IN PARTNERSHIP WITH:
CNRS

Ecole Polytechnique

Activity Report 2018

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. - 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]
Beniamino Accattoli [Inria, Researcher]
Kaustuv Chaudhuri [Inria, Researcher]
François Lamarche [Inria, Senior Researcher]
Stéphane Graham-Lengrand [CNRS, Researcher]
Gabriel Scherer [Inria, Researcher]
Lutz Straßburger [Inria, Researcher, HDR]

Post-Doctoral Fellow

Matteo Acclavio [Inria, until Nov 2018]

PhD Students

Andrea Condoluci [University of Bologna (Italy), until May 2018]
Ulysse Gerard [Inria]
Maico Carlos Leberle [Inria]
Matteo Manighetti [Inria]
François Thiré [Ecole Normale Supérieure Cachan]

Interns

Marianela Evelyn Morales Elena [Inria, until Mar 2018]
Alban Reynaud [Ecole Normale Supérieure Lyon, from Jun 2018 until Jul 2018]

Administrative Assistant
Maeva Jeannot [Inria]

Visiting Scientists
Gopalan Nadathur [University of Minnesota (USA), from Oct 2018 until Nov 2018]
Carlos Olarte [Federal University of Rio Grande do Norte (Brazil), from Oct 2018 until Nov 2018]
Elaine Pimentel [Federal University of Rio Grande do Norte (Brazil), from Oct 2018 until Nov 2018]

External Collaborator
Andrea Condoluci [University of Bologna (Italy), from Jun 2018 until Aug 2018]

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 [57], and normalization in different logics can justify the design of new and different functional programming languages [30].
- *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 [61], and different proof search strategies can be used to justify the design of new and different logic programming languages [60].

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 [32] 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 [70] 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 [66] 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 [67], 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 [59] [58] [62] [71], and on more recent work by Gacek, Miller, and Nadathur [44] [43]. 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* [33] [35] [37] [38] [46] [55] [56] 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 [48], [50] 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 [45] for linear logic but also in Robinson’s proof nets [68] 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., [54][2] leading to the notion of *atomic flow* and [49]. 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 [52] 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 [34] (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 [31].

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. Trustworthy implementations of theorem proving techniques

The production of real-world verified software has made it necessary to integrate results coming from different theorem provers in a single certification package. One approach to this integration task is 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 [13].
- 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* [29] 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 [64], [69], [73], [53]) are designed for quantifier-free problems, and quantifiers there are dealt with incomplete *clause instantiation* methods or *trigger*-based techniques [41]. 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* [63]. In [47], we argued that our solutions in particular provide a way to fulfil the "Strategy Challenge for SMT-solving" set by De Moura and Passmore [74].

4.2. Principled computation for strong lambda-calculi

The application domain of the *cost models and abstract machines for functional programs* line of work—when *application* is intended in concrete terms—is the implementation of proof assistants.

Both functional languages and proof assistants rely on the λ-calculus has reference model. Functional languages are built on the *weak* λ-calculus (where evaluation does not enter function bodies) whose theory is simple and whose implementation has been widely explored in the last decades. Proof assistants instead require the full power of the *strong* λ-calculus, whose theory is more involved and whose implementation has mostly been neglected by the literature.

The study of reasonable cost models naturally leads to a refined theory of implementations, where different techniques and optimisations are classified depending on their complexity (with respect to the cost model). This direction is particularly relevant for the strong λ-calculus, for which most implementations are developed in a *ad-hoc* way.

The theoretical study in particular pointed out that all available proof assistants are implemented following unreasonable implementation schemas, where *unreasonable* here means with potentially exponential overhead with respect to the number of steps in the calculus.

Beniamino Accattoli collaborates with Bruno Barras—one of the implementors of *Coq*, the most used proof assistant—and Claudio Sacerdoti Coen—one of the implementors of *Matita*—in order to develop a fine theory of implementation for proof assistants.

If *applications* are intended also at a more theoretical level, the study of reasonable cost models is also applicable to the development of quantitative denotational semantics, to higher-order approaches to complexity theory, and to implicit computational complexity.

5. Highlights of the Year

5.1. Highlights of the Year

D. Miller has been made General Chair of the LICS Conference Series for three years, starting July 2018.

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
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/

6.5. 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 [17]. 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.

The operational semantics of MLTS is given using natural semantics for evaluation. We shall view such natural semantics as a logical theory with a rich logic that includes both nominal abstraction and the ∇-quantifier: as a result, the natural semantic specification of MLTS can be given a succinct and elegant presentation.

We have developed a number of examples of how this new programming language can be used. Some of the most convincing of these examples are programs that manipuate untyped λ-terms. A web-based implementation of an MLTS interpreter is available to anyone with a modern web browser: simply visit https://trymlts.github.io/. Small MLTS programs can be composed and executed using that interpreter.

7.2. Proof theory for model checking

Participant: Dale Miller.

While model checking has often been considered as a practical alternative to building formal proofs, we have argued that the theory of sequent calculus proofs can be used to provide an appealing foundation for model checking [7]. Given that the emphasis of model checking is on establishing the truth of a property in a model, our framework concentrates on *additive* inference rules since these provide a natural description of truth values via inference rules. Unfortunately, using these rules alone can force the use of inference rules with an infinite number of premises. In order to accommodate more expressive and finitary inference rules, *multiplicative* rules must be used, but limited to the construction of *additive synthetic inference rules*: such synthetic rules are described using the proof-theoretic notions of polarization and focused proof systems. This framework provides a natural, proof-theoretic treatment of reachability and non-reachability problems, as well as tabled deduction, bisimulation, and winning strategies. (Q. Heath collaborated on several parts of this research effort.)

7.3. From syntactic proofs to combinatorial proofs

Participants: Matteo Acclavio, Lutz Straßburger.

We continued our research on combinatorial proofs as a notion of proof identity for classical logic. We managed to extend our results from last year: We show for various syntactic formalisms including sequent calculus, analytic tableaux, and resolution, how they can be translated into combinatorial proofs, and which notion of identity they enforce. This allows the comparison of proofs that are given in different formalisms.

These results have been presented at the MLA workshop ins Kanazawa and the IJCAR conference in Oxford, published in [25].

7.4. Proof nets for first-order additive linear logic

Participant: Lutz Straßburger.

In a joint work with Willem Heijltjes (University of Bath) and Dominic Hughes (UC Berkeley) we present canonical proof nets for first-order additive linear logic, the fragment of linear logic with sum, product, and first-order universal and existential quantification. We present two versions of our proof nets. One, witness nets, retains explicit witnessing information to existential quantification. For the other, unification nets, this information is absent but can be reconstructed through unification. Unification nets embody a central contribution of the paper: first-order witness information can be left implicit, and reconstructed as needed. Witness nets are canonical for first-order additive sequent calculus. Unification nets in addition factor out any inessential choice for existential witnesses. Both notions of proof net are defined through coalescence, an additive counterpart to multiplicative contractibility, and for witness nets an additional geometric correctness criterion is provided. Both capture sequent calculus cut-elimination as a one-step global composition operation.

These results are published in [26] and have been presented at the First workshop of the Proof Society in Ghent and at the 3rd FISP workshop in Vienna.

7.5. On the Decision Problem for MELL

Participant: Lutz Straßburger.

The decision problem for multiplicative exponential linear logic (MELL) is one of the most important open problems in the are of linear logic. in 2015 there has been an attempt by Bimbò to prove the decidability of MELL. However, we have found several mistakes in that work, and the main mistake is so serious that there is no obvious fix, and therefore the decidability of MELL remains to be open. As a side effect, our work contains a complete (syntactic) proof of the decidability of the relevant version of MELL, that is the logic obtained from MELL by replacing the linear logic contraction rule by a general unrestricted version of the contraction rule. These results are presented in [27].

7.6. OCaml metatheory

Participant: Gabriel Scherer.

We worked on the evolution of advanced features of the OCaml programming language, designing static analyses to ensure their safety through a scientific study their metatheory. Specifically, we worked on unboxed type declarations (during an internship by Simon Colin, M1 from École Polytechnique) and recursive value definitions (during an internship by Alban Reynaud, L3 from ENS Lyon). The two internships and followup work each resulted in both a change proposal to the OCaml implementation and a submission to an academic conference.

7.7. Merlin: understanding a language server

Participant: Gabriel Scherer.

Thomas Réfis (Jane Street) and Frédéric Bour maintain the Merlin language server of OCaml, a tool that provides language-aware features to text editors. We collaborated with them on dissecting the tool and explaining its design and evolution ([4]); the similarities and differences with usual compiler frontends may inform future language implementation work, and our language-agnostic presentation may be of use to tool designers for other languages and proof assistants.

7.8. Language interoperability: ML and a Linear language

Participant: Gabriel Scherer.

In a programming system where programs are created in one programming language, we consider the addition of another programming language that interoperates with the first – and the reimplementation of some library/system functions in this new language. This can increase expressivity, but it could also break some assumptions made by programmers. Typically, adding a bridge to C or assembly code can introduce memory-unsafe code in a previously-safe system. In [18], we formalize a notion of "graceful" interoperability between two languages in this setting, determined by full abstraction, that is, preservation of equational reasoning. We instantiate this general idea by extending ML with an advanced expert language with linear types and linear mutable cells.

7.9. First-class simultaneous substitutions in the two-level logic approach

Participant: Kaustuv Chaudhuri.

The *two-level logic approach* that underlies the Abella prover is excellent at reasoning about the inductive structure of terms with binding constructs, such as λ-terms from the λ-calculus. However, there is no built in support in Abella for reasoning about the inductive structure of (simultaneous) substitutions. This lack of this kind of support is often criticized in the λ-tree syntax representational style that is used in Abella; indeed, in a number of other systems based on this style, support for reasoning about substitutions is explicitly added into the trusted kernel. In [14] we show how to formalize substitutions in Abella in a fluent and high level manner, where all the meta-theory can be proven in a straightforward manner. We illustrate its use in giving a clean formulation of fact that the Howe extension of applicative similarity is a pre-congruence, a standard result from the meta-theory of the λ-calculus that requires sophistication in treating simultaneous substitutions.

7.10. Hybrid Linear Logic, revisited

Participant: Kaustuv Chaudhuri.

Hybrid Linear Logic (HyLL) was proposed by Chaudhuri and Despeyroux in 2010 as a meta-logic for reasoning about constrained transition systems, with applications to a number of domains including formal molecular biology [36]. This logic is an extension of (intuitionistic) linear logic with hybrid connectives that can reason about monoidal constraint domains such as instants of time or rate functions. *Linear logic with subexponential* is a different extension of linear logic that has been proposed as a mechanism for capturing certain well known constrained settings such as bigraphs [39] or concurrent constraint programming [65]. In a paper accepted to MSCS [5] we show how to relate these two extensions of linear logic by giving an embedding of HyLL into linear logic with subexponentials. Furthermore, we show that subexponentials are able to give an adequate encoding of CTL$*$, which is beyond the expressive power of HyLL. Thus, subexponentials appear to be the better choice as a foundation for constraints in linear logic.

7.11. Proof Nets and the Linear Substitution Calculus

Participant: Beniamino Accattoli.

This work [21] belongs to line of work *Cost Models and Abstract Machines for Functional Programs*, supported by the ANR project COCA HOLA, and it has been published in the proceedings of the international conference ICTAC 2018.

The *Linear Substitution Calculus* (LSC) is a refinement of the λ-calculus that is crucial for the study of cost models for functional programs, as it enables a sharp and yet simple decomposition of the evaluation of λ-terms, and it is employed in the proof of various results about cost models in the literature.

In this work we show that the LSC is isomorphic to the linear logic representation of the λ-calculus. More precisely, it is isomorphic to the *proof nets* presentation of such a fragment of linear logic. Proof nets are a graphical formalism, which—as most graphical formalisms—is handy for intuitions but not prone to formal reasoning. The result is relevant because it allows to manipulate formally a graphical formalism (proof nets) by means of an ordinary term syntax (the LSC).

7.12. Tight Typings and Split Bounds

Participants: Beniamino Accattoli, Stéphane Graham-Lengrand.

This joint work with Delia Kesner (Paris Diderot University) [12] belongs to line of work *Cost Models and Abstract Machines for Functional Programs*, supported by the ANR project COCA HOLA, and it has been published in the proceedings of the international conference ICFP 2018.

Intersection types are a classic tool in the study of the λ-calculus. They are known to characterise various termination properties.

It is also well-known that *multi types*, a variant of intersection types strongly related to linear logic, also characterise termination properties. Typing derivation of multi types, moreover, provide quantitative information such as the number of evaluation step and the size of the results, as first shown by de Carvalho.

In this work we provide some new results on this line of work, notably we provide the first quantitative study via multi types of the leftmost and linear head evaluation strategies. Moreover, we show that our approach covers also the other cases in the literature.

7.13. Types of Fireballs

Participant: Beniamino Accattoli.

This joint work with Giulio Guerrieri (Bologna University) [22] belongs to line of work *Cost Models and Abstract Machines for Functional Programs*, supported by the ANR project COCA HOLA, and it has been published in the proceedings of the international conference APLAS 2018.

The theory of the call-by-value λ-calculus has mostly been developed for *closed* programs, that is, programs without free variables. In the last few years, the authors dedicated considerable efforts to extend it to open terms, that is the case relevant for the implementation of proof assistants. The simplest presentation of the call-by-value λ-calculus for open terms is the *fireball calculus*.

In this work we extend the quantitative study via multi types mentioned in *Tight Typings and Split Bounds* to the fireball calculus.

7.14. Decision procedures for intuitionistic propositional logic

Participant: Stéphane Graham-Lengrand.

Provability in intuitionistic propositional logic is decidable and, as revealed by the works of, e.g., Vorobev [72], Hudelmaier [51] and Dyckhoff [42], proof theory can provide natural decision procedures, which have been implemented in various software. More precisely, a decision procedure is obtained by performing direct root-first proof-search in (different variants of) a sequent calculus system called LJT (aka G4ip); termination is ensured by a property of the sequent calculus called depth-boundedness.

Independently from this, Claessen and Rosen [40] recently proposed a decision procedure for the same logic, based on a methodology used in the field of Satisfiability-Modulo-Theories (SMT). Their implementation clearly outperforms the sequent-calculus-based implementations.

In 2018 we managed to establish of formal connection between the G4ip sequent calculus and the algorithm from [40], revealing the features that they share and the features that distinguish them. This connection is interesting because it gives a proof-theoretical light on SMT-solving techniques, and it opens the door to the design of an intuitionistic version of the CDCL algorithm used in SAT-solvers, which decides provability in classical logic.

7.15. Admissible Tools in the Kitchen of Intuitionistic Logic

Participants: Matteo Manighetti, Andrea Condoluci.

In this work we study the computational meaning of the inference rules that are admissible, but not derivable, in intuitionistic logic [16].

An inference rule is admissible for a logic if whenever its antecedent is derivable, its conclusion was already derivable without the rule. In classical logic, whenever this is the case, then also the implication between antecedent and conclusion is derivable. The notion of an admissible rule is therefore internalized in the logic.

This is not the case for intuitionistic logic, and some rules that are admissible are not derivable: therefore they need reasoning outside the usual intuitionistic logic in order to be reduced to purely intuitionistic derivation.

In this work we propose a proof system with term annotations and reduction rules to give a computational meaning to these reductions.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. 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.

8.1.2. 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.

8.2. International Research Visitors

8.2.1. Internships

Simon Colin did an M1 internship supervised by G. Scherer, conducting a static analysis to check the safety, in OCaml, of unboxing annotations on type declarations.

Alban Reynaud did an L3 internship supervised by G. Scherer, conducting a static analysis to check the safety, in OCaml, of recursive value declarations.

8.2.2. Visits to International Teams

8.2.2.1. Research Stays Abroad

S. Graham-Lengrand was an International Fellow at SRI International, for 25 months over a period of three years between 2015 and 2018.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

D. Miller is the General Chair of LICS (Logic In Computer Science), starting July 2018.

9.1.1.2. Member of the Organizing Committees

D. Miller is on the Steering Committee for the FSCD conference series and the CPP conference series.

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

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

B. Accattoli co-chaired LSFA 2018: 13th Workshop on Logical and Semantic Frameworks with Applications, Fortaleza, Brazil, September 26-28, 2018.

G. Scherer chaired ML2018: the ML Family Workshop 2018 in Saint Louis, US, on Friday September 28th 2018.

L. Straßburger chaired TYDI 2018: Workshop on "Twenty Years of Deep Inference" in Oxford July 7, 2018.

9.1.2.2. Member of the Conference Program Committees

B. Accattoli was on the PPDP 2018 Program Committe: 20th International Symposium on Principles and Practice of Declarative Programming, Frankfurt, Germany, 3–5 September 2018.

S. Graham-Lengrand was on the LFMTP 2018 Program Comittee: Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, Oxford, UK, 7 July 2018.

L. Straßburger was on the PC for LACompLing 2018: Symposium on Logic and Algorithms in Computational Linguistics, Stockholm, 28–31 August 2018

D. Miller was on the program committee for IJCAR-2018: 9th International Joint Conference on Automated Reasoning, Oxford, 14-17 July 2018.

D. Miller was a member of the jury for selecting the 2018 Ackermann Award (the EACSL award for outstanding doctoral dissertation in the field of Logic in Computer Science).

Member of the EATCS Distinguished Dissertation Award Committee since March 2013.

G. Scherer was on the POPL 2019 Program Committee: Principles Of Programming Languages, 13-19 January 2019 Cascais/Lisbon, Portugal

9.1.2.3. Reviewer

G. Scherer reviewed for Computer Science Logic (CSL).

L. Straßburger was reviewer for the following conferences:

- LICS 2018
- IJCAR 2018
- FSCD 2018
- AiML 2018
- ARQNL 2018

B. Accattoli reviewed for LICS 2018, FSCD 2018, PPDP 2018, LSFA 2018.

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

D. Miller is on the editorial board of the following journals:

- Journal of Automated Reasoning
- Journal of Applied Logics

9.1.3.2. Reviewer - Reviewing Activities

G. Scherer reviewed for Mathematical Structures in Computer Science (MSCS).

L. Straßburger was reviewer for the following journals:

- Transactions on Computational Logic, ToCL (2x)
- Logical Methods in Computer Science, LMCS
- Mathematical Structures in Computer Science, MSCS
- Journal of Logic, Language and Information, JLLI
- Journal of Automated Reasoning, JAR
- Notre Dame Journal of Formal Logic, NDJFL

B. Accattoli reviewed for Logical Methods in Computer Science (LMCS) and Theoretical Computer Science (TCS).

9.1.4. Invited Talks

S. Graham-Lengrand gave an invited talk at the JFLA 2018 (January), and an invited lecture series at the 8th Summer School on Formal Techniques (May).

B. Accattoli gave an invited talk at the *IFIP Working Group 1.6: Rewriting* on July 8 2018 in Oxford, Uk.

D. Miller was an invited speaker and panelist at the Workshop on Proof Theory and its Applications, 6–7 September 2018 in Ghent, Beligum.

D. Miller gave a colloquim talk at the Technical University of Vienna on 31 October 2018 and at the Cyber Security Lab, NTU, Singapore, 21 March 2018.

9.1.5. Scientific Expertise

G. Scherer participated to a scientific expertise of the implementation of the Tezos blockchain – implemented in OCaml.

9.1.6. Research Administration

L. Straßburger was reviewer for the NWO (Netherlands Organisation for Scientific Research).

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Licence : G. Scherer, Programmation Fonctionnelle, 50, L1, Paris 8 (Vincennes / Saint Denis), France

Licence : K. Chaudhuri, Programmation avancée en OCaml, 40 hours eq TD, L3, École polytechnique, France

Bachelor : K. Chaudhuri, Computer programming, principal instructor, École polytechnique, France (This program has no direct equivalent in the traditional French university system; the closest would be L1.)

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.

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

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

Summer School: B. Accattoli, "The Complexity of Beta-reduction", 4.5h, International School on Rewriting (ISR) 2018, Cali, Colombia.

9.2.2. Supervision

PhD : Sonia Marin, Modal Proof Theory through a Focused Telescope, Université Paris-Saclay, 30 January 2018, encadrant(s): Lutz Straßburger, Dale Miller.

PhD in progress: Ulysse Gérard and Matteo Manighetti supervised by Dale Miller.

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

PhD in progress: Maico Leberle supervised by Dale Miller and Beniamino Accattoli.

9.2.3. Juries

D. Miller was the a reporter for the PhD juries of Michael Lettmann (TU Vienna, 30 October 2018)

9.3. Popularization

L. Straßburger serves as member of the “commission développement technologique (CDT)” for Inria Saclay–Île-de-France (since June 2012).

F. Lamarche was site co-ordinator for the Activity Report for Inria Saclay–Ile-de-France.

9.3.1. Interventions

G. Scherer and M. Manighetti participated the “Fête de la Science” exhibit at Inria Saclay on the whole day of October 11th, 2018. They manned an activity on sorting algorithms for colored plastic pieces.

9.3.2. Internal action

G. Scherer spoke at the “Unithé ou café” meeting, a Saclay-internal popularization meeting, on February 1st, 2018.

10. 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

Articles in International Peer-Reviewed Journals

[3] M. ACCLAVIO. *Proof Diagrams for Multiplicative Linear Logic: Syntax and Semantics*, in "Journal of Automated Reasoning", May 2018 [*DOI :* 10.1007/s10817-018-9466-4], https://hal.archives-ouvertes.fr/hal-01931400

[4] F. BOUR, T. RÉFIS, G. SCHERER. *Merlin: a language server for OCaml (experience report)*, in "Proceedings of the ACM on Programming Languages", July 2018, vol. 2, n° ICFP, pp. 1 - 15, https://arxiv.org/abs/1807.06702 [*DOI :* 10.1145/3236798], https://hal.inria.fr/hal-01929161

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

[6] 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", 2018, vol. 2, n° POPL, https://arxiv.org/abs/1711.03050 [*DOI :* 10.1145/3158137], https://hal.inria.fr/hal-01646765

[7] Q. HEATH, D. MILLER. *A proof theory for model checking*, in "Journal of Automated Reasoning", 2018 [*DOI :* 10.1007/s10817-018-9475-3], https://hal.inria.fr/hal-01814006

[8] J. KOPPEL, G. SCHERER, A. SOLAR-LEZAMA. *Capturing the Future by Replaying the Past Functional Pearl*, in "Proceedings of the ACM on Programming Languages", July 2018, vol. 2, n° ICFP, pp. 1 - 29 [*DOI :* 10.1145/3236771], https://hal.inria.fr/hal-01929178

[9] R. KUZNETS, L. STRASSBURGER. *Maehara-style modal nested calculi*, in "Archive for Mathematical Logic", July 2018 [*DOI :* 10.1007/s00153-018-0636-1], https://hal.inria.fr/hal-01942240

[10] D. MILLER. *Mechanized metatheory revisited*, in "Journal of Automated Reasoning", 2018, https://hal.inria.fr/hal-01884210

[11] L. STRASSBURGER. *Deep inference and expansion trees for second-order multiplicative linear logic*, in "Mathematical Structures in Computer Science", November 2018, pp. 1-31 [*DOI :* 10.1017/S0960129518000385], https://hal.inria.fr/hal-01942410

International Conferences with Proceedings

[12] B. ACCATTOLI, S. GRAHAM-LENGRAND, D. KESNER. *Tight typings and split bounds*, in "23rd ACM International Conference on Functional Programming", St Louis, United States, M. FLATT (editor), September 2018, vol. 2, n° ICFP, pp. 1 - 30, https://arxiv.org/abs/1807.02358 [*DOI :* 10.1145/3236789], https://hal.archives-ouvertes.fr/hal-01936141

[13] M. P. BONACINA, S. GRAHAM-LENGRAND, N. SHANKAR. *Proofs in conflict-driven theory combination*, in "Proceedings of the 7th International Conference on Certified Programs and Proofs (CPP'18)", Los Angeles, United States, J. ANDRONICK, A. FELTY (editors), ACM Press, January 2018, vol. 18 [*DOI :* 10.1145/3167096], https://hal.archives-ouvertes.fr/hal-01935595

[14] K. CHAUDHURI. *A Two-Level Logic Perspective on (Simultaneous) Substitutions*, in "CPP 2018 - Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs", Los Angeles, United States, January 2018, https://hal.inria.fr/hal-01968139

[15] K. CHAUDHURI, U. GÉRARD, D. MILLER. *Computation-as-deduction in Abella: work in progress*, in "13th international Workshop on Logical Frameworks and Meta-Languages: Theory and Practice", Oxford, United Kingdom, July 2018, https://hal.inria.fr/hal-01806154

[16] A. CONDOLUCI, M. MANIGHETTI. *Admissible Tools in the Kitchen of Intuitionistic Logic*, in "Seventh International Workshop on Classical Logic and Computation (CL&C 2018)", Oxford, United Kingdom, Electronic Proceedings in Theoretical Computer Science, July 2018, vol. 281, pp. 10-23, https://hal.inria.fr/hal-01870112

[17] U. GÉRARD, D. MILLER. *Functional programming with $\lambda-tree$ syntax: a progress report*, in "13th international Workshop on Logical Frameworks and Meta-Languages: Theory and Practice", Oxford, United Kingdom, July 2018, https://hal.inria.fr/hal-01806129

[18] G. SCHERER, M. NEW, N. RIOUX, A. AHMED. *FabULous Interoperability for ML and a Linear Language*, in "International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)", Thessaloniki, Greece, C. BAIER, U. D. LAGO (editors), FabOpen image in new windowous Interoperability

for ML and a Linear Language, Springer, April 2018, vol. LNCS - Lecture Notes in Computer Science, n° 10803, https://arxiv.org/abs/1707.04984 [*DOI :* 10.1007/978-3-319-89366-2_8], https://hal.inria.fr/hal-01929158

National Conferences with Proceedings

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

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

Conferences without Proceedings

[21] B. ACCATTOLI. *Proof Nets and the Linear Substitution Calculus*, in "15th International Colloquium on Theoretical Aspects of Computing (ICTAC 2018)", Stellenbosch, South Africa, October 2018, https://hal.archives-ouvertes.fr/hal-01967532

[22] B. ACCATTOLI, G. GUERRIERI. *Types of Fireballs*, in "16th Asian Symposium on Programming Languages and System (APLAS 2018)", Wellington, New Zealand, December 2018, https://hal.archives-ouvertes.fr/hal-01967531

[23] F. BOBOT, S. GRAHAM-LENGRAND, B. MARRE, G. BURY. *Centralizing equality reasoning in MCSAT*, in "16th International Workshop on Satisfiability Modulo Theories (SMT 2018)", Oxford, United Kingdom, R. DIMITROVA, V. D'SILVA (editors), July 2018, https://hal.archives-ouvertes.fr/hal-01935591

[24] S. GRAHAM-LENGRAND, M. FÄRBER. *Guiding SMT solvers with Monte Carlo Tree Search and neural networks*, in "Third Conference on Artificial Intelligence and Theorem Proving (AITP'2018)", Aussois, France, T. C. HALES, C. KALISZYK, S. SCHULZ, J. URBAN (editors), March 2018, https://hal.archives-ouvertes.fr/hal-01935578

Scientific Books (or Scientific Book chapters)

[25] M. ACCLAVIO, L. STRASSBURGER. *From Syntactic Proofs to Combinatorial Proofs*, in "International Joint Conference on Automated Reasoning, IJCAR 2018", Springer, June 2018, pp. 481-497, https://hal.inria.fr/hal-01942275

Research Reports

[26] W. HEIJLTJES, D. J. D. HUGHES, L. STRASSBURGER. *Proof nets for first-order additive linear logic*, Inria, September 2018, n° RR-9201, https://hal.inria.fr/hal-01867625

[27] L. STRASSBURGER. *On the Decision Problem for MELL*, Inria Saclay Ile de France, September 2018, n° 9203, https://hal.inria.fr/hal-01870148

Other Publications

[28] D. MILLER. *Influences between logic programming and proof theory*, March 2018, HaPoP 2018: Fourth Symposium on the History and Philosophy of Programming, https://hal.inria.fr/hal-01900891

References in notes

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

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

[31] 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

[32] 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

[33] 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

[34] 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

[35] 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

[36] K. CHAUDHURI, J. DESPEYROUX. *A Hybrid Linear Logic for Constrained Transition Systems with Applications to Molecular Biology*, Inria, 2010

[37] 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

[38] 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

[39] K. CHAUDHURI, G. REIS. *An adequate compositional encoding of bigraph structure in linear logic with subexponentials*, in "Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)", M. DAVIS, A. FEHNKER, A. MCIVER, A. VORONKOV (editors), LNCS, Springer, November 2015, vol. 9450, pp. 146–161 [*DOI :* 10.1007/978-3-662-48899-7_11], http://chaudhuri.info/papers/lpar15bigraphs.pdf

[40] K. CLAESSEN, D. ROSÉN. *SAT Modulo Intuitionistic Implications*, in "Proc. of the the 20th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'15)", M. DAVIS, A. FEHNKER, A.

McIver, A. Voronkov (editors), LNCS, Springer-Verlag, November 2015, vol. 9450, pp. 622–637, http://dx.doi.org/10.1007/978-3-662-48899-7

[41] 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://smt2012.loria.fr/paper4.pdf

[42] R. Dyckhoff. *Contraction-free sequent calculi for intuitionistic logic*, in "J. of Symbolic Logic", 1992, vol. 57, n° 3, pp. 795–807

[43] 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

[44] 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

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

[46] 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° 1, http://www.lix.polytechnique.fr/~lengrand/Work/Reports/TTSC09.pdf

[47] 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

[48] A. Guglielmi. *A System of Interaction and Structure*, in "ACM Trans. on Computational Logic", 2007, vol. 8, n° 1

[49] A. Guglielmi, T. Gundersen. *Normalisation Control in Deep Inference Via Atomic Flows*, in "Logical Methods in Computer Science", 2008, vol. 4, n° 1:9, pp. 1–36, http://arxiv.org/abs/0709.1205

[50] 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

[51] J. Hudelmaier. *Bounds for Cut Elimination in Intuitionistic Logic*, Universität Tübingen, 1989

[52] D. Hughes. *Proofs Without Syntax*, in "Annals of Mathematics", 2006, vol. 164, n° 3, pp. 1065–1076

[53] 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

[54] 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

[55] 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

[56] 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

[57] 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

[58] 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

[59] 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

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

[61] 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

[62] 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

[63] 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

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

[65] C. OLARTE, V. NIGAM, E. PIMENTEL. *Subexponential concurrent constraint programming*, in "Theoretical Computer Science", 2015, http://dx.doi.org/10.1016/j.tcs.2015.06.031

[66] 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

[67] 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

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

[69] 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

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

[71] 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

[72] N. N. VOROB'EV. *A new algorithm for derivability in the constructive propositional calculus*, in "Amer. Math. Soc. Transl.", 1970, vol. 94, n^{o} 2, pp. 37–71

[73] L. M. DE MOURA, D. JOVANOVIĆ. *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

[74] 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