IN PARTNERSHIP WITH:
CNRS

Ecole Polytechnique

Activity Report 2019

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, end of the Project-Team: 2019 November 30

Keywords:

Computer Science and Digital Science:

A2.1. - Programming Languages
A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.5. - Constraint programming
A2.1.10. - Domain-specific languages
A2.2.1. - Static analysis
A2.4.3. - Proofs
A2.5.4. - Software Maintenance & Evolution
A7.2.1. - Decision procedures
A7.2.2. - Automated Theorem Proving
A7.2.3. - Interactive Theorem Proving
A7.3.1. - Computational models and calculability
A9.8. - Reasoning

Other Research Topics and Application Domains:

B9.5.1. - Computer science
B9.5.2. - Mathematics
B9.8. - Reproducibility

1. Team, Visitors, External Collaborators

Research Scientists

Dale Miller [Team leader, Inria, Senior Researcher, HDR]
Beniamino Accattoli [Inria, Researcher]
Kaustuv Chaudhuri [Inria, Researcher]
François Lamarche [Inria, Senior Researcher]
Stéphane Lengrand [Researcher, until Dec 2019, HDR]
Ian Mackie [CNRS, Researcher]
Gabriel Scherer [Inria, Researcher]
Lutz Strassburger [Inria, Researcher, HDR]

Post-Doctoral Fellows

Marianna Girlando [Inria, Post-Doctoral Fellow, from Nov 2019]
Luc Pellissier [Inria, Post-Doctoral Fellow, from Nov 2019]
Benjamin Ralph [Inria, Post-Doctoral Fellow, from Feb 2019]

PhD Students

Ulysse Gerard [Université Paris-Diderot (Paris 7), PhD Student, until Oct 2019]
Maico Carlos Leberle [Inria, PhD Student]
Matteo Manighetti [Inria, PhD Student]
Emilie Grienenberger [Inria, PhD Student, until Oct 2022]
Marianela Morales [Inria, PhD Student, from Aug 2019]

Technical staff
Matteo Acclavio [Telecom ParisTech, Engineer, from Dec 2019]

Intern and Apprentice
Francesco Mecca [Inria, from Sep 2019]

Administrative Assistant
Maeva Jeannot [Inria, Administrative Assistant, until Oct 2019]

Visiting Scientist
Claudio Sacerdoti Coen [Università di Bologna, from Aug 2019 until Sep 2019]

External Collaborator
Andrea Condoluci [Università di Bologna, until Jul 2019]

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 [51], and normalization in different logics can justify the design of new and different functional programming languages [32].
- *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 [55], and different proof search strategies can be used to justify the design of new and different logic programming languages [54].

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 [34] 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 [60] 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 [57] 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 [58], 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 [53] [52] [56] [61], and on more recent work by Gacek, Miller, and Nadathur [41] [40]. The Parsifal team, along with our colleagues in Minneapolis, Canberra, Singapore, and Cachan, 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* [35] [37] [38] [39] [43] [49] [50] 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 [44], [46] 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 category-theoretic 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 [42] for linear logic but also in Robinson’s proof nets [59] 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., [48][2] leading to the notion of *atomic flow* and [45]. 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 [47] 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 in the combinatorial proof formalism.

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 [36] (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 [33].

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. Automated Theorem Proving

The Parsifal 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 Parsial.)

4.2. Proof-assistants

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

4.3. Programming language design

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

5. Highlights of the Year

5.1. Highlights of the Year

- The journal Mathematical Structures in Computer Science published "A special issue on structural proof theory, automated reasoning and computation in celebration of Dale Miller's 60th birthday" – volume 29, Special issue 8, September 2019.
- Accattoli was invited speaker at the international conference FSCD 2019.

6. New Software and Platforms

6.1. 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/

6.2. Bedwyr

Bedwyr - A proof search approach to model checking

KEYWORD: Model Checker

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: Dale Miller
- URL: http://slimmer.gforge.inria.fr/bedwyr/

6.3. 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

6.4. Psyche

Proof-Search factorY for Collaborative HEuristics

KEYWORD: Automated theorem proving

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.csl.sri.com/users/sgl/

6.5. Maetning

FUNCTIONAL DESCRIPTION: Mætning is an automated theorem prover for intuitionistic predicate logic that is designed to disprove non-theorems.

- Contact: Kaustuv Chaudhuri
- URL: https://github.com/chaudhuri/maetning/

6.6. 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.

- Participants: Damien Doligez, Xavier Leroy, Fabrice Le Fessant, Luc Maranget, Gabriel Scherer, Alain Frisch, Jacques Garrigue, Marc Shinwell, Jeremy Yallop and Leo White
- Contact: Damien Doligez
- URL: https://ocaml.org/

7. New Results

7.1. Functional programming with λ-tree syntax

Participants: Ulysse Gerard, Dale Miller, Gabriel Scherer.

We have been designing a new functional programming language, MLTS, that uses the λ-*tree* syntax approach to encoding bindings that appear within data structures [20]. In this setting, bindings never become free nor escape their scope: instead, binders in data structures are permitted to *move* into binders within programs phrases. The design of MLTS—whose concrete syntax is based on that of OCaml—includes additional sites within programs that directly support this movement of bindings. Our description of MLTS includes a typing discipline that naturally extends the typing of OCaml programs.

In addition to the natural semantics for MLTS that we proposed in 2018, we also have a small-step operational semantics which gives in particular a fine-grained description of the runtime behavior of the ∇ operator in patterns. It leads in particular to a direct implementation in Lambda-Prolog (which does not contain a native ∇ operator) that allows more expressive constructs (higher-arity types) than our previous presentation.

7.2. Mechanized metatheory revisited

Participant: Dale Miller.

When proof assistants and theorem provers implement the metatheory of logical systems, they must deal with a range of syntactic expressions (e.g., types, formulas, and proofs) that involve variable bindings. Since most mature proof assistants do not have built-in methods to treat bindings, they have been extended with various packages and libraries that allow them to encode such syntax using, for example, De Bruijn numerals. In the paper, [10], Miller puts forward the argument that bindings are such an intimate aspect of the structure of expressions that they should be accounted for directly in the underlying programming language support for proof assistants and not via packages and libraries. He presents an approach to designing programming languages and proof assistants that directly supports bindings in syntax. The roots of this approach can be found in the *mobility* of binders between term-level bindings, formula-level bindings (quantifiers), and proof-level bindings (eigenvariables). In particular, the combination of Church's approach to terms and formulas (found in his Simple Theory of Types) and Gentzen's approach to proofs (found in his sequent calculus) yields a framework for the interaction of bindings with a full range of logical connectives and quantifiers. Miller also illustrates how that framework provides a direct and semantically clean treatment of computation and reasoning with syntax containing bindings.

7.3. New applications of Foundational Proof Certificates

Participants: Kaustuv Chaudhuri, Matteo Manighetti, Dale Miller.

The formal framework of *Foundational Proof Certificates* (FPC) was developed in previous years within the Parsifal team. We continue to push on their applications in a number of settings in computational logic. In 2019, we developed two such new applications.

In order to apply FPCs to the conventional setting of classical logic theorem provers, the FPC setting needed to treat proof evidence containing Skolem functions. Using FPC directly meant that we needed to do such certification without using the mathematical concepts of model-theoretic semantics (i.e., preservation of satisfiability) and choice principles (i.e., epsilon terms). Instead, our proof checking kernel is an implementation of Gentzen's sequent calculus, which directly supports quantifier alternation by using eigenvariables. In [19], we described deskolemization as a mapping from client-side terms, used in proofs generated by theorem provers, into kernel-side terms, used within our proof checking kernel. This mapping which associates skolemized terms to eigenvariables relies on using outer skolemization.

Property-based testing (PBT) is a technique for validating code against an executable specification by automatically generating test-data. In the paper [18], we presented a proof-theoretical reconstruction of this style of testing for relational specifications and employ FPCs to describe test generators. We did this by presenting certain kinds of "proof outlines" that can be used to describe various common generation strategies in the PBT literature, ranging from random to exhaustive, including their combination. We also address the shrinking of counterexamples as a first step towards their explanation. Once generation is accomplished, the testing phase boils down to a standard logic programming search. We could also we lift our techniques to treat data structures containing bindings using λ-*tree* syntax. The λProlog programming language is capable of performing both the generation and checking of tests. We validated this approach by tackling benchmarks in the metatheory of programming languages coming from related tools such as PLT-Redex Property-Based Testing via Proof Reconstruction. This work was done in collaboration with Roberto Blanco, a postdoc from Inria Paris, and Alberto Momigliano, a professor from the University of Milan.

7.4. Historical reflections on proof theory and logic programming

Participant: Dale Miller.

Miller has been working in the area of logic programming and proof theory for more than three decades. Some of his historical reflections on how these two topics influenced each other are contained in the paper [11]. While it is widely known that proof theory has been helpful in shaping the development of logic programming, particular of extensions to conventional Prolog, this paper also documents a few specific examples where logic programming influenced the development of some topics in proof theory.

7.5. Intuitionistic proofs without syntax

Participant: Lutz Straßburger.

We present Intuitionistic Combinatorial Proofs (ICPs), a concrete geometric semantics of intuitionistic logic based on the principles of classical combinatorial proofs. An ICP naturally factorizes into a linear fragment, a graphical abstraction of an IMLL proof net (an arena net), and a parallel contraction-weakening fragment (a skew fibration). ICPs relate to game semantics, and can be seen as a strategy in a Hyland-Ong arena, generalized from a tree-like to a dag-like strategy. Our first main result, Polynomial Full Completeness, is thatICPs as a semantics are complexity-aware: the translations to and from sequent calculus are size-preserving (up to a polynomial). By contrast, lambda-calculus and game semantics incur an exponential blowup. Our second main result, Local Canonicity, is that ICPs abstract fully and faithfully over the non-duplicating permutations of the sequent calculus. These results have been presented at the LICS 2019 conference [23].

7.6. Towards a combinatorial proof theory

Participants: Lutz Straßburger, Benjamin Ralph.

The main part of a classical combinatorial proof is a skew fibration, which precisely captures the behavior of weakening and contraction. Relaxing the presence of these two rules leads to certain substructural logics and substructural proof theory. We investigated what happens if we replace the skew fibration by other kinds of graph homomorphism. This leads us to new logics and proof systems that we call combinatorial. This has been presented at the TABLEAUX 2019 conference [22].

7.7. Combinatorial Proofs for Logics of Relevance and Entailment

Participants: Lutz Straßburger, Matteo Acclavio.

In this work (presented at the WoLLIC 2019 conference [16]) we characterize classical combinatorial proofs which also represent valid proofs for relevant logic with and without the mingle axiom. Moreover, we extend our syntax in order to represent combinatorial proofs for the more restrictive framework of entailment logic.

7.8. On combinatorial proofs for modal logic

Participants: Lutz Straßburger, Matteo Acclavio.

In this work [17], we extend combinatorial proofs to modal logics. The crucial ingredient for modeling the modalities is the use of a self-dual non-commutative operator that has first been observed by Retoré through pomset logic. Consequently, we had to generalize the notion of skew fibration from cographs to Guglielmi's relation webs. Our main result is a sound and complete system of combinatorial proofs for all normal and non-normal modal logics in the S4-tesseract. The proof of soundness and completeness is based on the sequent calculus with some added features from deep inference.

7.9. Deep inference and expansion trees for second-order multiplicative linear logic

Participant: Lutz Straßburger.

In this work, we introduce the notion of expansion tree for linear logic. As in Miller's original work, we have a shallow reading of an expansion tree that corresponds to the conclusion of the proof, and a deep reading which is a formula that can be proved by propositional rules. We focus our attention to MLL2, and we also present a deep inference system for that logic. This allows us to give a syntactic proof to a version of Herbrand's theorem. This has been published in an special issue of MSCS [12].

7.10. A fully labelled proof system for intuitionistic modal logics

Participants: Lutz Straßburger, Marianela Morales.

In this paper we present a labelled sequent system for intuitionistic modal logics such that there is not only one, but two relation symbols appearing in sequents: one for the accessibility relation associated with the Kripke semantics for normal modal logics and one for the preorder relation associated with the Kripke semantics for intuitionistic logic. This puts our system in close correspondence with the standard birelational Kripke semantics for intuitionistic modal logics. As a consequence it can encompass a wider range of intuitionistic modal logics than existing labelled systems. We also show an internal cut elimination proof for our system [30].

7.11. Types by Need

Participants: Beniamino Accattoli, Maico Leberle.

This joint work with Giulio Guerrieri (Post-doc at Bath University) [27] develops a multi type system for call-by-need evaluation of the -calculus. The type system is obtained by combining features by well-known systems for call-by-name and call-by-value. It characterizes termination, and, moreover, its type derivations provide precise information about the number of steps to reach the result. The novelty is that, while the systems for call-by-name and call-by-value are obtained by the linear logic interpretation of these evaluation schemes, call-by-need has no linear logic interpretation.

7.12. Sharing Equality is Linear

Participants: Beniamino Accattoli, Andrea Condoluci, Claudio Sacerdoti Coen.

This work [28] studies how to compare higher-order programs with sharing for sharing equality, that is, for equality of their unshared underlying programs. The point, of course, is to do it efficiently, without unsharing the programs, that would otherwise introduce an exponential blow-up. We develop the first algorithm linear in the size of the shared terms, by adapting the famous Patterson and Wegman algorithm for first-order unification.

7.13. Crumbling Abstract Machines

Participants: Beniamino Accattoli, Andrea Condoluci, Claudio Sacerdoti Coen.

This joint work with Giulio Guerrieri (Post-doc at Bath University) [26] studies a new compilation technique for functional programs, dubbed *crumbling* and resembling the transformation into administrative normal form of Flanagan, Sabry, Duba, and Felleisen. It is shown that it simplifies the design of abstract machines without altering the complexity of the overhead. Moreover, it smoothly scales up to open terms and it does not suffer of the slowdowns of administrative normal forms pointed out by Kennedy.

7.14. Factorization and Normalization, Essentially

Participant: Beniamino Accattoli.

This joint work with Claudia Faggian (CNRS researcher at Paris Diderot) and Giulio Guerrieri (Post-doc at Bath University) [15] refines a rewriting technique for proving factorization and normalization theorems for λ-calculi, that are theorems providing foundations to the design of functional programming languages and proof assistants. We both simplify and extend the scope of a widely used technique by Takahashi. At the concrete level, the new abstract technique is applied to four relevant case studies.

7.15. A Fresh Look at the -Calculus

Participant: Beniamino Accattoli.

This paper [25] is the trace of the invited talk given by Accattoli at FSCD 2019. More than just an abstract, the paper is a lengthy overview of the research on λ-calculus, cost models, sharing, and abstract machines pursued by Accattoli and his co-authors in the last 10 years.

7.16. Abstract Machines for Open Call-by-Value

Participant: Beniamino Accattoli.

This journal paper in collaboration with Giulio Guerrieri (Post-doc at Bath University) [4] outlines a theory of abstract machines for the call-by-value -calculus with open terms. It refines and extends the results by the same authors from 2017, which were among the selected ones from the international conference FSEN 2017 for publication in a journal.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Grants with Industry

8.1.1. OCaml Software Foundation

Participant: Gabriel Scherer.

The OCaml Software Foundation (OCSF), [1] 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.

Gabriel Scherer serves as the director of the foundation.

8.1.2. Funding from Nomadic Labs

Participant: Gabriel Scherer.

Nomadic Labs, a Paris-based company, has implemented the Tezos blockchain and cryptocurrency entirely in OCaml. This year, 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 two 3-year grants, in collaboration with the Cambium team 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. Regional Initiatives

9.1.1. DIM-RFSI

Gabriel Scherer obtained funding from the Région Île-de-France to hire a post-doc, Luc Pellissier, to work on canonical representation of programs (linking proof theory and category-theory approaches), in collaboration with Adrien Guatto in IRIF (Université Paris 7).

9.2. National Initiatives

9.2.1. ANR

COCA HOLA: Cost Models for Complexity Analyses of Higher-Order Languages, coordinated by B. Accattoli, 2016–2019.

FISP: The Fine Structure of Formal Proof Systems and their Computational Interpretations, coordinated by Lutz Straßburger in collaboration with Université Paris 7, Universität Innsbruck and TU Wien, 2016–2019.

9.2.2. Competitivity Clusters

UPScale: Universality of Proofs in SaCLay, a Working Group of LabEx DigiCosme, organized by Chantal Keller (LRI) with regular participation from Parsifal members and a post-doc co-supervision.

9.3. International Research Visitors

9.3.1. Visits of International Scientists

Claudio Sacerdoti Coen (Universita di Bologna, Italy) spent a month visiting Beniamino Accattoli thanks to funding for short-term international visits.

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

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events: Organisation

10.1.1.1. General Chair, Scientific Chair

Dale Miller is currently General Chair for the ACM/IEEE Symposium on Logic in Computer Science (LICS).

10.1.1.2. Member of the Organizing Committees

Dale Miller is member of the Steering Committee for CPP, FSCD, and LFMTP and is a member of the Executive Committee of the ACM Special Interest Group on Logic and Compuation (SIGLOG).

Gabriel Scherer is part of the Steering Committee of the ML Family Workshop and the OCaml Workshop.

10.1.2. Scientific Events: Selection

10.1.2.1. Chair of Conference Program Committees

Dale Miller was program committee co-chair for the workshop on Logical Frameworks and Meta Languages: Theory and Practice (LFMTP), 2019, Vancouver, Canada.

10.1.2.2. Member of the Conference Program Committees

- Gabriel Scherer: JFLA 2020.
- Beniamino Accattoli: LSFA 2019.
- Dale Miller is a PC member for the 10th International Joint Conference on Automated Reasoning (IJCAR-2020), the 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-23), and the Workshop on Trends, Extensions, Applications and Semantics of Logic Programming (TEASE-LP).

10.1.2.3. Reviewer

- Gabriel Scherer: POPL 2020, ESOP 2020.
- Lutz Straßburger: FoSSaCS 2020, CSL 2020, TABLEAUX 2019, WoLLIC 2019, LICS 2019.
- Beniamino Accattoli: FoSSaCS 2020, POPL 2020, ICALP 2019, ICTAC 2019, PPDP 2019, FSCD 2019, CSL 2019, LSFA 2019.

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

Dale Miller is on the editorial board of the Journal of Automated Reasoning (Springer) and the Jounral of Applied Logic (Elsevier). He has also been an editor for a special issue of FSCD 2017 for Logical Methods in Computer Science.

10.1.3.2. Reviewer - Reviewing Activities

- Lutz Straßburger: APAL, ToCL, BSL, LMCS.
- Beniamino Accattoli: LMCS, TCS.
- Dale Miller has served on the evaluation committee for the EATCS Distinguished Dissertation Award and the EACSL Ackermann Award. He was also the Chair of the Herbrand Award Committee of the Association for Automated Reasoning.

10.1.4. Invited Talks

- Beniamino Accattoli has been invited speaker at FSCD 2019.
- Dale Miller has been an invited speaker at the Workshop on Proof Theory for Automated Deduction, Automated Deduction for Proof Theory (23-25 October 2019, Funchal, Madeira) and the Third Tübingen Conference on Proof-Theoretic Semantics, 27-30 March 2019.

10.1.5. Scientific Expertise

- Lutz Straßburger: reviewer for the Israel Science Foundation.
- Dale Miller has been a member of an international review panel for the Distinguished Professor Grant at the Swedish Research Council.
- Gabriel Scherer: reviewer for ANR.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Licence : G. Scherer, "*Introduction á la programmation fonctionnelle*", 50h, L1, Paris 8 (Vincennes - Saint Denis), France

Master: B. Accattoli, "*Logique linéaire et paradigmes logiques du calcul*", 18h, M2, Master Parisien de Recherche en Informatique (MPRI), France.

Master: D. Miller, "*Logique linéaire et paradigmes logiques du calcul*", 18h, M2, Master Parisien de Recherche en Informatique (MPRI), France.

Summer School: G. Scherer, "*Programmation fonctionnelle en OCaml*", 12h, public d'ingénieurs de recherche, formation ANF (CNRS), France

Summer School: L. Straßburger: "*Introduction to Deep Inference*", 10h, ESSLLI 2020, Riga, Latvia

Summer School: B. Accattoli, "λ-*Calculus and Reasonable Cost Models*", 15h, Escuela de Ciencia Informaticas (ECI 2019), Buenos Aires, Argentina.

10.2.2. Supervision

- PhD in progress: Marianela Morales, "Combinatorial Proof Theory for Modal Logic", 1/10/2019, Lutz Straßburger.
- PhD in progress: Maico Leberle, "Call-by-need and Reasonable Cost Models", Beniamino Accattoli.
- PhD completed: Ulysse Gérard, "Computing with relations, functions, and bindings", 18 October 2019, Ecole Doctorale de l'Institut Polytechnique de Paris, advised by Dale Miller.
- PhD in progress: Matteo Manighetti, "Structural proof theory for induction in linear logic", advised by Dale Miller since 1/10/2017.
- PhD in progress: Marianela Morales, "Combinatorial Proof Theory for Modal Logic", 1/10/2019, Lutz Straßburger

10.2.3. Juries

Dale Miller was a reportor for the PhD thesis of Aurore Alcolei (ENS Lyon, 17 October 2019).

10.3. Popularization

10.3.1. Interventions

G. Scherer and M. Manighetti participated the "Fête de la Science" exhibit at Inria Saclay on the whole day of October 11th, 2019.

G. Scherer presented the research domain of certified programming to an audience of computer security professionals at the "Pass the Salt" conference in Lille, on Wednesday July 3rd.

11. 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] 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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[3] U. GÉRARD. *Computing with relations, functions, and bindings*, École Polytechnique ; Inria Saclay, October 2019, https://hal.archives-ouvertes.fr/tel-02414237

Articles in International Peer-Reviewed Journals

[4] B. ACCATTOLI, G. GUERRIERI. *Abstract Machines for Open Call-by-Value*, in "Science of Computer Programming", 2019, vol. 184 [*DOI :* 10.1016/J.SCICO.2019.03.002], https://hal.archives-ouvertes.fr/hal-02415780

[5] T. BROCK-NANNESTAD, D. ILIK. *An Intuitionistic Formula Hierarchy Based on High-School Identities*, in "Mathematical Logic Quarterly", May 2019, vol. 65, n° 1, pp. 57-79, https://arxiv.org/abs/1601.04876 [*DOI :* 10.1002/MALQ.201700047], https://hal.inria.fr/hal-01354181

[6] K. CHAUDHURI, C. OLARTE, E. PIMENTEL, J. DESPEYROUX. *Hybrid Linear Logic, revisited*, in "Mathematical Structures in Computer Science", 2019, forthcoming [*DOI :* 10.1017/S0960129518000439], https://hal.inria.fr/hal-01968154

[7] J. COURTIEL, K. YEATS, N. ZEILBERGER. *Connected chord diagrams and bridgeless maps*, in "The Electronic Journal of Combinatorics", November 2019, https://hal.archives-ouvertes.fr/hal-01650141

[8] Q. HEATH, D. MILLER. *A proof theory for model checking*, in "Journal of Automated Reasoning", December 2019, vol. 63, n° 4, pp. 857-885 [*DOI :* 10.1007/s10817-018-9475-3], https://hal.inria.fr/hal-01814006

[9] R. KUZNETS, L. STRASSBURGER. *Maehara-style modal nested calculi*, in "Archive for Mathematical Logic", May 2019, vol. 58, n° 3-4, pp. 359-385 [*DOI :* 10.1007/s00153-018-0636-1], https://hal.inria.fr/hal-01942240

[10] D. MILLER. *Mechanized metatheory revisited*, in "Journal of Automated Reasoning", October 2019, vol. 63, n° 3, pp. 625-665 [*DOI :* 10.1007/s10817-018-9483-3], https://hal.inria.fr/hal-01884210

[11] D. MILLER. *Reciprocal Influences Between Proof Theory and Logic Programming*, in "Philosophy & Technology", August 2019 [*DOI :* 10.1007/s13347-019-00370-x], https://hal.inria.fr/hal-02368867

[12] L. STRASSBURGER. *Deep inference and expansion trees for second-order multiplicative linear logic*, in "Mathematical Structures in Computer Science", September 2019, vol. 29, pp. 1030-1060 [*DOI :* 10.1017/S0960129518000385], https://hal.inria.fr/hal-01942410

[13] L. STRASSBURGER. *On the decision problem for MELL*, in "Theoretical Computer Science", May 2019, vol. 768, pp. 91-98 [*DOI :* 10.1016/J.TCS.2019.02.022], https://hal.inria.fr/hal-02386746

[14] L. STRASSBURGER. *The problem of proof identity, and why computer scientists should care about Hilbert's 24th problem*, in "Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences", January 2019, vol. 377, n° 2140, 20180038 p. [*DOI :* 10.1098/RSTA.2018.0038], https://hal.inria.fr/hal-02475417

International Conferences with Proceedings

[15] B. ACCATTOLI, C. FAGGIAN, G. GUERRIERI. *Factorization and Normalization, Essentially*, in "APLAS 2019 - 17th Asian Symposium on Programming Languages and Systems", Bali, Indonesia, Springer, December 2019 [*DOI :* 10.1007/978-3-030-34175-6_9], https://hal.archives-ouvertes.fr/hal-02411556

[16] M. ACCLAVIO, L. STRASSBURGER. *On Combinatorial Proofs for Logics of Relevance and Entailment*, in "WoLLIC 2019 - 26th International Workshop on Logic, Language, Information, and Computation", Utrecht, Netherlands, June 2019, pp. 1-16 [*DOI :* 10.1007/978-3-662-59533-6_1], https://hal.inria.fr/hal-02390426

[17] M. ACCLAVIO, L. STRASSBURGER. *On Combinatorial Proofs for Modal Logic*, in "TABLEAUX 2019 - 28t International Conference on Automated Reasoning with Analytic Tableaux and Related Methods", London, United Kingdom, Springer, September 2019, pp. 223-240 [*DOI :* 10.1007/978-3-030-29026-9_13], https://hal.inria.fr/hal-02390400

[18] R. BLANCO, D. MILLER, A. MOMIGLIANO. *Property-Based Testing via Proof Reconstruction*, in "PPDP 2019 - 21st International Symposium on Principles and Practice of Programming Languages", Porto, Portugal, ACM Press, October 2019, pp. 1-13 [*DOI :* 10.1145/3354166.3354170], https://hal.inria.fr/hal-02368931

[19] K. CHAUDHURI, M. MANIGHETTI, D. MILLER. *A Proof-Theoretic Approach to Certifying Skolemization*, in "CPP 2019 - 8th ACM SIGPLAN International Conference", Cascais, Portugal, ACM Press, January 2019, pp. 78-90 [*DOI :* 10.1145/3293880.3294094], https://hal.inria.fr/hal-02368946

[20] U. GÉRARD, D. MILLER, G. SCHERER. *Functional programming with λ-tree syntax*, in "PPDP 2019 - 21st International Symposium on Principles and Practice of Programming Languages", Porto, Portugal, ACM Press, October 2019, pp. 1-16 [*DOI :* 10.1145/3354166.3354177], https://hal.inria.fr/hal-02368906

[21] W. HEIJLTJES, D. J. D. HUGHES, L. STRASSBURGER. *Proof Nets for First-Order Additive Linear Logic*, in "FSCD 2019 - 4th International Conference on Formal Structures for Computation and Deduction", Dortmund, Germany, Proceedings of 4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019), June 2019, vol. 131, pp. 22:1-22:22 [*DOI :* 10.4230/LIPICS.FSCD.2019.22], https://hal.inria.fr/hal-02386942

[22] B. RALPH, L. STRASSBURGER. *Towards a Combinatorial Proof Theory*, in "TABLEAUX 2019 - 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods", London, United Kingdom, August 2019, pp. 259-276 [*DOI :* 10.1007/978-3-030-29026-9_15], https://hal.inria.fr/hal-02390417

[23] L. STRASSBURGER, W. HEIJLTJES, D. J. D. HUGHES. *Intuitionistic proofs without syntax*, in "LICS 2019 - 34th Annual ACM/IEEE Symposium on Logic in Computer Science", Vancouver, Canada, IEEE, June 2019, pp. 1-13 [*DOI :* 10.1109/LICS.2019.8785827], https://hal.inria.fr/hal-02386878

National Conferences with Proceedings

[24] S. COLIN, R. LEPIGRE, G. SCHERER. *Unboxing Mutually Recursive Type Definitions in OCaml*, in "JFLA 2019 - 30 èmes journées francophones des langages applicatifs", Les Rousses, France, January 2019, https://arxiv.org/abs/1811.02300 , https://hal.inria.fr/hal-01929508

Conferences without Proceedings

[25] B. ACCATTOLI. *A Fresh Look at the λ-Calculus*, in "FSCD 2019 - 4th International Conference on Formal Structures for Computation and Deduction", Dortmund, Germany, June 2019 [*DOI :* 10.4230/LIPICS.FSCD.2019.1], https://hal.archives-ouvertes.fr/hal-02415786

[26] B. ACCATTOLI, A. CONDOLUCI, G. GUERRIERI, C. S. COEN. *Crumbling Abstract Machines*, in "PPDP 2019 - 21st International Symposium on Principles and Practice of Programming Languages", Porto, Portugal, October 2019 [*DOI :* 10.1145/3354166.3354169], https://hal.archives-ouvertes.fr/hal-02415766

[27] B. ACCATTOLI, G. GUERRIERI, M. LEBERLE. *Types by Need*, in "ESOP 2019 - 28th European Symposium on Programming", Prague, Czech Republic, April 2019 [*DOI :* 10.1007/978-3-030-17184-1_15], https://hal.archives-ouvertes.fr/hal-02415758

[28] A. CONDOLUCI, B. ACCATTOLI, C. S. COEN. *Sharing Equality is Linear*, in "PPDP 2019 - 21st International Symposium on Principles and Practice of Programming Languages", Porto, Portugal, October 2019 [*DOI :* 10.1145/3354166.3354174], https://hal.archives-ouvertes.fr/hal-02415769

Books or Proceedings Editing

[29] B. ACCATTOLI, C. OLARTE (editors). *Preface*, The proceedings of LSFA 2018, the 13th Workshop on Logical and Semantic Frameworks with Applications (LSFA'18), Elsevier, August 2019, vol. 344, pp. 1-2 [*DOI :* 10.1016/J.ENTCS.2019.07.001], https://hal.archives-ouvertes.fr/hal-02415802

Other Publications

[30] S. MARIN, M. MORALES, L. STRASSBURGER. *A fully labelled proof system for intuitionistic modal logics*, September 2019, working paper or preprint, https://hal.inria.fr/hal-02390454

[31] A. A. TUBELLA, L. STRASSBURGER. *Introduction to Deep Inference*, August 2019, Lecture, https://hal.inria.fr/hal-02390267

References in notes

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

[33] 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

[34] 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

[35] 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

[36] 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

[37] 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

[38] 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

[39] 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

[40] 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

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

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

[43] 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.csl.sri.com/users/sgl/Work/Reports/TTSC09.pdf

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

[45] 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

[46] 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

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

[48] 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

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

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

[51] 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

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

[53] 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

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

[55] 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

[56] 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

[57] 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^{o} 1632, pp. 202–206

[58] 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

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

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

[61] 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