IN PARTNERSHIP WITH:
CNRS

Ecole Polytechnique

Activity Report 2017

Project-Team PARSIFAL

Proof search and reasoning with logic specifications

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

RESEARCH CENTER
Saclay - Île-de-France

THEME
Proofs and Verification

Table of contents

Project-Team PARSIFAL

Creation of the Project-Team: 2007 July 01

Keywords:

Computer Science and Digital Science:

A2.1.1. - Semantics of programming languages
A2.1.11. - Proof languages
A2.4.2. - Model-checking
A2.4.3. - Proofs
A7.2. - Logic in Computer Science

Other Research Topics and Application Domains:

B9.4.1. - Computer science
B9.4.2. - Mathematics
B9.6. - Reproducibility

1. Personnel

Research Scientists

Dale Miller [Team leader, Inria, Senior Researcher]
Beniamino Accattoli [Inria, Researcher]
Kaustuv Chaudhuri [Inria, Researcher]
Francois Lamarche [Inria, Senior Researcher]
Stéphane Lengrand [CNRS, Researcher]
Gabriel Scherer [Inria, Researcher, from Sep 2017]
Lutz Straßburger [Inria, Researcher]

Post-Doctoral Fellow

Matteo Acclavio [Inria, from Oct 2017]

PhD Students

Roberto Blanco Martinez [Inria]
Andrea Condoluci [Bologna University, from Sep 2017]
Ulysse Gerard [Inria]
Quentin Heath [Inria, until Apr 2017]
Maico Carlos Leberle [Inria, from Oct 2017]
Matteo Manighetti [Inria, from May 2017]
Sonia Marin [Inria, until Oct 2017]

Technical staff

Marco Volpe [Inria]

Interns

Marianela Evelyn Morales Elena [Inria, from Dec 2017]
Riccardo Treglia [Inria, from Mar 2017 until May 2017]

Administrative Assistant

Christine Aklouche [Inria]

Visiting Scientists

Carlos Olarte [Universidade Federal do Rio Grande do Norte, Brazil, Jun 2017]
Elaine Pimentel [Universidade Federal do Rio Grande do Norte, Brazil, Jun 2017]

2. Overall Objectives

2.1. Main themes

The aim of the Parsifal team is to develop and exploit *proof theory* and *type theory* in the specification, verification, and analysis of computational systems.

- *Expertise*: the team conducts basic research in proof theory and type theory. In particular, the team is developing results that help with automated deduction and with the manipulation and communication of formal proofs.
- *Design*: based on experience with computational systems and theoretical results, the team develops new logical principles, new proof systems, and new theorem proving environments.
- *Implementation*: the team builds prototype systems to help validate basic research results.
- *Examples*: the design and implementation efforts are guided by examples of specification and verification problems. These examples not only test the success of the tools but also drive investigations into new principles and new areas of proof theory and type theory.

The foundational work of the team focuses on *structural* and *analytic* proof theory, *i.e.*, the study of formal proofs as algebraic and combinatorial structures and the study of proof systems as deductive and computational formalisms. The main focus in recent years has been the study of the *sequent calculus* and of the *deep inference* formalisms.

An important research question is how to reason about computational specifications that are written in a *relational* style. To this end, the team has been developing new approaches to dealing with induction, co-induction, and generic quantification. A second important question is of *canonicity* in deductive systems, *i.e.*, when are two derivations "essentially the same"? This crucial question is important not only for proof search, because it gives an insight into the structure and an ability to manipulate the proof search space, but also for the communication of *proof objects* between different reasoning agents such as automated theorem provers and proof checkers.

Important application areas currently include:

- Meta-theoretic reasoning on functional programs, such as terms in the λ-calculus
- Reasoning about behaviors in systems with concurrency and communication, such as the π-calculus, game semantics, *etc.*
- Combining interactive and automated reasoning methods for induction and co-induction
- Verification of distributed, reactive, and real-time algorithms that are often specified using modal and temporal logics
- Representing proofs as documents that can be printed, communicated, and checked by a wide range of computational logic systems.
- Development of cost models for the evaluation of proofs and programs.

3. Research Program

3.1. General overview

There are two broad approaches for computational specifications. In the *computation as model* approach, computations are encoded as mathematical structures containing nodes, transitions, and state. Logic is used to *describe* these structures, that is, the computations are used as models for logical expressions. Intensional operators, such as the modals of temporal and dynamic logics or the triples of Hoare logic, are often employed to express propositions about the change in state.

The *computation as deduction* approach, in contrast, expresses computations logically, using formulas, terms, types, and proofs as computational elements. Unlike the model approach, general logical apparatus such as cut-elimination or automated deduction becomes directly applicable as tools for defining, analyzing, and animating computations. Indeed, we can identify two main aspects of logical specifications that have been very fruitful:

- *Proof normalization*, which treats the state of a computation as a proof term and computation as normalization of the proof terms. General reduction principles such as β-reduction or cut-elimination are merely particular forms of proof normalization. Functional programming is based on normalization [70], and normalization in different logics can justify the design of new and different functional programming languages [42].
- *Proof search*, which views the state of a computation as a a structured collection of formulas, known as a *sequent*, and proof search in a suitable sequent calculus as encoding the dynamics of the computation. Logic programming is based on proof search [75], and different proof search strategies can be used to justify the design of new and different logic programming languages [73].

While the distinction between these two aspects is somewhat informal, it helps to identify and classify different concerns that arise in computational semantics. For instance, confluence and termination of reductions are crucial considerations for normalization, while unification and strategies are important for search. A key challenge of computational logic is to find means of uniting or reorganizing these apparently disjoint concerns.

An important organizational principle is structural proof theory, that is, the study of proofs as syntactic, algebraic and combinatorial objects. Formal proofs often have equivalences in their syntactic representations, leading to an important research question about *canonicity* in proofs – when are two proofs "essentially the same?" The syntactic equivalences can be used to derive normal forms for proofs that illuminate not only the proofs of a given formula, but also its entire proof search space. The celebrated *focusing* theorem of Andreoli [44] identifies one such normal form for derivations in the sequent calculus that has many important consequences both for search and for computation. The combinatorial structure of proofs can be further explored with the use of *deep inference*; in particular, deep inference allows access to simple and manifestly correct cut-elimination procedures with precise complexity bounds.

Type theory is another important organizational principle, but most popular type systems are generally designed for either search or for normalization. To give some examples, the Coq system [84] that implements the Calculus of Inductive Constructions (CIC) is designed to facilitate the expression of computational features of proofs directly as executable functional programs, but general proof search techniques for Coq are rather primitive. In contrast, the Twelf system [79] that is based on the LF type theory (a subsystem of the CIC), is based on relational specifications in canonical form (*i.e.*, without redexes) for which there are sophisticated automated reasoning systems such as meta-theoretic analysis tools, logic programming engines, and inductive theorem provers. In recent years, there has been a push towards combining search and normalization in the same type-theoretic framework. The Beluga system [80], for example, is an extension of the LF type theory with a purely computational meta-framework where operations on inductively defined LF objects can be expressed as functional programs.

The Parsifal team investigates both the search and the normalization aspects of computational specifications using the concepts, results, and insights from proof theory and type theory.

3.2. Inductive and co-inductive reasoning

The team has spent a number of years in designing a strong new logic that can be used to reason (inductively and co-inductively) on syntactic expressions containing bindings. This work is based on earlier work by McDowell, Miller, and Tiu [72] [71] [76] [85], and on more recent work by Gacek, Miller, and Nadathur [4] [57]. The Parsifal team, along with our colleagues in Minneapolis, Canberra, Singapore, and Cachen, have been building two tools that exploit the novel features of this logic. These two systems are the following.

- Abella, which is an interactive theorem prover for the full logic.
- Bedwyr, which is a model checker for the "finite" part of the logic.

We have used these systems to provide formalize reasoning of a number of complex formal systems, ranging from programming languages to the λ-calculus and π-calculus.

Since 2014, the Abella system has been extended with a number of new features. A number of new significant examples have been implemented in Abella and an extensive tutorial for it has been written [1].

3.3. Developing a foundational approach to defining proof evidence

The team is developing a framework for defining the semantics of proof evidence. With this framework, implementers of theorem provers can output proof evidence in a format of their choice: they will only need to be able to formally define that evidence's semantics. With such semantics provided, proof checkers can then check alleged proofs for correctness. Thus, anyone who needs to trust proofs from various provers can put their energies into designing trustworthy checkers that can execute the semantic specification.

In order to provide our framework with the flexibility that this ambitious plan requires, we have based our design on the most recent advances within the theory of proofs. For a number of years, various team members have been contributing to the design and theory of *focused proof systems* [47] [50] [51] [52] [59] [68] [69] and we have adopted such proof systems as the corner stone for our framework.

We have also been working for a number of years on the implementation of computational logic systems, involving, for example, both unification and backtracking search. As a result, we are also building an early and reference implementation of our semantic definitions.

3.4. Deep inference

Deep inference [61], [63] is a novel methodology for presenting deductive systems. Unlike traditional formalisms like the sequent calculus, it allows rewriting of formulas deep inside arbitrary contexts. The new freedom for designing inference rules creates a richer proof theory. For example, for systems using deep inference, we have a greater variety of normal forms for proofs than in sequent calculus or natural deduction systems. Another advantage of deep inference systems is the close relationship to categorical proof theory. Due to the deep inference design one can directly read off the morphism from the derivations. There is no need for a counter-intuitive translation.

The following research problems are investigated by members of the Parsifal team:

- Find deep inference system for richer logics. This is necessary for making the proof theoretic results of deep inference accessible to applications as they are described in the previous sections of this report.
- Investigate the possibility of focusing proofs in deep inference. As described before, focusing is a way to reduce the non-determinism in proof search. However, it is well investigated only for the sequent calculus. In order to apply deep inference in proof search, we need to develop a theory of focusing for deep inference.

3.5. Proof nets, atomic flows, and combinatorial proofs

Proof nets graph-like presentations of sequent calculus proofs such that all "trivial rule permutations" are quotiented away. Ideally the notion of proof net should be independent from any syntactic formalism, but most notions of proof nets proposed in the past were formulated in terms of their relation to the sequent calculus. Consequently we could observe features like "boxes" and explicit "contraction links". The latter appeared not only in Girard's proof nets [58] for linear logic but also in Robinson's proof nets [81] for classical logic. In this kind of proof nets every link in the net corresponds to a rule application in the sequent calculus.

Only recently, due to the rise of deep inference, new kinds of proof nets have been introduced that take the formula trees of the conclusions and add additional "flow-graph" information (see e.g., [67][5] leading to the notion of *atomic flow* and [62]. On one side, this gives new insights in the essence of proofs and their normalization. But on the other side, all the known correctness criteria are no longer available.

Combinatorial proofs [65] are another form syntax-independent proof presentation which separates the multiplicative from the additive behaviour of classical connectives.

The following research questions investigated by members of the Parsifal team:

- Finding (for classical and intuitionistic logic) a notion of canonical proof presentation that is deductive, i.e., can effectively be used for doing proof search.
- Studying the normalization of proofs using atomic flows and combinatorial proofs, as they simplify the normalization procedure for proofs in deep inference, and additionally allow to get new insights in the complexity of the normalization.
- Studying the size of proofs use combinatorial proofs.

3.6. Cost Models and Abstract Machines for Functional Programs

In the *proof normalization* approach, computation is usually reformulated as the evaluation of functional programs, expressed as terms in a variation over the λ-calculus. Thanks to its higher-order nature, this approach provides very concise and abstract specifications. Its strength is however also its weakness: the abstraction from physical machines is pushed to a level where it is no longer clear how to measure the complexity of an algorithm.

Models like Turing machines or RAM rely on atomic computational steps and thus admit quite obvious cost models for time and space. The λ-calculus instead relies on a single non-atomic operation, β-reduction, for which costs in terms of time and space are far from evident.

Nonetheless, it turns out that the number of β-steps is a reasonable time cost model, i.e.,it is polynomially related to those of Turing machines and RAM. For the special case of *weak evaluation* (i.e., reducing only β-steps that are not under abstractions)—which is used to model functional programming languages—this is a relatively old result due to Blelloch and Greiner [48] (1995). It is only very recently (2014) that the strong case—used in the implementation models of proof assistants—has been solved by Accattoli and Dal Lago [43].

With the recent recruitment of Accattoli, the team's research has expanded in this direction. The topics under investigations are:

1. *Complexity of Abstract Machines*. Bounding and comparing the overhead of different abstract machines for different evaluation schemas (weak/strong call-by-name/value/need λ-calculi) with respect to the cost model. The aim is the development of a complexity-aware theory of the implementation of functional programs.
2. *Reasonable Space Cost Models*. Essentially nothing is known about reasonable space cost models. It is known, however, that environment-based execution model—which are the mainstream technology for functional programs—do not provide an answer. We are exploring the use of the non-standard implementation models provided by Girard's Geometry of Interaction to address this question.

4. Application Domains

4.1. Integrating a model checker and a theorem prover

The goal of combining model checking with inductive and co-inductive theorem is appealing. The strengths of systems in these two different approaches are strikingly different. A model checker is capable of exploring a finite space automatically: such a tool can repeatedly explore all possible cases of a given computational space. On the other hand, a theorem prover might be able to prove abstract properties about a search space. For example, a model checker could attempt to discover whether or not there exists a winning strategy for, say, tic-tac-toe while an inductive theorem prover might be able to prove that if there is a winning strategy for one board then there is a winning strategy for any symmetric version of that board. Of course, the ability to combine proofs from these systems could drastically reduce the amount of state exploration and verification of proof certificates that are needed to prove the existence of winning strategies.

Our first step to providing an integration of model checking and (inductive) theorem proving was the development of a strong logic, that we call $\mathcal{G}$, which extends intuitionistic logic with notions of least and greatest fixed points. We had developed the proof theory of this logic in earlier papers [4] [57]. We have now recently converted the Bedwyr system so that it formally accepts almost all definitions and theorem statements that are accepted by the inductive theorem prover Abella. Thus, these two systems are proving theorems in the same logic and their results can now be shared.

Bedwyr's tabling mechanism has been extended so that its it can make use of previously proved lemmas. For instance, when trying to prove that some board position has a winning strategy, an available stored lemma can now be used to obtain the result if some symmetric board position is already in the table.

Heath and Miller have shown how model checking can be seen as constructing proof in (linear) logic [64]. For more about recent progress on providing checkable proof certificates for model checking, see the web site for Bedwyr http://slimmer.gforge.inria.fr/bedwyr/.

4.2. Implementing trusted proof checkers

Traditionally, theorem provers—whether interactive or automatic—are usually monolithic: if any part of a formal development was to be done in a particular theorem prover, then the whole of it would need to be done in that prover. Increasingly, however, formal systems are being developed to integrate the results returned from several, independent and high-performance, specialized provers: see, for example, the integration of Isabelle with an SMT solver [56] as well as the Why3 and ESC/Java systems.

Within the Parsifal team, we have been working on foundational aspects of this multi-prover integration problem. As we have described above, we have been developing a formal framework for defining the semantics of proof evidence. We have also been working on prototype checkers of proof evidence which are capable of executing such formal definitions. The proof definition language described in the papers [54], [53] is currently given an implementation in the λProlog programming language [74]. This initial implementation will be able to serve as a "reference" proof checker: others who are developing proof evidence definitions will be able to use this reference checker to make sure that they are getting their definitions to do what they expect.

Using λProlog as an implementation language has both good and bad points. The good points are that it is rather simple to confirm that the checker is, in fact, sound. The language also supports a rich set of abstracts which make it impossible to interfere with the code of the checker (no injection attacks are possible). On the negative side, the performance of our λProlog interpreters is lower than that of specially written checkers and kernels.

4.3. Trustworthy implementations of theorem proving techniques

Instead of integrating different provers by exchanging proof evidence and relying on a backend proof-checker, another approach to integration consists in re-implementing the theorem proving techniques as proof-search strategies, on an architecture that guarantees correctness.

Inference systems in general, and focused sequent calculi in particular, can serve as the basis of such an architecture, providing primitives for the exploration of the search space. These form a trusted *Application Programming Interface* that can be used to program and experiment various proof-search heuristics without worrying about correctness. No proof-checking is needed if one trusts the implementation of the API.

This approach has led to the development of the Psyche engine, and to its latest branch CDSAT.

Three major research directions are currently being explored, based on the above:

- The first one is about formulating automated reasoning techniques in terms of inference systems, so that they fit the approach described above. While this is rather standard for technique used in first-order Automated Theorem Provers (ATP), such as resolution, superposition, etc, this is much less standard in SMT-solving, the branch of automated reasoning that can natively handle reasoning in a combination of mathematical theories: the traditional techniques developed there usually organise the collaborations between different reasoning black boxes, whose opaque mechanisms less clearly

connect to proof-theoretical inference systems. We are therefore investigating new foundations for reasoning in combinations of theories, expressed as fine-grained inference systems, and developed the *Conflict-Driven Satisfiability framework* for these foundations [19].

- The second one is about understanding how to deal with quantifiers in presence of one or more theories: On the one hand, traditional techniques for quantified problems, such as *unification* [41] or *quantifier elimination* are usually designed for either the empty theory or very specific theories. On the other hand, the industrial techniques for combining theories (Nelson-Oppen, Shostak, MCSAT [78], [82], [86], [66]) are designed for quantifier-free problems, and quantifiers there are dealt with incomplete *clause instantiation* methods or *trigger*-based techniques [55]. We are working on making the two approaches compatible.
- The above architecture's modular approach raises the question of how its different modules can safely cooperate (in terms of guaranteed correctness), while some of them are trusted and others are not. The issue is particularly acute if some of the techniques are run concurrently and exchange data at unpredictable times. For this we explore new solutions based on Milner's *LCF* [77]. In [60], we argued that our solutions in particular provide a way to fulfil the "Strategy Challenge for SMT-solving" set by De Moura and Passmore [87].

5. New Software and Platforms

5.1. Abella

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.

- Participants: Dale Miller, Gopalan Nadathur, Kaustuv Chaudhuri, Mary Southern, Matteo Cimini, Olivier Savary-Bélanger and Yuting Wang
- Partner: Department of Computer Science and Engineering, University of Minnesota
- Contact: Kaustuv Chaudhuri
- URL: http://abella-prover.org/

5.2. Bedwyr

Bedwyr - A proof search approach to model checking
FUNCTIONAL DESCRIPTION: Bedwyr is a generalization of logic programming that allows model checking directly on syntactic expressions that possibly contain bindings. This system, written in OCaml, is a direct implementation of two recent advances in the theory of proof search.

It is possible to capture both finite success and finite failure in a sequent calculus. Proof search in such a proof system can capture both may and must behavior in operational semantics. Higher-order abstract syntax is directly supported using term-level lambda-binders, the nabla quantifier, higher-order pattern unification, and explicit substitutions. These features allow reasoning directly on expressions containing bound variables.

The distributed system comes with several example applications, including the finite pi-calculus (operational semantics, bisimulation, trace analyses, and modal logics), the spi-calculus (operational semantics), value-passing CCS, the lambda-calculus, winning strategies for games, and various other model checking problems.

- Participants: Dale Miller, Quentin Heath and Roberto Blanco Martinez
- Contact: Quentin Heath
- URL: http://slimmer.gforge.inria.fr/bedwyr/

5.3. Checkers

Checkers - A proof verifier

KEYWORDS: Proof - Certification - Verification
FUNCTIONAL DESCRIPTION: Checkers is a tool in Lambda-prolog for the certification of proofs. Checkers consists of a kernel which is based on LKF and is based on the notion of ProofCert.

- Participants: Giselle Machado Nogueira Reis, Marco Volpe and Tomer Libal
- Contact: Tomer Libal
- URL: https://github.com/proofcert/checkers

5.4. Psyche

Proof-Search factorY for Collaborative HEuristics
FUNCTIONAL DESCRIPTION: Psyche is a modular platform for automated or interactive theorem proving, programmed in OCaml and built on an architecture (similar to LCF) where a trusted kernel interacts with plugins. The kernel offers an API of proof-search primitives, and plugins are programmed on top of the API to implement search strategies. This architecture is set up for pure logical reasoning as well as for theory-specific reasoning, for various theories.

RELEASE FUNCTIONAL DESCRIPTION: It is now equipped with the machinery to handle quantifiers and quantifier-handling techniques. Concretely, it uses meta-variables to delay the instantiation of existential variables, and constraints on meta-variables are propagated through the various branches of the search-space, in a way that allows local backtracking. The kernel, of about 800 l.o.c., is purely functional.

- Participants: Assia Mahboubi, Jean-Marc Notin and Stéphane Graham-Lengrand
- Contact: Stéphane Graham-Lengrand
- URL: http://www.lix.polytechnique.fr/~lengrand/Psyche/

6. New Results

6.1. Separating Functional Computation from Relations

Participants: Ulysse Gérard, Dale Miller.

The logical foundation of arithmetic generally starts with a quantificational logic over relations. Of course, one often wishes to have a formal treatment of functions within this setting. Both Hilbert and Church added choice operators (such as the epsilon operator) to logic in order to coerce relations that happen to encode functions into actual functions. Others have extended the term language with confluent term rewriting in order to encode functional computation as rewriting to a normal form (e.g., the Dedukti proof checking project [46]) It is possible to take a different approach that does not extend the underlying logic with either choice principles or with an equality theory. Instead, we use the familiar two-phase construction of focused proofs and capture functional computation entirely within one of these phases. As a result, computation of functions can remain purely relational even when it is computing functions. This result, which appearred in [22], could be used to add to the Abella theorem prover a primitive method for doing deterministic computations.

6.2. Translating between implicit and explicit versions of proof

Participants: Roberto Blanco, Zakaria Chihani, Dale Miller.

As we have demonstrated within the Parsifal team, the Foundational Proof Certificate (FPC) framework can be used to define the semantics of a wide range of proof evidence. We have given such definitions for a number of textbook proof systems as well as for the proof evidence output from some existing theorem proving systems. An important decision in designing a proof certificate format is the choice of how many details are to be placed within certificates. Formats with fewer details are smaller and easier for theorem provers to output but they require more sophistication from checkers since checking will involve some proof reconstruction. Conversely, certificate formats containing many details are larger but are checkable by less sophisticated checkers. Since the FPC framework is based on well-established proof theory principles, proof certificates can be manipulated in meaningful ways. In fact, we have shown how it is possible to automate moving from implicit to explicit (*elaboration*) and from explicit to implicit (*distillation*) proof evidence via the proof checking of a *pair of proof certificates*. Performing elaboration makes it possible to transform a proof certificate with details missing into a certificate packed with enough details so that a simple kernel (without support for proof reconstruction) can check the elaborated certificate. This design allows us to trust in only a single, simple checker of explicitly described proofs but trust in a range of theorem provers employing a range of proof structures. Experimental results of using this design appear in

6.3. Combinatorial Flows

Participant: Lutz Straßburger.

Combinatorial flows are a variation of combinatorial proofs that allow for the substitution of proofs into proofs (instead of just substituting formulas). This makes combinatorial flows p-equvalent to Frege systems with substitution, which are the strongest proof systems with respect to p-simulation, as studied in proof complexity. Since combinatorial flows have a polynomial correctness criterion, they can also be seen as an improvement to atomic flows (which do not have a correctness criterion). This work has been presented at the FCSD 2017 conference [37], [28]

6.4. Justification Logic for Constructive Modal Logic

Participants: Lutz Straßburger, Sonia Marin.

Justification logic is a family of modal logics generalizing the Logic of Proofs LP, introduced by Artemov in [45]. The original motivation, which was inspired by works of Kolmogorov and Gödel in the 1930's, was to give a classical provability semantics to intuitionistic propositional logic. The language of the Logic of Proofs can be seen as a modal language where occurrences of the $\Box$-modality are replaced with terms, also known as *proof polynomials*, *evidence terms*, or *justification terms*, depending on the setting. The intended meaning of the formula '$t : A$' is 't *is a proof of* A' or, more generally, the reason for the validity of A. Thus, the justification language is viewed as a refinement of the modal language, with one provability construct $\Box$ replaced with an infinite family of specific proofs. In a joint work with Roman Kuznets (TU Wien), we add a second type of terms, which we call *witness terms* and denote by Greek letters. Thus, a formula $\diamond A$ is to be realized by '$\mu : A$'. The intuitive understanding of these terms is based on the view of $\diamond$ modality as representing consistency (with $\Box$ still read as provability). The term μ justifying the consistency of a formula is viewed as an abstract witnessing model for the formula. We keep these witnesses abstract so as not to rely on any specific semantics. All the operations on witness terms that we employ to ensure the realization theorem for CK, CD, CT, and $CS4$. This work has been presented at the IMLA 2017 workshop [40]

6.5. Proof Theory of Indexed Nested Sequents

Participants: Lutz Straßburger, Sonia Marin.

Indexed nested sequents are an extension of nested sequents allowing a richer underlying graph-structure that goes beyond the plain tree-structure of pure nested sequents. For this reason they can be used to give deductive systems to modal logics which cannot be captured by pure nested sequents. In this work we show how the standard cut-elimination procedure for nested sequents can be extended to indexed nested sequents, and we discuss how indexed nested sequents can be used for intuitionistic modal logics. These results have been presented at the TABLEAUX 2017 conference [24], [35]

6.6. On the Length of Medial-Switch-Mix Derivations

Participant: Lutz Straßburger.

Switch and medial are two inference rules that play a central role in many deep inference proof systems. In specific proof systems, the mix rule may also be present. In a joint work with Paola Bruscoli (University of Bath) we show that the maximal length of a derivation using only the inference rules for switch, medial, and mix, modulo associativity and commutativity of the two binary connectives involved, is quadratic in the size of the formula at the conclusion of the derivation. This shows, at the same time, the termination of the rewrite system. This result has been presented at the International Workshop on Logic, Language, Information, and Computation 2017 [20].

6.7. Maehara-style Modal Nested Calculi

Participant: Lutz Straßburger.

In a joint work with Roman Kuznets (TU Wien), we develop multi-conclusion nested sequent calculi for the fifteen logics of the intuitionistic modal cube between IK and IS5. The proof of cut-free completeness for all logics is provided both syntactically via a Maehara-style translation and semantically by constructing an infinite birelational countermodel from a failed proof search [83]. Interestingly, the Maehara-style translation for proving soundness syntactically fails due to the hierarchical structure of nested sequents. Consequently, we only provide the semantic proof of soundness. The countermodel construction used to prove completeness required a completely novel approach to deal with two independent sources of non-termination in the proof search present in the case of transitive and Euclidean logics.

6.8. Combining inference systems in the CDSAT framework

Participant: Stéphane Graham-Lengrand.

In 2016 we had designed a methodology [49], based on *inference systems*, for combining theories in SMT-solving, that supersedes the existing approaches, namely that of Nelson-Oppen [78] and that of MCSAT [86], [66]. While soundness and completeness of our approach were proved in 2016, we further developed, in 2017, the meta-theory of this system, now called CDSAT for *Conflict-Driven Satisfiability*, in particular with

- a proof of termination for the CDSAT system, and the identification of sufficient conditions, on the theory modules to be combined, for the global termination of the system to hold;
- a learning mechanism, whereby the system discovers lemmas along the run, which can be used later to speed-up the rest of the run;
- an enrichment of the CDSAT system with proof-object generation, and the identification of proof-construction primitives that can be used to make the answers produced by CDSAT correct-by-construction.

The first result, together with the introduction of the CDSAT framework, was publishing this year in [19]. The last two results are described in a paper accepted for publication at CPP in 2018.

6.9. Theory modules for CDSAT

Participant: Stéphane Graham-Lengrand.

The CDSAT system described above is a framework for the combination of theory modules, so it is only useful inasmuch many theories can be captured as CDSAT theory modules. Theory modules are essentially given by a set of inference rules and, for each input problem, a finite set of expressions that are allowed to be used by CDSAT at runtime. These ingredients need to satisfy some requirement for soundness, completess, and termination of CDSAT. In 2017 we identified such theory modules for the following theories

- Boolean logic;
- Linear Rational Arithmetic;
- Equality with Uninterpreted Function symbols;
- Any theory whose ground satisfiability is decidable, if one is willing to give up the fine-grained aspect of inference rules;
- Bitvectors (core fragment).

The first four cases of theories were published in [19], while the Bitvector theory was published in [21].

6.10. Environments and the Complexity of Abstract Machines

Participant: Beniamino Accattoli.

This joint work with Bruno Barras (Inria) [30] belongs to line of work *Cost Models and Abstract Machines for Functional Languages*, supported by the ANR project COCA HOLA.

We study various notions of environments (local, global, split) for abstract machines for functional languages, from a complexity and implementative point of view.

An environment is a data structure used to implement sharing of subterms. There are two main styles. The most common one is to have many local environments, one for every piece of code in the data structures of the machine. A minority of works uses a single global environment instead. Up to now, the two approaches have been considered equivalent, in particular at the level of the complexity of the overhead: they have both been used to obtain bilinear bounds, that is, linear in the number of beta steps and in the size of the initial term.

Our main result is that local environments admit implementations that are asymptotically faster than global environments, lowering the dependency from the size of the initial term from linear to logarithmic, thus improving the bounds in the literature. We also show that a third style, split environments, that are in between local and global ones, has the benefits of both. Finally, we provide a call-by-need machine with split environments for which we prove the new improved bounds on the overhead.

6.11. The Negligible and Yet Subtle Cost of Pattern Matching

Participant: Beniamino Accattoli.

This joint work with Bruno Barras (Inria) [31] belongs to line of work *Cost Models and Abstract Machines for Functional Languages*, supported by the ANR project COCA HOLA.

In this work we extend results about time cost models for the λ-calculus to a larger language, namely the λ-calculus with constructors and pattern matching. We consider all natural evaluation strategies, that is, call-by-name, call-by-value, and call-by-need.

The results are expected, and considered folklore, but we show that the question is subtler than it seems at first sight, by exhibiting some counter-example for naive formulations of the extensions. The, we show the actual results for the right extensions.

6.12. Implementing Open Call-by-Value

This joint work with Giulio Guerrieri (Oxford University) [32] belongs to line of work *Cost Models and Abstract Machines for Functional Languages*, supported by the ANR project COCA HOLA.

The theory of the call-by-value λ-calculus relies on weak evaluation and closed terms, that are natural hypotheses in the study of programming languages. To model proof assistants, however, strong evaluation and open terms are required. Open call-by-value is the intermediate setting of weak evaluation with open terms, on top of which Grégoire and Leroy designed the abstract machine of Coq. This paper provides a theory of abstract machines for open call-by-value. The literature contains machines that are either simple but inefficient, as they have an exponential overhead, or efficient but heavy, as they rely on a labelling of environments and a technical optimization. We introduce a machine that is simple and efficient: it does not use labels and it implements open call-by-value within a bilinear overhead. Moreover, we provide a new fine understanding of how different optimizations impact on the complexity of the overhead.

6.13. Further Formalizing the Meta-Theory of Linear Logic

Participants: Kaustuv Chaudhuri, Leonardo Lima, Giselle Reis.

We have continued our formalization of the meta-theory of substructural logics by giving a fully formal proof of cut-elimination (and hence of completeness) for focused classical first-order linear logic. This is the first time that this complete system has had a fully formalized proof.

This formalization serves as a *tour de force* of Abella's ability to reason about mutual induction and support sophisticated binding constructs.

An extended invited paper is currently under review, to possibly appear in a special issue of *Theoretical Computer Science* in 2018.

6.14. Formalized Meta-Theory of Simultaneous Substitutions

Participant: Kaustuv Chaudhuri.

It has long been claimed that a logical framework must have sophisticated built-in support for reasoning about formal substitutions in order to formalize relational meta-theorems such as strong normalization (using a logical relations style argument) or that applicative simulation is a pre-congruence. A number of type-theoretic frameworks in recent years, such as Beluga, have indeed started to incorporate such constructs in their core systems.

We have recently shown how to implement the meta-theory of simultaneous substitutions in the Abella system without any modification or extension of the (trusted) kernel, and without sacrificing any expressivity. The results of this paper will appear in the ACM Conference on Certified Programming in Jannuary 2018.

Our hope is that this work will be continued in the near future to build a specification language based on contextual LF in Abella, similar to how Abella/LF handles (ordinary) LF.

6.15. Hybrid Linear Logic Revisited

Participants: Kaustuv Chaudhuri, Joëlle Despeyroux, Carlos Olarte, Elaine Pimentel.

We have written a comprehensive account of hybrid linear logic (HyLL) and its relation to a number of related linear logic variants such as subexponential logic. One of the new and novel examples that we have fully worked out is how to encode CTL and CTL* in HyLL, which shows that HyLL can indeed serve as a logical framework for representing and reasoning about constrained transition systems, such as biochemical networks.

This account will appear in a special issue of MSCS in 2018.

6.16. Correctness of Speculative Optimizations with Dynamic Deoptimization

Participant: Gabriel Scherer.

This joint work with Olivier Flückiger, Ming-Ho Yee Ming-Ho, Aviral Goel, Amal Ahmed and Jan Vitek was initiated during Gabriel Scherer's post-doctoral stay at Northeastern University, Boston, USA.

Practitioners from the software industry find it difficult to implement Just-In-Time (JIT) compilers for dynamic programming languages, such as Javascript: they don't know how to reason on the correctness of their optimizations in the context of Just-In-Time code generation and deoptimization. We explain how to adapt reasoning approaches and proof techniques from standard compiler research to this new setting.

This work [14] will appear in POPL 2018.

7. Partnerships and Cooperations

7.1. European Initiatives

7.1.1. FISP: ANR blanc International

Participants: Kaustuv Chaudhuri, François Lamarche, Sonia Marin, Dale Miller, Lutz Straßburger.

Title: The Fine Structure of Formal Proof Systems and their Computational Interpretations

Duration: 01/01/2016 – 31/10/2019

Partners:

University Paris VII, PPS (PI: Michel Parigot)

Inria Saclay–IdF, EPI Parsifal (PI: Lutz Straßburger)

University of Innsbruck, Computational Logic Group (PI: Georg Moser)

Vienna University of Technology, Theory and Logic Group (PI: Matthias Baaz)

Total funding by the ANR: 316 805 EUR

The FISP project is part of an ambitious, long-term project whose objective is to apply the powerful and promising techniques from structural proof theory to central problems in computer science for which they have not been used before, especially the understanding of the computational content of proofs, the extraction of programs from proofs and the logical control of refined computational operations. So far, the work done in the area of computational interpretations of logical systems is mainly based on the seminal work of Gentzen, who in the mid-thirties introduced the sequent calculus and natural deduction, along with the cut-elimination procedure. But that approach shows its limits when it comes to computational interpretations of classical logic or the modelling of parallel computing. The aim of our project, based on the complementary skills of the teams, is to overcome these limits. For instance, deep inference provides new properties, namely full symmetry and atomicity, which were not available until recently and opened new possibilities at the computing level, in the era of parallel and distributed computing.

7.1.2. COCA HOLA: ANR JCJC Project

Participant: Beniamino Accattoli.

Title: COst model for Complexity Analyses of Higher-Order programming LAnguages.

Collaborators: Ugo Dal Lago (University of Bologna & Inria), Delia Kesner (Paris Diderot University), Damiano Mazza (CNRS & Paris 13 University), Claudio Sacerdoti Coen (University of Bologna).

Duration: 01/10/2016 – 31/09/2019

Total funding by the ANR: 155 280 EUR

The COCA HOLA project aims at developing complexity analyses of higher-order computations, i.e. that approach to computation where the inputs and outputs of a program are not simply numbers, strings, or compound data-types, but programs themselves. The focus is not on analysing fixed programs, but whole programming languages. The aim is the identification of adequate units of measurement for time and space, i.e. what are called reasonable cost models. The problem is non-trivial because the evaluation of higher-order languages is defined abstractly, via high-level operations, leaving the implementation unspecified. Concretely, the project will analyse different implementation schemes, measuring precisely their computational complexity with respect to the number of high-level operations, and eventually develop more efficient new ones. The goal is to obtain a complexity-aware theory of implementations of higher-order languages with both theoretical and practical downfalls.

The projects stems from recent advances on the theory of time cost models for the lambda-calculus, the computational model behind the higher-order approach, obtained by the principal investigator and his collaborators (who are included in the project).

COCA HOLA will span over three years and is organised around three work packages, essentially:

1. extending the current results to encompass realistic languages;
2. explore the gap between positive and negative results in the literature;
3. use ideas from linear logic to explore space cost models, about which almost nothing is known.

7.2. International Initiatives

7.2.1. Participation in Other International Programs

7.2.1.1. PHC Amadeus: Analytic Calculi for Modal Logics

Participants: Kaustuv Chaudhuri, Sonia Marin, Giselle Reis, Lutz Straßburger.

Title: Analytic Calculi for Modal Logics

Duration: 01/01/2016 – 31/12/2017

Austrian Partner: TU Wien, Institute for Computer Science (Department III)

Modal logics are obtained from propositional logics by adding modalities $\Box$ and $\diamond$, meaning necessity and possibility. Originally studied by philosophers in order to reason about knowledge and belief, modal logics have nowadays many applications in computer science. Well known examples are epistemic logics, which allow to formally reason about the knowledge of independently acting and interacting agents, temporal logics, which allow to reason about temporal properties of processes, and authentication logics, which are used to formally reason about authentication protocols.

The purpose of this project is to develop a proof theory for variants of modal logic that have applications in modern computer science but that have been neglected by traditional proof theory so far.

7.3. International Research Visitors

7.3.1. Visits of International Scientists

7.3.1.1. Internships

Riccardo Treglia was an intern funded by COCA HOLA during March, April, and May 2017. He was advised by Accattoli and worked on the complexity analysis of abstract machines for the λ-calculus.

7.3.2. Visits to International Teams

7.3.2.1. Research Stays Abroad

Stéphane Graham-Lengrand spent 8 months, from January 2017 to August 2017, at SRI International, Computer Science Lab. This visit developed a collaboration with N. Shankar, MP Bonacina, and D. Jovanovic, on new algorithms and new architectures for automated and interactive theorem proving, as well as on new programme verification techniques.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Organisation

8.1.1.1. General Chair, Scientific Chair

D. Miller has been selected to be the LICS General Chair for three years starting in July 2018.

8.1.1.2. Member of the Organizing Committees

Lutz Straßburger was member of the organizing committee for the second FISP meeting in Paris

8.1.2. Scientific Events Selection

8.1.2.1. Chair of Conference Program Committees

D. Miller was the Program Committee chair for the FSCD'17: Second International Conference on Formal Structures for Computation and Deduction, Oxford, 3-6 September.

D. Miller was on the Steering Committee for the FSCD series of International Conference on Formal Structures for Computation and Deduction.

D. Miller was a member of the jury for selecting the 2017 Ackermann Award (the EACSL award for outstanding doctoral dissertation in the field of Logic in Computer Science).

D. Miller was a member of the 2012, 2016, and 2017 Herbrand Award Committee of the Association for Automated Reasoning.

D. Miller is also a member of the SIGLOG advisory board, starting November 2015.

B. Accattoli was one of the two Program Committee chairs of the 6th International Workshop on Confluence (IWC 2017).

K. Chaudhuri as a co-chair of the Program Committee for the workshop on Structures and Deduction, co-located with FSCD.

8.1.2.2. Member of the Conference Program Committees

D. Miller was on the Program Committee of the following international conferences.

- 26th International Conference on Automated Deduction, Gothenburg, Sweden, 6-11 August.

B. Accattoli was on the Program Committee of the following international workshops.

- LOLA 2017: Syntax and Semantics of Low-Level Languages, Reykjavik, Iceland, 19 June.
- WPTE 2017: 4th Workshop on Rewriting Techniques for Program Transformations and Evaluation, Oxford, UK, 8 September.
- DICE-FOPARA 2017: 8th Workshop on Developments in Implicit Computational complExity and 5th Workshop on Foundational and Practical Aspects of Resource Analysis, Uppsala, Sweden, 22–23 April.

S. Graham-Lengrand was on the Program Committee of the following international workshops.

- AFM 2017: Automated Formal Methods, Menlo Park, USA, 19 May.
- PxTP 2017: 5th Workshop on Proof eXchange for Theorem Proving, Brasilia, Brazil, 4 September.

K. Chaudhuri was on the Program Committee of the following international workshops.

- LFMTP 2017: Logical Frameworks and Meta-languages: Theory and Practice, Oxford, U.K.
- LSFA 2017: Logical and Semantic Frameworks with Applications, Brasilia, Brazil

G. Scherer was on the Program Committee of the following international conference.

- Trends in Functional Programming, University of Kent at Canterbury, UK, 19-21 June.

8.1.2.3. Reviewer

- Lutz Straßburger reviewed submissions for the following conferences: LICS 2017, LPAR-21, FoSSaCS 2018, LFCS 2018
- B. Accattoli was a reviewer for the international conferences LICS 2017 (twice) and FSCD 2017.
- F. Lamarche was reviewer for CSL 2017.
- S. Graham-Lengrand was a reviewer for the international conferences LICS 2017 (three times), CADE 2017, AFM 2017, CSL 2017, TYPES 2017, PxTP 2017, FOSSACS 2018.
- G. Scherer reviewed submissions for the following conferences: JFLA 2018, FoSSaCS 2018, as well as for the PriSC 2018 (Principle of Secure Compilation) workshop.

8.1.3. Journal

8.1.3.1. Member of the Editorial Boards

D. Miller is on the editorial board of the following journals: ACM Transactions on Computational Logic, Journal of Automated Reasoning (Springer), and Journal of Applied Logic (Elsevier).

K. Chaudhuri served as a guest editor for a special issue of Mathematical Structures of Computer Science devoted to Logical Frameworks.

8.1.3.2. Reviewer - Reviewing Activities

- Lutz Straßburger did reviewing work for the following journals: Journal of Applied Logic (JAL), Studia Logica, Mathematical Structures in Computer Science (MSCS), Logical Methods in Computer Science (LMCS), Journal of Logic and Computation (JLC), Journal of Automated Reasoning (JAR).
- B. Accattoli was a reviewer for the international journals Transactions on Computational Logic (TOCL, ACM), Mathematical Structures in Computer Science (MSCS, Cambridge University Press), Logical Methods in Computer Science (LMCS), Journal of Automated Reasoning (JAR, Springer), Annals of Pure and Applied Logic (APAL, Elsevier).
- S. Graham-Lengrand was a reviewer for the following international journals: Theory of Computing Systems (TOCS), Annals of Pure and Applied Logic (APAL), Mathematical Structures in Computer Science (MSCS), Logical Methods in Computer Science (LMCS), Journal of Automated Reasoning (JAR), Bulletin of Symbolic Logic (BSL).
- G. Scherer was a reviewer for the international journal Mathematical Structures in Computer Science (MSCS).

8.1.4. Invited Talks

- D. Miller gave invited talks at the following two regularly held international meetings.
  - LAP 2017: Sixth Conference on Logic and Applications, 18-22 September 2018, Dubrovnik, Croatia.
  - PADL 2017: Nineteenth International Symposium on Practical Aspects of Declarative Languages, 16-17 January 2017, Paris.
- Lutz Straßburger gave an invited talk at the 4th International Workshop on Structures and Deduction (SD 2017), affiliated with FSCD'17.
- B. Accattoli gave an invited talk at LSFA 2017, the 12th Workshop on Logical and Semantic Frameworks with Applications, Brasilia, Brazil, 23-24 September.
- S. Graham-Lengrand gave an invited talk at CSLI 2017, the 6th CSLI Workshop on Logic, Rationality & Intelligent Interaction, University of Stanford, Palo Alto, USA, 3-4 June.

8.1.5. Research Administration

L. Straßburger serves on the "commission développement technologique (CDT)" for Inria Saclay–Île-de-France (since June 2012).

F. Lamarche was "responsable de centre" Saclay – Ile de France for Raweb.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Master: D. Miller, "*MPRI 2-1: Logique linéaire et paradigmes logiques du calcul*", 12 hours, M2, Master Parisien de Recherche en Informatique, France.

Lutz Straßburger gave a course on "Efficient Proof Systems for Modal Logics" at ESSLLI 2017 (joint with Roman Kuznets, TU Wien)

Master: B. Accattoli, "*MPRI 2-1: Logique linéaire et paradigmes logiques du calcul*", 9 hours, M2, Master Parisien de Recherche en Informatique, France.

B. Accattoli taught the mini-course *the complexity of β-reduction*, 3 hours, at the International School on Rewriting 2017, Eindhoven, The Netherlands, 3-7 July.

Licence: S. Graham-Lengrand, "*INF412: Fondements de l'Informatique: Logique, Modèles, Calcul*", 32 hours eq. TD, L3, École Polytechnique, France.

Master: S. Graham-Lengrand, "*INF551: Computational Logic*", 45 hours eq. TD, M1, École Polytechnique, France.

Licence: K. Chaudhuri, "*INF431*: Concurrence" and "*INF441*: Programmation avancée", 80 hours eq. TD, L2, Ecole polytechnique, France.

8.2.2. Supervision

PhD in progress: Sonia Marin, 1 Nov 2014, supervised by L. Straßburger and D. Miller

PhD in progress: Roberto Blanco, Ulysse Gérard, and Matteo Manighetti, supervised by D. Miller

PhD in progress: François Thiré (since 1st October 2016), supervised by S. Graham-Lengrand (joint with G. Dowek)

8.2.3. Juries

D. Miller was a reporter for the habilitation of Olivier Hermant, 20 April 2017.

9. Bibliography

Major publications by the team in recent years

[1] D. BAELDE, K. CHAUDHURI, A. GACEK, D. MILLER, G. NADATHUR, A. TIU, Y. WANG. *Abella: A System for Reasoning about Relational Specifications*, in "Journal of Formalized Reasoning", 2014, vol. 7, n° 2, pp. 1-89 [*DOI :* 10.6092/ISSN.1972-5787/4650], https://hal.inria.fr/hal-01102709

[2] K. CHAUDHURI, N. GUENOT, L. STRASSBURGER. *The Focused Calculus of Structures*, in "Computer Science Logic: 20th Annual Conference of the EACSL", Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, September 2011, pp. 159–173, http://drops.dagstuhl.de/opus/frontdoor.php?source_opus=3229

[3] M. FAROOQUE, S. GRAHAM-LENGRAND, A. MAHBOUBI. *A bisimulation between DPLL(T) and a proof-search strategy for the focused sequent calculus*, in "Proceedings of the 2013 International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP 2013)", A. MOMIGLIANO, B. PIENTKA, R. POLLACK (editors), ACM Press, September 2013, http://dx.doi.org/10.1145/2503887.2503892

[4] A. GACEK, D. MILLER, G. NADATHUR. *Nominal abstraction*, in "Information and Computation", 2011, vol. 209, n° 1, pp. 48–73, http://arxiv.org/abs/0908.1390

[5] A. GUGLIELMI, T. GUNDERSEN, L. STRASSBURGER. *Breaking Paths in Atomic Flows for Classical Logic*, in "Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS 2010)", Edinburgh, United Kingdom, July 2010, pp. 284–293 [*DOI :* 10.1109/LICS.2010.12], http://www.lix.polytechnique.fr/~lutz/papers/AFII.pdf

[6] C. LIANG, D. MILLER. *Focusing and Polarization in Linear, Intuitionistic, and Classical Logics*, in "Theoretical Computer Science", 2009, vol. 410, n° 46, pp. 4747–4768

[7] C. LIANG, D. MILLER. *A Focused Approach to Combining Logics*, in "Annals of Pure and Appl. Logic", 2011, vol. 162, n° 9, pp. 679–697 [*DOI :* 10.1016/J.APAL.2011.01.012], http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lku.pdf

[8] D. MILLER. *A proposal for broad spectrum proof certificates*, in "CPP: First International Conference on Certified Programs and Proofs", J.-P. JOUANNAUD, Z. SHAO (editors), LNCS, 2011, vol. 7086, pp. 54–69, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/cpp11.pdf

[9] L. STRASSBURGER. *Extension without Cut*, in "Annals of Pure and Appl. Logic", 2012, vol. 163, n° 12, pp. 1995–2007

[10] L. STRASSBURGER. *Combinatorial Flows and Their Normalisation*, in "2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK", D. MILLER (editor), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, vol. 84, pp. 31:1–31:17

[11] A. TIU, D. MILLER. *Proof Search Specifications of Bisimulation and Modal Logics for the π-calculus*, in "ACM Trans. on Computational Logic", 2010, vol. 11, n° 2, http://arxiv.org/abs/0805.2785

Publications of the year

Articles in International Peer-Reviewed Journals

[12] B. ACCATTOLI, C. SACERDOTI COEN. *On the Value of Variables*, in "Information and Computation", August 2017, vol. 255, pp. 224 - 242 [*DOI :* 10.1016/J.IC.2017.01.003], https://hal.archives-ouvertes.fr/hal-01675373

[13] Z. CHIHANI, D. MILLER, F. RENAUD. *A Semantic Framework for Proof Evidence*, in "Journal of Automated Reasoning", 2017, vol. 59, n° 3, pp. 287-330 [*DOI :* 10.1007/S10817-016-9380-6], https://hal.inria.fr/hal-01390912

[14] O. FLÜCKIGER, G. SCHERER, M.-H. YEE, A. GOEL, A. AHMED, J. VITEK. *Correctness of Speculative Optimizations with Dynamic Deoptimization*, in "Proceedings of the ACM on Programming Languages", 2017, https://arxiv.org/abs/1711.03050 , forthcoming [*DOI :* 10.1145/3158137], https://hal.inria.fr/hal-01646765

[15] D. MILLER. *Proof Checking and Logic Programming*, in "Formal Aspects of Computing", 2017, vol. 29, n° 3, pp. 383-399 [*DOI :* 10.1007/S00165-016-0393-Z], https://hal.inria.fr/hal-01390901

Articles in Non Peer-Reviewed Journals

[16] D. ILIK. *Perspectives for proof unwinding by programming languages techniques*, in "IfColog Journal of Logics and their Applications (FLAP)", November 2017, vol. 4, n° 10, pp. 3487-3508, https://arxiv.org/abs/1605.09177 , https://hal.inria.fr/hal-01354180

International Conferences with Proceedings

[17] R. BLANCO, Z. CHIHANI, D. MILLER. *Translating Between Implicit and Explicit Versions of Proof*, in "CADE 26 - 26th International Conference on Automated Deduction", Gothenburg, Sweden, August 2017, https://hal.inria.fr/hal-01645016

[18] R. BLANCO, D. MILLER, A. MOMIGLIANO. *Property-Based Testing via Proof Reconstruction Work-in-progress*, in "LFMTP 17: Logical Frameworks and Meta-Languages: Theory and Practice", Oxford, United Kingdom, September 2017, https://hal.inria.fr/hal-01646788

[19] M. P. BONACINA, S. GRAHAM-LENGRAND, N. SHANKAR. *Satisfiability Modulo Theories and Assignments*, in "CADE 2017 - 26th International Conference on Automated Deduction", Gothenburg, Sweden, August 2017 [*DOI :* 10.1007/978-3-319-63046-5_4], https://hal.archives-ouvertes.fr/hal-01615830

[20] P. BRUSCOLI, L. STRASSBURGER. *On the Length of Medial-Switch-Mix Derivations*, in "WoLLIC 2017 - Logic, Language, Information, and Computation", London, United Kingdom, July 2017, https://hal.inria.fr/hal-01635933

[21] S. GRAHAM-LENGRAND, D. JOVANOVIĆ. *An MCSAT treatment of Bit-Vectors (preliminary report)*, in "SMT 2017 - 15th International Workshop on Satisfiability Modulo Theories", Heidelberg, Germany, July 2017, https://hal.archives-ouvertes.fr/hal-01615837

[22] U. GÉRARD, D. MILLER. *Separating Functional Computation from Relations*, in "26th EACSL Annual Conference on Computer Science Logic (CSL 2017)", Stockholm, Sweden, LIPIcs, August 2017, vol. 82, n^{o} 23, pp. 23:1–23:17, https://hal.inria.fr/hal-01615683

[23] D. ILIK. *On the exp-log normal form of types: Decomposing extensional equality and representing terms compactly*, in "Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages", Paris, France, January 2017, pp. 387-399, https://arxiv.org/abs/1502.04634 , https://hal.inria.fr/hal-01167162

[24] S. MARIN, L. STRASSBURGER. *Proof theory for indexed nested sequents*, in "TABLEAUX 2017 - Automated Reasoning with Analytic Tableaux and Related Methods", Brasilia, Brazil, LNCS, September 2017, vol. 10501, pp. 81-97, https://hal.inria.fr/hal-01635935

[25] D. MILLER. *Linear logic as a logical framework*, in "Proceedings of Structures and Deduction (SD) 2017", Oxford, United Kingdom, September 2017, https://hal.archives-ouvertes.fr/hal-01615664

[26] D. MILLER. *Mechanized Metatheory Revisited: An Extended Abstract* , in "Post-proceedings of TYPES 2016", Novi Sad, Serbia, 2017 [*DOI :* 10.4230/LIPICS], https://hal.inria.fr/hal-01615681

[27] G. SCHERER. *Deciding equivalence with sums and the empty type*, in "POPL 2017", Paris, France, POPL 2017- Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, January 2017, https://arxiv.org/abs/1610.01213 - This work was presented at POPL 2017: Principles of Programming Languages [*DOI :* 10.01213], https://hal.inria.fr/hal-01646064

[28] L. STRASSBURGER. *Combinatorial Flows and their Normalisation*, in "FSCD 2017 - 2nd International Conference on Formal Structures for Computation and Deduction", Oxford, United Kingdom, Leibniz International Proceedings in Informatics (LIPIcs), September 2017, vol. 84, pp. 311 - 3117 [*DOI :* 10.4230/LIPICS.FSCD.2017.31], https://hal.inria.fr/hal-01635931

National Conferences with Proceedings

[29] G. BARANY, G. SCHERER. *Génération aléatoire de programmes guidée par la vivacité*, in "JFLA: Journées Francophones des Langages Applicatifs", Banyuls-sur-Mer, France, January 2018, https://hal.inria.fr/hal-01682691

Conferences without Proceedings

[30] B. ACCATTOLI, B. BARRAS. *Environments and the Complexity of Abstract Machines*, in "The 19th International Symposium on Principles and Practice of Declarative Programming", Namur, Belgium, October 2017 [*DOI :* 10.1145/3131851.3131855], https://hal.archives-ouvertes.fr/hal-01675358

[31] B. ACCATTOLI, B. BARRAS. *The Negligible and Yet Subtle Cost of Pattern Matching*, in "Programming Languages and Systems - 15th Asian Symposium", Suzhou, China, November 2017, https://hal.archives-ouvertes.fr/hal-01675369

[32] B. ACCATTOLI, G. GUERRIERI. *Implementing Open Call-by-Value*, in "Fundamentals of Software Engineering - 7th International Conference", Teheran, Iran, April 2017, https://hal.archives-ouvertes.fr/hal-01675365

[33] D. MILLER. *Using linear logic and proof theory to unify computational logic*, in "Proceedings of Trends in Linear Logic and Applications (TLLA 17)", Oxford, United Kingdom, September 2017, https://hal.archives-ouvertes.fr/hal-01615673

Books or Proceedings Editing

[34] D. MILLER (editor). *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Oxford, United Kingdom, September 2017, https://hal.inria.fr/hal-01615598

Research Reports

[35] S. MARIN, L. STRASSBURGER. *On the Proof Theory of Indexed Nested Sequents for Classical and Intuitionistic Modal Logics*, Inria, April 2017, n^{o} RR-9061, https://hal.inria.fr/hal-01515797

[36] L. STRASSBURGER, R. KUZNETS. *Maehara-style Modal Nested Calculi*, Inria Saclay, November 2017, n^{o} RR-9123, https://hal.inria.fr/hal-01644750

[37] L. STRASSBURGER. *Combinatorial Flows and Proof Compression*, Inria Saclay, March 2017, n^{o} RR-9048, https://hal.inria.fr/hal-01498468

[38] L. STRASSBURGER. *Deep Inference, Expansion Trees, and Proof Graphs for Second Order Propositional Multiplicative Linear Logic*, Inria Saclay Ile de France, May 2017, n^{o} RR-9071, 38 p. , https://hal.inria.fr/hal-01526831

Other Publications

[39] J. COURTIEL, K. YEATS, N. ZEILBERGER. *Connected chord diagrams and bridgeless maps*, November 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01650141

[40] R. KUZNETS, S. MARIN, L. STRASSBURGER. *Justification logic for constructive modal logic* *, July 2017, IMLA 2017 - 7th Workshop on Intuitionistic Modal Logic and Applications, https://hal.inria.fr/hal-01614707

References in notes

[41] J. A. ROBINSON, A. VORONKOV (editors). *Handbook of Automated Reasoning*, Elsevier and MIT press, 2001

[42] S. ABRAMSKY. *Computational Interpretations of Linear Logic*, in "Theoretical Computer Science", 1993, vol. 111, pp. 3–57

[43] B. ACCATTOLI, U. DAL LAGO. *Beta reduction is invariant, indeed*, in "Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014", 2014, pp. 8:1–8:10, http://doi.acm.org/10.1145/2603088.2603105

[44] J.-M. ANDREOLI. *Logic Programming with Focusing Proofs in Linear Logic*, in "Journal of Logic and Computation", 1992, vol. 2, n^{o} 3, pp. 297–347

[45] S. N. ARTEMOV. *Operational modal logic*, Cornell University, 1995, n^{o} MSI 95–29

[46] A. ASSAF, G. BUREL, R. CAUDERLIER, D. DELAHAYE, G. DOWEK, C. DUBOIS, F. GILBERT, P. HALMAGRAND, O. HERMANT, R. SAILLARD. *Dedukti: a Logical Framework based on the λΠ-Calculus Modulo Theory*, 2016, Unpublished, http://www.lsv.ens-cachan.fr/~dowek/Publi/expressing.pdf

[47] D. BAELDE, D. MILLER, Z. SNOW. *Focused Inductive Theorem Proving*, in "Fifth International Joint Conference on Automated Reasoning (IJCAR 2010)", J. GIESL, R. HÄHNLE (editors), LNCS, 2010, n^{o} 6173, pp. 278–292 [*DOI :* 10.1007/978-3-642-14203-1], http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/ijcar10.pdf

[48] G. E. BLELLOCH, J. GREINER. *Parallelism in Sequential Functional Languages*, in "Proceedings of the seventh international conference on Functional programming languages and computer architecture, FPCA 1995, La Jolla, California, USA, June 25-28, 1995", 1995, pp. 226–237, http://doi.acm.org/10.1145/224164.224210

[49] M. P. BONACINA, S. GRAHAM-LENGRAND, N. SHANKAR. *A model-constructing framework for theory combination*, Universita degli Studi di Verona, September 2016, n^{o} RR-99/2016, https://hal.inria.fr/hal-01425305

[50] K. CHAUDHURI. *The Focused Inverse Method for Linear Logic*, Carnegie Mellon University, December 2006, Technical report CMU-CS-06-162, http://reports-archive.adm.cs.cmu.edu/anon/2006/CMU-CS-06-162.pdf

[51] K. CHAUDHURI, N. GUENOT, L. STRASSBURGER. *The Focused Calculus of Structures*, in "Computer Science Logic: 20th Annual Conference of the EACSL", Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, September 2011, pp. 159–173 [*DOI :* 10.4230/LIPICS.CSL.2011.159], http://drops.dagstuhl.de/opus/volltexte/2011/3229/pdf/16.pdf

[52] K. CHAUDHURI, S. HETZL, D. MILLER. *A Multi-Focused Proof System Isomorphic to Expansion Proofs*, in "Journal of Logic and Computation", June 2014 [*DOI :* 10.1093/LOGCOM/EXU030], http://hal.inria.fr/hal-00937056

[53] Z. CHIHANI, D. MILLER, F. RENAUD. *Checking Foundational Proof Certificates for First-Order Logic (extended abstract)*, in "Third International Workshop on Proof Exchange for Theorem Proving (PxTP 2013)", J. C. BLANCHETTE, J. URBAN (editors), EPiC Series, EasyChair, 2013, vol. 14, pp. 58–66

[54] Z. CHIHANI, D. MILLER, F. RENAUD. *Foundational proof certificates in first-order logic*, in "CADE 24: Conference on Automated Deduction 2013", M. P. BONACINA (editor), Lecture Notes in Artificial Intelligence, 2013, n^{o} 7898, pp. 162–177

[55] C. DROSS, S. CONCHON, J. KANIG, A. PASKEVICH. *Reasoning with Triggers*, in "10th Intern. Worksh. on Satisfiability Modulo Theories, SMT 2012", P. FONTAINE, A. GOEL (editors), EPiC Series, EasyChair, June 2012, vol. 20, pp. 22–31, http://www.easychair.org/publications/?page=2135488790

[56] P. FONTAINE, J.-Y. MARION, S. MERZ, L. P. NIETO, A. TIU. *Expressiveness + Automation + Soundness: Towards Combining SMT Solvers and Interactive Proof Assistants*, in "TACAS: Tools and Algorithms for the Construction and Analysis of Systems, 12th International Conference", H. HERMANNS, J. PALSBERG (editors), LNCS, Springer, 2006, vol. 3920, pp. 167–181, http://dx.doi.org/10.1007/11691372_11

[57] A. GACEK, D. MILLER, G. NADATHUR. *Combining generic judgments with recursive definitions*, in "23th Symp. on Logic in Computer Science", F. PFENNING (editor), IEEE Computer Society Press, 2008, pp. 33–44, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lics08a.pdf

[58] J.-Y. GIRARD. *Linear Logic*, in "Theoretical Computer Science", 1987, vol. 50, pp. 1–102

[59] S. GRAHAM-LENGRAND, R. DYCKHOFF, J. MCKINNA. *A Focused Sequent Calculus Framework for Proof Search in Pure Type Systems*, in "Logical Methods in Computer Science", 2011, vol. 7, n^{o} 1, http://www.lix.polytechnique.fr/~lengrand/Work/Reports/TTSC09.pdf

[60] S. GRAHAM-LENGRAND. *Slot Machines: an approach to the Strategy Challenge in SMT solving (presentation only)*, in "13th International Workshop on Satisfiability Modulo Theories", San Francisco, United States, July 2015, https://hal.inria.fr/hal-01211209

[61] A. GUGLIELMI. *A System of Interaction and Structure*, in "ACM Trans. on Computational Logic", 2007, vol. 8, n^{o} 1

[62] A. GUGLIELMI, T. GUNDERSEN. *Normalisation Control in Deep Inference Via Atomic Flows*, in "Logical Methods in Computer Science", 2008, vol. 4, n^{o} 1:9, pp. 1–36, http://arxiv.org/abs/0709.1205

[63] A. GUGLIELMI, L. STRASSBURGER. *Non-commutativity and MELL in the Calculus of Structures*, in "Computer Science Logic, CSL 2001", L. FRIBOURG (editor), LNCS, Springer-Verlag, 2001, vol. 2142, pp. 54–68

[64] Q. HEATH, D. MILLER. *A framework for proof certificates in finite state exploration*, in "Proceedings of the Fourth Workshop on Proof eXchange for Theorem Proving", C. KALISZYK, A. PASKEVICH (editors), Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, August 2015, n^{o} 186, pp. 11–26 [*DOI :* 10.4204/EPTCS.186.4], http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/pxtp2015.pdf

[65] D. HUGHES. *Proofs Without Syntax*, in "Annals of Mathematics", 2006, vol. 164, n^{o} 3, pp. 1065–1076

[66] D. JOVANOVIĆ, C. BARRETT, L. DE MOURA. *The Design and Implementation of the Model Constructing Satisfiability Calculus*, in "Proc. of the 13th Int. Conf. on Formal Methods In Computer-Aided Design (FMCAD '13)", FMCAD Inc., 2013, Portland, Oregon, http://www.cs.nyu.edu/~barrett/pubs/JBdM13.pdf

[67] F. LAMARCHE, L. STRASSBURGER. *Naming Proofs in Classical Propositional Logic*, in "Typed Lambda Calculi and Applications, TLCA 2005", P. URZYCZYN (editor), LNCS, Springer, 2005, vol. 3461, pp. 246–261

[68] C. LIANG, D. MILLER. *Focusing and Polarization in Linear, Intuitionistic, and Classical Logics*, in "Theoretical Computer Science", 2009, vol. 410, n° 46, pp. 4747–4768, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tcs09.pdf

[69] C. LIANG, D. MILLER. *A Focused Approach to Combining Logics*, in "Annals of Pure and Applied Logic", 2011, vol. 162, n° 9, pp. 679–697, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lku.pdf

[70] P. MARTIN-LÖF. *Constructive Mathematics and Computer Programming*, in "Sixth International Congress for Logic, Methodology, and Philosophy of Science", Amsterdam, North-Holland, 1982, pp. 153–175

[71] R. MCDOWELL, D. MILLER. *Reasoning with Higher-Order Abstract Syntax in a Logical Framework*, in "ACM Trans. on Computational Logic", 2002, vol. 3, n° 1, pp. 80–136, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/mcdowell01.pdf

[72] R. MCDOWELL, D. MILLER. *A Logic for Reasoning with Higher-Order Abstract Syntax*, in "Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science", Warsaw, Poland, G. WINSKEL (editor), IEEE Computer Society Press, July 1997, pp. 434–445

[73] D. MILLER. *Forum: A Multiple-Conclusion Specification Logic*, in "Theoretical Computer Science", September 1996, vol. 165, n° 1, pp. 201–232

[74] D. MILLER, G. NADATHUR. *Programming with Higher-Order Logic*, Cambridge University Press, June 2012, http://dx.doi.org/10.1017/CBO9781139021326

[75] D. MILLER, G. NADATHUR, F. PFENNING, A. SCEDROV. *Uniform Proofs as a Foundation for Logic Programming*, in "Annals of Pure and Applied Logic", 1991, vol. 51, pp. 125–157

[76] D. MILLER, A. TIU. *A Proof Theory for Generic Judgments: An extended abstract*, in "Proc. 18th IEEE Symposium on Logic in Computer Science (LICS 2003)", IEEE, June 2003, pp. 118–127, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lics03.pdf

[77] R. MILNER. *LCF: A Way of Doing Proofs with a Machine*, in "Proc. of the 8th Intern. Symp. on the Mathematical Foundations of Computer Science", J. BECVÁR (editor), LNCS, Springer, 1979, vol. 74, pp. 146-159

[78] G. NELSON, D. C. OPPEN. *Simplification by Cooperating Decision Procedures*, in "ACM Press Trans. on Program. Lang. and Syst.", October 1979, vol. 1, n° 2, pp. 245–257, http://dx.doi.org/10.1145/357073.357079

[79] F. PFENNING, C. SCHÜRMANN. *System Description: Twelf — A Meta-Logical Framework for Deductive Systems*, in "16th Conference on Automated Deduction", Trento, H. GANZINGER (editor), LNAI, Springer, 1999, n° 1632, pp. 202–206

[80] B. PIENTKA, J. DUNFIELD. *Beluga: A Framework for Programming and Reasoning with Deductive Systems (System Description)*, in "Fifth International Joint Conference on Automated Reasoning", J. GIESL, R. HÄHNLE (editors), LNCS, 2010, n^{o} 6173, pp. 15–21

[81] E. P. ROBINSON. *Proof Nets for Classical Logic*, in "Journal of Logic and Computation", 2003, vol. 13, pp. 777–797

[82] R. E. SHOSTAK. *Deciding Combinations of Theories*, in "J. ACM", 1984, vol. 31, n^{o} 1, pp. 1–12, http://dx.doi.org/10.1145/2422.322411

[83] L. STRASSBURGER, R. KUZNETS. *Maehara-style Modal Nested Calculi*, Inria Saclay, November 2017, n^{o} RR-9123, https://hal.inria.fr/hal-01644750

[84] THE COQ DEVELOPMENT TEAM. *The Coq Proof Assistant Version 8.3 Reference Manual*, Inria, October 2010

[85] A. TIU, D. MILLER. *Proof Search Specifications of Bisimulation and Modal Logics for the π-calculus*, in "ACM Trans. on Computational Logic", 2010, vol. 11, n^{o} 2, http://arxiv.org/abs/0805.2785

[86] L. M. DE MOURA, D. JOVANOVIC. *A Model-Constructing Satisfiability Calculus*, in "Proc. of the 14th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)", R. GIACOBAZZI, J. BERDINE, I. MASTROENI (editors), LNCS, Springer-Verlag, 2013, vol. 7737, pp. 1–12, http://dx.doi.org/10.1007/978-3-642-35873-9_1

[87] L. M. DE MOURA, G. O. PASSMORE. *The Strategy Challenge in SMT Solving*, in "Automated Reasoning and Mathematics - Essays in Memory of William W. McCune", M. P. BONACINA, M. E. STICKEL (editors), LNCS, Springer, 2013, vol. 7788, pp. 15–44 [*DOI :* 10.1007/978-3-642-36675-8_2], http://dx.doi.org/10.1007/978-3-642-36675-8