripke semantics for intuitionistic modal logic.

The result of this year are published in [13, 26, 20, 19]

7.3 A Practical Mode System for Recursive Definitions

Participants: Gabriel Scherer

External Collaborators: Alban Reynaud (ENS Lyon), Jeremy Yallop (University of Cambridge, UK)

This works exposes a new approach to checking recursive definitions, designed to solve a technical difficulty in the design of the OCaml programming language, that was causing the compiler to accept erroneous (memory-unsafe) programs involving certain classes of recursive definitions. Our involvement in this work started during the 2018 internship of Alban Reynaud in our project-team, followed by development work resulting in the inclusion of our new criterion in the OCaml compiler in 2019, and finally an academic publication (to be published in January 2021 [4]) that details the new check, studies its theory and proves its correctness in an idealized setting.

7.4 Translation-validation of a (simple) pattern-matching compiler

Participants: Gabriel Scherer

External Collaborators: Francesco Mecca (University of Turin, Italy)

We propose an algorithm for the translation validation of a pattern matching compiler for a small subset of the OCaml pattern matching features. Given a source program and its compiled version the algorithm checks wheter the two are equivalent or produce a counter example in case of a mismatch.

Our equivalence algorithm works with decision trees. Source patterns are converted into a decision tree using matrix decomposition. Target programs, described in a subset of the Lambda intermediate representation of the OCaml compiler, are turned into decision trees by applying symbolic execution.

This work, presenting the results of Francesco Mecca's master internship in our project-team, was presented at the "ML Family Workshop" in August 2020 [22].

7.5 Extending the Inferno type-inference approach to realistic type-system features

Participants: Gabriel Scherer, Olivier Martinot

Inferno is a software library from François Pottier (EPI Cambium) to implement constraint-based type-inference in a pleasant, declarative style. It contains a proof-of-concept inference engine for a very small programming language, but it is not obvious how to scale its declarative style to richer language features.

Olivier Martinot, as a master intern and then a beginning PhD student, has been working with Gabriel Scherer on extending the Inferno approach to more language features, hoping to eventually cover a large subset of the OCaml type system.

Olivier presented a part of his master work at the "ML Family Workshop" 2020, [21].

7.6 Inferential Semantics as Argumentative Dialogues

Participants: Luc Pellissier

External Collaborators: Davide Catta (LIRMM, Montpellier), Christian Rétoré (LIRMM, Montpellier)

According to inferentialism, the meaning of a statement lies in its argumentative use, its justifications, its refutations and more generally its deductive relation to other statements. Luc Pellissier worked on a first step towards an "implementation" of the inferentialist view of meaning, and a first proposal for a logical structure which describes an argumentation. The work [11] proposes a simple notion of argumentative dialogue, which can be either carried in purely logical terms or in natural language.

7.7 A distributed and trusted web of formal proofs

Participants: Dale Miller

Most computer checked proofs are tied to the particular technology of a prover's software. While sharing results between proof assistants is a recognized and desirable goal, the current organization of theorem proving tools makes such sharing an exception instead of the rule. In fact, the current architecture of proof assistants and formal proofs needs to be turned inside-out. That is, instead of having a few mature theorem provers include within them their formally checked theorems and proofs, proof assistants should sit on the edge of a web of formal proofs and proof assistants should be exporting their proofs so that they can exist independently of any theorem prover. While it is necessary to maintain the dependencies between definitions, theories, and theorems, no explicit library structure should be imposed on this web of formal proofs. Thus a theorem and its proofs should not necessarily be located at a particular URL or within a particular prover's library. While the world of symbolic logic and proof theory certainly allows for proofs to be seen as global and permanent objects, there is a lot of research and engineering work that is needed to make this possible. The W3Proof *Action Exploratoire* is based on these observations and goals.

This work has been published in [14].

7.8 Using Lambda-Prolog as a proof refiner within Coq

Participants: Matteo Manighetti, Dale Miller

External Collaborators: Roberto Blanco, Max Planck Institute for Security and Privacy; Enrico Tassi, Inria Sophia Antipolis.

Logic programming implementations of the foundational proof certificate (FPC) framework are capable of checking a wide range of proof evidence. Proof checkers based on logic programming can make use of both unification and backtracking search to allow varying degrees of proof reconstruction to take place during proof checking. Such proof checkers are also able to elaborate proofs lacking full details into proofs containing much more detail. We are using the Coq-Elpiplugin, which embeds an implementation of λProlog into Coq, to check proof certificates supplied by external (to Coq) provers and to elaborate them into the fully detailed proof terms that are needed for checking by the Coq kernel.

This work has been published in [25]. This work was also presented as "work-in-progress" at the LFMTP 2020 workshop in June 2020.

7.9 Extrinsically Typed Operational Semantics for Functional Languages

Participants: Dale Miller

External Collaborators: Matteo Cimini, University of Massachusetts, Lowell; Jeremy Siek, Indiana University, Bloomington.

In this work, we present a type system over language definitions that classifies parts of the operational semantics of a language in input, and models a common language design organization. The resulting typing discipline guarantees that the language at hand is automatically type sound. Thanks to the use of types to model language design, our type checker has a high-level view on the language being analyzed and can report messages using the same jargon of language designers. We have implemented our type system in the LANG-N-CHECK tool, and we have applied it to derive the type soundness of several functional languages, including those with recursive types, polymorphism, exceptions, lists, sums, and several common types and operators. Our system is designed to output proof scripts of language correctness using the Abella theorem prover.

This work has been published in [12].

7.10 The Machinery of Interaction

Participants: Beniamino Accattoli

External Collaborators: Ugo Dal Lago, University of Bologna & Inria; Gabriele Vanoni, University of Bologna & Inria.

This work revisits the Interaction Abstract Machine (IAM), a machine based on Girard's Geometry of Interaction. It is an unusual machine, radically different with respect to the mainstream paradigm of

environment-based machines for functional languages. The soundness proof in the literature—due to Danos, Regnier & Herbelin—is convoluted and passes through various other formalisms. Here we provide a new direct proof of its correctness, based on a variant of Sands's improvements, a natural notion of bisimulation. Moreover, our proof is carried out on a new presentation of the IAM, defined as a machine acting directly on λ-terms,rather than on linear logic proof nets.

The work was the first step towards the complexity analysis of the IAM. As such, it belongs to the research theme *Foundations of complexity analysis for functional programs*.

This work has been published in [5].

7.11 The Distributive lambda-Calculus

Participants: Beniamino Accattoli

External Collaborators: Alejandro Díaz-Caro, CONICET & University of Buenos Aires & University Nacional de Quilmes (Argentina).

We introduce a simple extension of the λ-calculus with pairs—called the distributive λ-calculus—obtained by adding a computational interpretation of the distributivity isomorphism $A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C)$ of simple types. We study the calculus both as an untyped and as a simply typed setting. Key features of the untyped calculus are confluence, the absence of clashes of constructs, that is, evaluation never gets stuck, and a leftmost-outermost normalization theorem, obtained with straightforward proofs. With respect to simple types, we show that the new rules satisfy subject reduction if types are considered up to the distributivity isomorphism. The main result is strong normalization for simple types up to distributivity. The proof is a smooth variation over the one for the λ-calculus with pairs and simple types.

This work arose as a collaboration while Accattoli was visiting the university of Buenos Aires in the summer 2019 to teach at the ECI 2019 summer school.

This work has been published in [6].

7.12 Tight Typings and Split Bounds, Fully Developed

Participants: Beniamino Accattoli

External Collaborators: Stéphane Graham-Lengrand, SRI International; Delia Kesner, Université de Paris & CNRS.

This line of work extends a work titled "Tight Typings and Split Bounds" previously published in the conference ICFP 2018 by the same authors, and included in 2018 report of the PARSIFAL team.

Essentially, that paper was about the use of *multi types*, a variant of intersection types, to extract bounds over evaluation lengths and the size of normal form of typed terms, with respect to different evaluation strategies and notions of normal form.

Here we refined and extended many of the results in the conference versions, producing an extended journal version [3]

7.13 Factorize Factorization

Participants: Beniamino Accattoli

External Collaborators: Claudia Faggian, Université de Paris & CNRS, France; Giulio Guerrieri University of Bath, UK.

This is a work about rewriting theory, with applications to the λ-calculus and its variants.

We present a new technique for proving factorization theorems for compound rewriting systems in a modular way, which is inspired by the Hindley-Rosen technique for confluence. Specifically, our approach is well adapted to deal with extensions of the call-by-name and call-by-value λ-calculi.

The technique is first developed abstractly. We isolate a sufficient condition (called linear swap) for lifting factorization from components to the compound system, and which is compatible with β-reduction. We then closely analyze some common factorization schemas for the λ-calculus.

Concretely, we apply our technique to diverse extensions of the λ-calculus, among which de' Liguoro and Piperno's non-deterministic λ-calculus and – for call-by-value – Carraro and Guerrieri's shuffling calculus. For both calculi the literature contains factorization theorems. In both cases, we give a new proof which is neat, simpler than the original, and strikingly shorter.

The work has been published in [7]

7.14 The proof theory of skew monoidal categories and related structures

Participants: Noam Zeilberger

External Collaborators: Tarmo Uustalu, Reykjavik University and Tallinn University of Technology; Niccolò Veltri, Tallinn University of Technology.

Skew monoidal categories are a well-motivated generalization of monoidal categories where the three structural laws of left and right unitality and associativity are not required to be isomorphisms but merely transformations in a particular direction. They have been thoroughly studied from a categorical perspective since being axiomatized by Szlachányi (2012), and in a programming languages context, they were considered by Uustalu as an outgrowth of his influential work on *relative monads* (Altenkirch, Chapman, Uustalu 2015). The simpler setting where one drops the unit laws and only keeps an ordered associativity law $(A \otimes B) \otimes C \to A \otimes (B \otimes C)$ is the algebraic counterpart of the classical *Tamari order*, which has likewise seen a resurgence of interest from the enumerative combinatorics community following an unexpected result by Chapoton (2006) on the number of intervals in Tamari lattices.

In "A sequent calculus for a semi-associative law" (FSCD'2017, extended version published as LMCS 15:1, 2019), Zeilberger observed that the Tamari order is precisely captured by a sequent calculus with a simple variation of the rules in Lambek (1958). This was used to give a surprising application of proof theory to combinatorics, in particular a new proof of Chapoton's result by way of a cut-elimination and focusing theorem for the sequent calculus, as well as a new proof of the original result by Friedman and Tamari (1961) that the order satisfies a lattice property.

These two independent threads were brought together in the collaboration by the three named authors, which resulted in four papers published in 2020. [23] is a longer version of a paper at MFPS'2018, in which we showed how the sequent calculus for the Tamari order may be extended to a sequent calculus that precisely captures skew monoidal categories, in the sense that we can use it to prove a coherence theorem. In [17], we further extended this approach to capture "partially skew" categories with different sets of normality conditions. Our MFPS'2020 paper [16] had a more semantic bent, studying the relationship between skew monoidal categories and *skew closed categories*. Finally, [15] investigated the proof theory of skew prounital closed categories, which have good motivations as models of ordered linear lambda calculus.

7.15 Bifibrations of polycategories and classical linear logic

Participants: Noam Zeilberger

External Collaborators: Nicolas Blanco, University of Birmingham.

Polycategories are known to give rise to models of classical linear logic in so-called representable polycategories with duals, which ask for the existence of various polymaps satisfying the different universal properties needed to define tensor, par, and negation. We begin by explaining how these different universal properties can all be seen as instances of a single notion of universality of a polymap parameterised by an input or output object, which also generalises the classical notion of universal multimap in a multicategory. We then proceed to introduce a definition of in-cartesian and out-cartesian polymaps relative to a refinement system (= strict functor) of polycategories, in such a way that universal polymaps can be understood as a special case. In particular, we obtain that a polycategory is a representable polycategory with duals if and only if it is bifibred over the terminal polycategory 1. Finally, we present a Grothendieck correspondence between bifibrations of polycategories and pseudofunctors into MAdj, the (weak) 2-polycategory of multivariable adjunctions. When restricted to bifibrations over 1 we get back the correspondence between *-autonomous categories and Frobenius pseudomonoids in MAdj that was recently observed by Shulman.

This work arose in the context of Blanco's doctoral thesis research, and has been published in [10].

7.16 Combinatorial study of linear and planar lambda calculus

Participants: Noam Zeilberger

External Collaborators: Olivier Bodini, LIPN, Université Sorbonne Paris Nord; Alexandros Singh, LIPN, Université Sorbonne Paris Nord

In graph theory, a "map" is another name for a graph embedded on a surface in such a way that the surface is cut up into a collection of simply-connected regions. The study of *map enumeration* has been an active subfield of combinatorics since the pioneering work of Bill Tutte in the 1960s, and quite surprisingly appears to have deep connectionss to the combinatorics of lambda calculus. Notably, bijections between different families of linear lambda terms and different families of maps were independently discovered by Bodini, Gardy, and Jacquot (2013) and by Zeilberger and Giorgetti (2015), and have since been the subject of a variety of followup works (for an overview, see the introduction to Zeilberger, "A theory of linear typings as flows on 3-valent graphs", LICS'2018).

In this work, we dive deeper into the study of the combinatorics of linear lambda calculus, focusing on the analysis of different parameters of lambda terms and their map-theoretic counterparts. For instance, under the bijections mentioned above, *closed subterms* of a linear lambda term correspond to *bridges* in the corresponding map, i.e., edges whose deletion increases the number of connected components. We proved that the limit distribution of the number of closed proper subterms of a random linear lambda term is a Poisson distribution of parameter 1 (= the asymptotic probability of having k closed proper subterms is $1/(k!e)$), therefore allowing us to conclude exactly the same for the limit distribution of the number of bridges in a random map, as a surprising application of lambda calculus to graph theory.

This is work that occurs in the context of Singh's doctoral thesis research, and was presented as a talk at CLA'2020.

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

CIFRE Thesis Inria - Siemens

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

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.

Funding from Nomadic Labs

Participants Gabriel Scherer.

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.
- "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.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Inria associate team not involved in an IIL

COMPRONOM

Title: Combinatorial Proof Normalization

Duration: 2020 - 2022

Coordinator: Lutz Straßburger

Partners:

- Department of Computer Science, University of Bath (United Kingdom)
- Department of Computer Science, University College London, (United Kingdom)

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.

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

9.2 International research visitors

9.2.1 Visits of international scientists

- Nicolas Blanco (PhD visitor from Birmingham, 1 Oct 2020 – 31 Nov 2020, supervised by Noam Zeilberger)

9.3 National initiatives

ANR JCJC project COCA HOLA: Cost Models for Complexity Analyses of Higher-Order Languages, coordinated by B. Accattoli, 2016–2021, ANR-16-CE40-004-01.

10 Dissemination

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

General chair, scientific chair

- With Katarzyna Grygiel (Jagiellonian University, Kraków, Poland), Noam Zeilberger co-organized the 15th Workshop on Computational Logic and Applications (CLA 2020) as a virtual workshop, which took place over two days 12–13 October 2020. Grygiel and Zeilberger also served as co-chairs of the CLA 2020 program committee.
- Kaustuv Chaudhuri was the Conference Chair for the International Joint Conference on Automated Deduction (IJCAR) 2020.
- Dale Miller is the General Chair of the ACM/IEEE Symposium on Logic in Computer Science (LICS).

Member of the organizing committees

- Gabriel Scherer was a co-organizer of the Programming Language Mentoring Workshop (PLMW) affiliated with POPL'21, held on Monday 18th and Tuesday 19th of January 2021.
- Kaustuv Chaudhuri served on the organizing committee for the IJCAR-FSCD 2020 joint meeting. Due to the Covid pandemic this meeting was organized entirely on-line and had nearly 1000 registered participants. Both IJCAR and FSCD broke their attendance records by over a factor of 5.
- Dale Miller is on the Advisory Board of the ACM SigLog (Special Interest Group on Logic and Computation).

Other Participation in the Organization of Scientific Events

- Giti Omidvar was Student Volunteer in for the ICFP'2020 conference
- Marianela Morales was Student Volunteer at ICFP2020 (International Conference on Functional Programming)

10.1.2 Scientific events: selection

Member of the conference program committees

- Noam Zeilberger was a member of the program committee for the following workshops: CLA'2020, TEASE-LP'2020.
- Dale Miller was a member of the program committee for the following meetings: WFLP 2020 (28th International Workshop on Functional and Logic Programming), IJCAR-2020 (10th International Joint Conference on Automated Reasoning), TEASE-LP (Workshop on Trends, Extensions, Applications and Semantics of Logic Programming).

- Beniamino Accattoli was a member of the program committee for the following meetings: LSFA 2020 (The 15th International Workshop on Logical and Semantic Frameworks, with Applications, IWC 2020 (9th International Workshop on Confluence), TERMGRAPH 2020 (11th International Workshop on Computing with Terms and Graphs).

Reviewer

- Lutz Straßburger was reviewer for the following conferences: FoSSACS'2020, CSL'2020, CSL'2021
- Gabriel Scherer was reviewer for the conference CSL'2021, and a member of the POPL'21 Artifact Evaluation Committee.
- Noam Zeilberger was reviewer for FOSSACS'2020.
- Dale Miller was a reviewer LPAR-23 (23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning).
- Marianna Girlando was reviewer for AiML 2020.
- Matteo Manighetti has been reviewer for LPAR-23.
- Beniamino Accattoli has been reviewer for PPDP 2020.

10.1.3 Journal

Member of the editorial boards

- Dale Miller is on the Editorial Board of the Journal of Automated Reasoning published by Springer. Dale Miller is also the Area editor for the Journal of Applied Logic published by Elsevier. since 2003.

Reviewer - reviewing activities

- Lutz Straßburger was reviewer for Bulletin of Symbolic Logic (BSL) and for Logical Methods in Computer Science (LMCS)
- Lutz Straßburger was reviewer for the Book "Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics"
- Noam Zeilberger was reviewer for Theory and Applications of Categories (TAC).
- Marianna Girlando was reviewer for Transactions on Computational Logics (ACM ToCL)
- Matteo Acclavio was reviewer for Studia Logica

10.1.4 Invited talks

- Lutz Straßburger was invited speaker at the Sixth Ticamore Meeting (October 2020, Marseille) https://ticamore.logic.at/marseille2020/
- Lutz Straßburger gave an invited lecture at the Dagstuhl Seminar "20061: SAT and Interactions", Schloss Dagstuhl, Germany, February 2020 https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=20061
- Kaustuv Chaudhuri was an invited speaker at the joint meeting of the Linearity and the Trends in Linear Logic (LINEARITY-TLLA) 2020 workshop, co-located with IJCAR-FSCD 2020.
- Miller was an invited speaker at Tease-LP 2020 (Trends, Extensions, Applications and Semantics of Logic Programming, an ETAPS Workshop) and at ICDCIT (16th International Conference on Distributed Computing and Internet Technology), Bhubaneswar, Odisha, India. 9-12 January 2020.

10.1.5 Leadership within the scientific community

- Kaustuv Chaudhuri is a member of the steering committee of the International Joint Conference on Automated Deduction (IJCAR), term of 2020-2022.

- Dale Miller is a member of the Steering Committee of Certified Programs and Proofs (CPP). He was also a member of the Steering Committee of FSCD until August 2020.

10.1.6 Research administration

- Kaustuv Chaudhuri and Gabriel Scherer form a titulaire-suppléant pair at the "Conseil de Laboratoire" (lab council) of LIX, the research laboratory (and UMR) that Partout belongs to.

- Gabriel Scherer is a member of Saclay's CLHSCT (Comité Local Hygiène, Santé et Conditions de Travail), a committee that has been rather demanding during the COVID-19 outbreak.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Noam Zeilberger was an instructor for the first-year Bachelors course at Ecole Polytechnique, CSE102 (Computer Programming).

- Kaustuv Chaudhuri taught the third year elective course CSE-302, *Compiler Design*, at the Ecole Polytechnique.

- Dale 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 12 hours during Spring and 15 hours during Fall 2020.

- Beniamino 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 12 hours.

- Marianna Girlando was teaching "Logique 2", Université Paris 1 Panthéon-Sorbonne, 26h CM pour Licence 2 en Philosophie, janvier-juin 2020

- Matteo Acclavio was teaching a course at the American University of Paris (54h): Introduction to Computer Programming II

- Giti Omidvar was teaching assistant for the course CSE201 at Ecole Polytechnique (from mid-November 2020 to mid-January 2021)

- Matteo Manighetti was teaching "Bases de la programmation orientée objet" at the IUT d'Orsay, part of Université Paris-Saclay, 84 hours (TD) from January to May 2020

- Matteo Manighetti was teaching "Logique S3", UFR de Philosophie, Université Paris 1 Panthéon-Sorbonne, 22 CM hours (33 équiv. TD) from September to December 2020

- Marianela Morales was Teaching Assistant for the course "Computer Programming" (CSE102) 2019-2020 at École Polytechnique, second semester of Bachelor of Science 1

- Marianela Morales was Teaching Assistant for the course "Computer Programming" at École Polytechnique (CSE102), second semester of Bachelor of Science 1 (2020-2021).

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

10.2.2 Supervision

- We supervised the following Master students in 2020:
  - Francesco Mecca, supervised by Gabriel Scherer (October 2019 – March 2020, M2 from University of Turin)
  - Olivier Martinot, supervised by Gabriel Scherer (March 2020 – August 2020, M2 MPRI)
  - Giti Omidvar, supervised by Lutz Straßburger (March 2020 – August 2020, M2 MPRI)
  - Jui-Hsuan Wu, supervised by Lutz Straßburger (April 2020 – August 2020, M2 MPRI)

- We are supervising the following PhD students:
  - Marianela Morales, supervised by Lutz Straßburger (since October 2020, bourse Polytechnique)
  - Wendlasida Ouedraogo, supervised by Danko Ilik and Lutz Straßburger (since January 2020, CIFRE with Siemens Mobilty)
  - Giti Omidvar, supervised by Lutz Straßburger (since October 2020, CORDI)
  - Olivier Martinot, supervised by Gabriel Scherer (since September 2020, industrial research grant).
  - Nicolas Blanco, supervised by Paul Levy and Noam Zeilberger (since October 2018, University of Birmingham scholarship)
  - Alexandros Singh, supervised by Olivier Bodini and Noam Zeilberger (since September 2019, bourse Ecole Doctorale Galilée)
  - Matteo Manighetti, supervised by Dale Miller (since October 2017, bourse CORDI)
  - Emily Grienenberger supervised jointly by Gilles Dowek and Miller (since October 2019).
  - Maico Carlos Leberle, supervised by Miller and Accattoli (since October 2017).

11 Scientific production

11.1 Major publications

[1] M. Acclavio, R. Horne and L. Straßburger. 'Logic beyond formulas: a proof system on graphs'. In: *LICS 2020 - 35th ACM/IEEE Symposium on Logic in Computer Science*. Saarbrücken, Germany: ACM, July 2020, pp. 38–52. DOI: 10.1145/3373718.3394763. URL: https://hal.inria.fr/hal-02560105.

[2] D. Miller. 'A Distributed and Trusted Web of~Formal Proofs'. In: *ICDCIT 2020 - 16th International Conference on Distributed Computing and Internet Technology*. Bhubaneswar, India, Jan. 2020, pp. 21–40. DOI: 10.1007/978-3-030-36987-3_2. URL: https://hal.inria.fr/hal-02468229.

11.2 Publications of the year

International journals

[3] B. Accattoli, S. Graham-Lengrand and D. Kesner. 'Tight typings and split bounds, fully developed'. In: *Journal of Functional Programming* 30 (2020). DOI: 10.1017/S095679682000012X. URL: https://hal.inria.fr/hal-03089347.

[4] A. Reynaud, G. Scherer and J. Yallop. 'A practical mode system for recursive definitions'. In: *Proceedings of the ACM on Programming Languages* 5.POPL (4th Jan. 2021), pp. 1–29. DOI: 10.1145/3434326. URL: https://hal.inria.fr/hal-03125031.

International peer-reviewed conferences

[5] B. Accattoli, U. Dal Lago and G. Vanoni. 'The Machinery of Interaction'. In: PPDP '20 - 22nd International Symposium on Principles and Practice of Declarative Programming. Bologna, Italy, 8th Sept. 2020, pp. 1–15. DOI: 10.1145/3414080.3414108. URL: https://hal.inria.fr/hal-03089342.

[6] B. Accattoli and A. Díaz-Caro. 'Functional Pearl: The Distributive λ-Calculus'. In: FLOPS 2020 - 15th International Symposium on Functional and Logic Programming. Akita, Japan, 2nd Sept. 2020, pp. 33–49. DOI: 10.1007/978-3-030-59025-3_3. URL: https://hal.inria.fr/hal-03089254.

[7] B. Accattoli, C. Faggian and G. Guerrieri. 'Factorize Factorization'. In: CSL 2021: 29th EACSL Annual Conference on Computer Science Logic. Vol. 183. CSL 2021: 29th EACSL Annual Conference on Computer Science Logic. Ljubljana, Slovenia, 25th Jan. 2021. DOI: 10.4230/LIPIcs.CSL.2021.22. URL: https://hal.archives-ouvertes.fr/hal-03044338.

[8] M. Acclavio, R. Horne and L. Straßburger. 'Logic beyond formulas: a proof system on graphs'. In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany*. LICS 2020 - 35th ACM/IEEE Symposium on Logic in Computer Science. Saarbrücken, Germany, 1st May 2020, pp. 38–52. DOI: 10.1145/3373718.3394763. URL: https://hal.inria.fr/hal-02560105.

[9] M. Acclavio and R. Maieli. 'Generalized connectives for multiplicative linear logic'. In: *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*. CSL 2020 - 28th EACSL annual conference on Computer Science Logic. Vol. 152. Leibniz International Proceedings in Informatics (LIPIcs). Barcelona, Spain, 6th Jan. 2020, 6:1–6:15. DOI: 10.4230/LIPIcs.CSL.2020.6. URL: https://hal.archives-ouvertes.fr/hal-02492258.

[10] N. Blanco and N. Zeilberger. 'Bifibrations of Polycategories and Classical Linear Logic'. In: *Proc. of 36th Conf. on Mathematical Foundations of Programming Semantics, MFPS XXXIV*. MFPS 2020 - 36th Conf. on Mathematical Foundations of Programming Semantics. Vol. 352. Proc. of 34th Conf. on Mathematical Foundations of Programming Semantics, MFPS XXXIV. Paris, France, Oct. 2020, pp. 29–52. DOI: 10.1016/j.entcs.2020.09.003. URL: https://hal.archives-ouvertes.fr/hal-03031058.

[11] D. Catta, L. Pellissier and C. Retoré. 'Inferential Semantics as Argumentative Dialogues'. In: *Distributed Computing and Artificial Intelligence, Special Sessions, 17th International Conference; Distributed Computing and Artificial Intelligence, Special Sessions, 17th International Conference*. DCAI 2020 - 17th International Conference on Distributed Computing and Artificial Intelligence. Vol. AISC. Special Sessions 1242. L´Aquila, Italy, 29th July 2021, pp. 72–81. DOI: 10.1007/978-3-030-53829-3_7. URL: https://hal.archives-ouvertes.fr/hal-02922646.

[12] M. Cimini, D. Miller and J. G. Siek. 'Extrinsically Typed Operational Semantics for Functional Languages'. In: SLE 2020 - 13th ACM SIGPLAN/International Conference on Software Language Engineering. Virtual, United States, 16th Nov. 2020. URL: https://hal.inria.fr/hal-03007256.

[13] M. Girlando and L. Straßburger. 'MOIN: A Nested Sequent Theorem Prover for Intuitionistic Modal Logics (System Description)'. In: *Proceedings of the conference Automated Reasoning - 10th International Joint Conference, IJCAR 2020; Proceedings of the conference Automated Reasoning - 10th International Joint Conference, IJCAR 2020*. IJCAR 2020 - 10th International Joint Conference. Paris, France, 24th June 2020, pp. 398–407. DOI: 10.1007/978-3-030-51054-1_25. URL: https://hal.inria.fr/hal-02457240.

[14] D. Miller. 'A Distributed and Trusted Web of Formal Proofs'. In: ICDCIT 2020 - 16th International Conference on Distributed Computing and Internet Technology. Bhubaneswar, India, 9th Dec. 2020, pp. 21–40. DOI: 10.1007/978-3-030-36987-3_2. URL: https://hal.inria.fr/hal-02468229.

[15] T. Uustalu, N. Veltri and N. Zeilberger. 'Deductive systems and coherence for skew prounital closed categories'. In: *Proc. of 15th Int. Wksh. on Logical Frameworks and Metalanguages: Theory and Practice, LFMTP 2020*. LFMTP 2020 - 15th Int. Wksh. on Logical Frameworks and Metalanguages: Theory and Practice. Vol. 332. Proc. of 15th Int. Wksh. on Logical Frameworks and Metalanguages: Theory and Practice, LFMTP 2020. Paris, France, 2021, pp. 35–53. DOI: 10.4204/eptcs.332.3. URL: https://hal.archives-ouvertes.fr/hal-03031107.

[16] T. Uustalu, N. Veltri and N. Zeilberger. 'Eilenberg-Kelly Reloaded'. In: *Proc. of 36th Conf. on Mathematical Foundations of Programming Semantics, MFPS XXXIV*. MFPS 2020 - 36th Conf. on Mathematical Foundations of Programming Semantics. Vol. 352. Proc. of 34th Conf. on Mathematical Foundations of Programming Semantics, MFPS XXXIV. Paris, France, Oct. 2020, pp. 233–256. DOI: 10.1016/j.entcs.2020.09.012. URL: https://hal.archives-ouvertes.fr/hal-03031068.

[17] T. Uustalu, N. Veltri and N. Zeilberger. 'Proof theory of partially normal skew monoidal categories'. In: *Proc. of 3rd Applied Category Theory Conf., ACT 2020*. ACT 2020 - 3rd Applied Category Theory Conference. Vol. 333. Proc. of 3rd Applied Category Theory Conf., ACT 2020. Cambridge, MA, United States, 2021, pp. 230–246. DOI: 10.4204/eptcs.333.16. URL: https://hal.archives-ouvertes.fr/hal-03031087.

National peer-reviewed Conferences

[18] F. Bour, B. Clément and G. Scherer. 'Tail Modulo Cons'. In: JFLA 2021 - Journées Francophones des Langages Applicatifs. Saint Médard d'Excideuil, France, 6th Apr. 2021. URL: https://hal.inria.fr/hal-03146495.

Conferences without proceedings

[19] M. Girlando, B. Lellmann, N. Olivetti, S. Pesce and G. L. Pozzato. 'Theorem proving for Lewis Logics of Counterfactual Reasoning'. In: CILC 2020 - 35th Edition of the Italian Conference on Computational Logic. Rende / Virtual, Italy: https://cilc2020.demacs.unical.it/, 13th Oct. 2020. URL: https://hal.archives-ouvertes.fr/hal-03080670.

[20] S. Marin and M. Morales. 'Fully structured proof theory for intuitionistic modal logics'. In: AiML 2020 - Advances in Modal Logic. Helsinki, Finland, Aug. 2020. URL: https://hal.archives-ouvertes.fr/hal-03048959.

[21] O. Martinot and G. Scherer. 'Quantified Applicatives: API design for type-inference constraints'. In: ML Family Workshop. Jersey City / Online, United States, 27th Aug. 2020. URL: https://hal.inria.fr/hal-03145040.

[22] F. Mecca and G. Scherer. 'Translation validation of a pattern-matching compiler'. In: ML Family Workshop. New Jersey /Online, United States, 27th Aug. 2020. URL: https://hal.inria.fr/hal-03145030.

Scientific book chapters

[23] T. Uustalu, N. Veltri and N. Zeilberger. 'The sequent calculus of skew monoidal categories (extended version)'. In: *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*. Outstanding Contributions to Logic. 2020. URL: https://hal.archives-ouvertes.fr/hal-03031139.

Reports & preprints

[24] M. Acclavio, R. Horne and L. Straßburger. *An Analytic Propositional Proof System On Graphs*. 23rd Dec. 2020. URL: https://hal.inria.fr/hal-03087392.

[25] R. Blanco, M. Manighetti and D. Miller. *FPC-Coq: Using ELPI to elaborate external proof evidence into Coq proofs*. Inria Saclay, 5th July 2020. URL: https://hal.inria.fr/hal-02974002.

[26] M. Girlando and M. Morales. *MOILab: towards a labelled theorem prover for intuitionistic modal logics*. 9th Dec. 2020. URL: https://hal.archives-ouvertes.fr/hal-03048966.