IN PARTNERSHIP WITH:
CNRS

Ecole Polytechnique

Activity Report 2015

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:

2.1.1. - Semantics of programming languages
2.1.11. - Proof languages
2.4.2. - Verification
2.4.3. - Proofs
7.4. - Logic in Computer Science

Other Research Topics and Application Domains:

9.4.1. - Computer science
9.4.2. - Mathematics

1. Members

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, HdR]
Lutz Straßburger [Inria, Researcher, HdR]

Engineers

Danko Ilik [Inria]
Tomer Libal [Inria]
Noam Zeilberger [Inria]

PhD Students

Roberto Blanco Martinez [Inria, ProofCert]
Hichem Chihani [Inria, ProofCert]
Ulysse Gerard [Inria, ProofCert from Apr 2015]
Quentin Heath [Ecole Polytechnique]
Sonia Marin [Inria, ProofCert]

Post-Doctoral Fellows

Taus Brock-Nannestad [Inria]
Giselle Reis [Inria]
Matthias Puech [Inria]
Hugh Paul Steele [Inria, from Dec 2015]
Marco Volpe [Inria]

Visiting Scientist

Chuck Liang [Professor, 25 May to 15 June 2015]

Administrative Assistant

Christine Aklouche [Inria]

Others

Horace Blanc [Inria, Intern, Apr 2015 until Sep 2015]

Leonardo Augusto Lima Ferreira Da Silva [Inria, Intern, Oct 2015 until March 2016]

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 [71], and normalization in different logics can justify the design of new and different functional programming languages [48].
- *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 [77], and different proof search strategies can be used to justify the design of new and different logic programming languages [75].

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 [50] 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 [85] 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 [81] 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 [82], 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 [73] [72] [78] [86], and on more recent work by Gacek, Miller, and Nadathur [4] [63]. The Parsifal team, along with our colleagues in Minneapolis, Canberra, Singapore, and Cachen, have been building two tools that exploit the novel features of this logic. These two systems are the following.

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

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

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

3.3. Developing a foundational approach to defining proof evidence

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

In order to provide our framework with the flexibility that this ambitious plan requires, we have based our design on the most recent advances within the theory of proofs. For a number of years, various team members have been contributing to the design and theory of *focused proof systems* [51] [54] [56] [57] [65] [69] [70] 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 [66], [68] is a novel methodology for presenting deductive systems. Unlike traditional formalisms like the sequent calculus, it allows rewriting of formulas deep inside arbitrary contexts. The new freedom for designing inference rules creates a richer proof theory. For example, for systems using deep inference, we have a greater variety of normal forms for proofs than in sequent calculus or natural deduction systems. Another advantage of deep inference systems is the close relationship to categorical proof theory. Due to the deep inference design one can directly read off the morphism from the derivations. There is no need for a counter-intuitive translation.

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

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

3.5. Proof nets and atomic flows

Proof nets and atomic flows are abstract (graph-like) presentations of 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 [64] for linear logic but also in Robinson's proof nets [83] 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., [6], [5] and [67]. 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.

This directly leads to the following research questions investigated by members of the Parsifal team:

- Finding (for classical logic) a notion of proof nets that is deductive, i.e., can effectively be used for doing proof search. An important property of deductive proof nets must be that the correctness can be checked in linear time. For the classical logic proof nets by Lamarche and Straßburger [6] this takes exponential time (in the size of the net).
- Studying the normalization of proofs in classical logic using atomic flows. Although there is no correctness criterion they allow to simplify the normalization procedure for proofs in deep inference, and additionally allow to get new insights in the complexity of the normalization.

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 [53] (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 [49].

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

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

4. Application Domains

4.1. Integrating a model checker and a theorem prover

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

Our first step to providing an integration of model checking and (inductive) theorem proving was the development of a strong logic, that we call $\mathcal{G}$, which extends intuitionistic logic with notions of least and greatest fixed points. We had developed the proof theory of this logic in earlier papers [4] [63]. We have now recently converted the Bedwyr system so that it formally accepts almost all definitions and theorem statements that are accepted by the inductive theorem prover Abella. Thus, these two systems are proving theorems in the same logic and their results can now be shared.

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

For more about recent progress on providing checkable proof certificates for model checking, see the web site for Bedwyr http://slimmer.gforge.inria.fr/bedwyr/.

4.2. Implementing trusted proof checkers

Traditionally, theorem provers—whether interactive or automatic—are usually monolithic: if any part of a formal development was to be done in a particular theorem prover, then the whole of it would need to be done in that prover. Increasingly, however, formal systems are being developed to integrate the results returned from several, independent and high-performing, specialized provers: see, for example, the integration of Isabelle with an SMT solver [62] as well as the Why3 and ESC/Java systems.

Within the Parsifal team, we have been working on foundational aspects of this multi-prover integration problem. As we have described above, we have been developing a formal framework for defining the semantics of proof evidence. We have also been working on prototype checkers of proof evidence which are capable of executing such formal definitions. The proof definition language described in the papers [59], [58] is currently given an implementation in the λProlog programming language [76]. This initial implementation will be able to serve as a "reference" proof checker: others who are developing proof evidence definitions will be able to use this reference checker to make sure that they are getting their definitions to do what they expect.

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

4.3. Trustworthy implementations of theorem proving techniques

Instead of integrating different provers by exchanging proof evidence and relying on a backend proof-checker, another approach to integration consists in re-implementing the theorem proving techniques as proof-search strategies, on an architecture that guarantees correctness. Focused systems can serve as the basis of such an architecture, identifying points for choice and backtracking, and 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.

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

- The first 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* [47] 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) are designed for quantifier-free problems, and quantifiers are there dealt with incomplete *clause instantiation* methods or *trigger*-based techniques [61]. 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* [79]. In [30], we argued that our solutions in particular provide a way to fulfil the "Strategy Challenge for SMT-solving" set by De Moura and Passmore [88].

5. Highlights of the Year

5.1. Highlights of the Year

Accattoli's paper with Ugo Dal Lago titled "Beta Reduction is Invariant, Indeed" that appeared in CSL-LICS 2014 was invited to a special issue of LMCS of selected papers from that meeting.

Miller ended his second and final term as the Editor-in-Chief of the ACM Transactions on Computational Logic. He remains as an Area Editor for Proof Theory.

Miller was a plenary speaker at the joint meeting of LOPSTR 2015 and PPDP 2015 (Siena, Italy) and was an invited speaker at LSFA 2015 (Natal Brazil) and in the Session on History and Philosophy of Computing at the 15th Congress of Logic, Methodology and Philosophy of Science, Helsinki.

Miller spoke at the ETH Zurich Department of Computer Science Distinguished Colloquium Series on April 20.

Graham-Lengrand gave an invited talk at the IFIP Working Group 1.6 on Term Rewriting, on the occasion of its 2015 annual meeting in Warsaw, Poland.

Accattoli was an invited speaker at the 16th International Workshop on Logic and Computational Complexity (Kyoto, Japan, 5th of July).

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.

In 2015, Abella has been extended with

- support for polymorphic definitions and theorems;
- schemas and automatically derived theorems about them;
- the ability to record and replay automated search;
- a number of new examples from process calculi, including a contributed example from Horace Blanc about relating the π-calculus and the λ-calculus.

One further development is that Abella can now be compiled into JavaScript and run completely inside any modern browser, thanks to the `js_of_ocaml` compiler from OCaml bytecode to JavaScript. We expect this to become rather crucial in popularization of Abella, particularly in a pedagogical context, since it does not require any local software installation—just a modern web browser.

- Participants: Dale Miller, Kaustuv Chaudhuri, Horace Blanc
- Partner: Department of Computer Science and Engineering, University of Minnesota
- Contact: Kaustuv Chaudhuri
- URL: http://abella-prover.org/
- Online version: http://abella-prover.org/tutorial/try

6.2. Bedwyr

Bedwyr - A proof search approach to model checking
FUNCTIONAL DESCRIPTION

Bedwyr is a generalization of logic programming that allows model checking directly on syntactic expression possibly containing 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.

- Participant: Roberto Blanco Martinez
- Contact: Quentin Heath
- 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: Tomer Libal and Giselle Reis
- Contact: Tomer Libal
- URL: https://github.com/proofcert/checkers

6.4. Mætning

KEYWORDS: Automated Theorem Proving - Intuitionistic Logic

FUNCTIONAL DESCRIPTION

Mætning is an automated theorem prover for first-order intuitionistic logic that is particularly suited for efficiently finding *disproofs*, i.e., for establishing that a given goal query is not provable. It is based on the focused inverse method [54] [74], but augmented by a mechanism for building finite approximations for infinite search spaces that nevertheless guarantee soundness of a disproof.

Mætning has been released under the terms of the MIT license.

- Participants: Taus Brock-Nannestad, Kaustuv Chaudhuri
- Contact: Kaustuv Chaudhuri
- URL: https://github.com/chaudhuri/maetning

6.5. Psyche

KEYWORDS: Proof-search - Correct-by-construction approach - Programmable 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 small kernel interacts with plugins and decision procedures. The major effort in 2015 was a complete redesign of its architecture to allow the safe cooperation of various procedures (for e.g. different theories).

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

7. New Results

7.1. The Checkers Proof Certifier

Participants: Tomer Libal, Giselle Reis, Hichem Chihani.

We presented a system description [29] of the Checkers proof certifier, which implements some of the theoretical ideas developed in the ProofCert project. This version of the system is capable of certifying a subset of the E-Prover superposition theorem prover. The system is mainly written in λProlog with a proof importing module written in Ocaml. The system is designed to allow modularity when designing the semantical translations of proof systems. For this capacity, the system supports, J. A. Robinson's resolution and the paramodulation technique of G. Robinson and L. Wos. On top of that, minimal support for some inference rules of the E-Prover was added.

7.2. Regular Patterns in Second-Order Unification

Participant: Tomer Libal.

We presented a paper [33] detailing a higher-order pre-unification procedure with improved termination over existing procedures. The classic higher-order unification procedure was presented by G. Huet in 1975 and is still used as the main unification procedure for higher-order automated theorem provers. This procedure does not terminate. In this project we have investigated the reasons for that and have shown that by choosing a specific (but complete) search strategy, an additional set of non-unifiable problems can be detected. As an example, we have shown that all unification problems generated by the Leo-III theorem prover when proving Cantor's theorem are decided by this procedure, in contrast to the classical unification procedure.

7.3. Static guarantees for message-passing computation

Participant: Stéphane Graham-Lengrand.

LCF [79] is a proof-search architecture, where search strategies are programmed via an API and successful proof-search runs are guaranteed correct, relying on the use of an abstract type `theorem`. We adapted the approach and defined principles for message-passing software architectures (where modules interact by exchanging messages), with the objective of guaranteeing message provenance and integrity. The principles rely on abstract types to *sign* messages at no run-time cost, and more generally rely on type-checking to provide static guarantees (i.e. at compile-time) that the messages produced by a trusted piece of code will not be altered or faked by an untrusted piece of code. We developed this primarily for safe theorem proving architectures, but the approach can be applied to other software architectures where modules with different levels of trust interact.

7.4. Proof-search with quantifiers and theories

Participant: Stéphane Graham-Lengrand.

We published our approach to proof-search on quantified problems in presence of one theory [22], where we identify the specifications required of the theory for the proof-search process to be sound and complete. Theories with unification procedures or quantifier elimination procedures satisfy our specifications, where *constraint streams* and *constraint projections* play a key role key. Interestingly enough, Bjorner and Janota [52] independently achieved a similar result with *model projections*. Our theory-generic approach allows a clear formulation of what it could mean to combine several quantifier-handling theories, hopefully generalising what the Nelson-Oppen combination technique does in a quantifier-free context. We recently obtained two new results towards this:

- First, the cumbersome, stream-querying, and backtracking mechanisms that were required to implement [22] have been re-expressed in a more satisfying message-passing computational framework.
- Second, we re-expressed the standard quantifier-free combination techniques, mentioned above, as a concurrent message-passing interaction between different theory-specific procedures, and simplified their proofs of correctness. This led to the major redesign of Psyche, mentioned above.

7.5. Realizability semantics of abstract focusing, formalized

Participant: Stéphane Graham-Lengrand.

In [21] we presented a parametric system for abstract focusing, building on Zeilberger's work [87], and parametrically capturing classical and intuitionistic focused systems. We presented its semantics, building on Munch-Maccagnoni's work [80], in terms of *abstract realizability models* (which were independently identified by Krivine). The goal was to emphasize the similarities and differences between focusing and realizability, in the way they exploit the *polarities* of formulae. The system and its semantics led to a substantially formalisation in the proof assistant Coq.

7.6. The Meta-Theory of Bisimulation-Up-To

Participants: Kaustuv Chaudhuri, Matteo Cimini, Dale Miller.

The method of proof by bisimulation has proved to be a very successful technique for showing the equivalence of processes. Unfortunately, in process calculi with infinite transition systems, such as in calculi with a replication operator, finding a bisimulation requires exploring an infinite search space, which moreover often tends to have rather intricate and complex structure. One way to combat this complexity—i.e., reduce the size of candidate bisimulation sets—is to identify redundancies among their members and then to replace redundant classes by unique inhabitants. This yields families of *bisimulation-up-to* proof methods that are parametric over the redundancy relation. For instance, if we consider bisimilarity itself as the redundancy, then we obtain *bisimulation up to bisimilarity*; with this relation, the singleton set $\{(!a, !!a)\}$ is a candidate set for showing that the processes $!a$ and $!!a$ are bisimilar, for example, when the bisimulation set with redundancies is infinite.

Since *a priori* there is no restriction on such redundancy relations, a key theoretical question is when a bisimulation-up-to relation is *sound*, i.e., that it is contained in a bisimulation. In the literature there have been a number of techniques proposed for showing soundness, but they often require the use of complex reasoning about lattices of fixed points. In [19] (CPP'15) we show how to use the built-in coinduction facilities of the Abella theorem prover to produce comparatively lightweight proofs of the soundness of many common bisimulation-up-to techniques for CCS and the π-calculus. A key feature of our approach is that we can use the facilities already provided by the Abella system for reasoning about the binding constructs for the π-calculus.

7.7. Characterizing Independence in Type Theory

Participants: Kaustuv Chaudhuri, Yuting Wang.

In formal proof languages based on type theory, it is often the case that a theorem is proved for a certain kind of typing context, but needs to be used in a different context. For example, theorems about natural numbers may be proved in an empty typing context, since the type of natural numbers contains no higher-order features (i.e., natural numbers are closed), but we may need to use these properties of natural numbers when reasoning about λ-terms in De Bruijn notation, where the typing context is non-empty. In such a situation, it is useful to automatically transport the existing theorems to the new kinds of contexts, since we know that the theorem in question cannot depend on the properties of λ-terms. While this example is rather trivial, it becomes non-trivial when theorems are proved about higher-order data structures, which are commonly encountered when reasoning about syntax with binding constructs.

One way to achieve such reuse automatically is a technique called *subordination*, which is based on analyzing the constructors for a certain type and defining syntactic criteria under which certain normal terms of one type can have subterms of another type. Unfortunately, the classical definition of subordination lacks a proof-theoretic justification, and has surprising properties in third-order (and higher) signatures.

In [36] (TLCA'15), we propose a proof-theoretic characterization of a kind of dual to subordination, called *independence*, that characterizes when normal terms of one type *cannot* contain subterms of another type. This is achieved by means of proving an inductive *strengthening* lemma about the signatures in the two-level logic approach. We also show how to automatically prove such lemmas in certain commonly encountered situations in the theorem prover Abella.

7.8. Disproving Non-Theorems with Saturating Search

Participants: Taus Brock-Nannestad, Kaustuv Chaudhuri.

High-performance automated reasoning techniques such as resolution and the inverse method are well suited for proving *true* conjectures, but are ill-behaved for *false* conjectures. For example, for a simple theory of even numbers that states that 0 is even and that $n + 2$ is even whenever n is even, it is obviously the case that the conjecture "3 is even" is unprovable, but the algorithm would loop forever proving "0 is even", "2 is even", "4 is even", etc. This behavior is observed even in the best saturation-based (i.e., forward-reasoning) theorem provers.

In [25] (TABLEAUX'15), we show how to finitely constrain the search space of saturation-based theorem provers by the use of *unsound* extensions of the goal query. These unsound extensions, when combined with forward subsumption, guarantee that only a finite number of consequences would ever be constructed based on any goal query, so the proof search procedure is guaranteed to terminate. If a proof is found among them that does not use the unsound extensions, then we can can succeed with that proof. If no proof is found, then we can soundly assert that the original goal query was also unprovable, since even a weakened version of it was unprovable. The only other possibility is that a proof is found using the unsound extension; in this case, we use the particular instance of unsoundness to refine the original unsound goal to prevent it from being found again, while maintaining the invariant that the search space is finite, and rerun the search. Since first-order logic is undecidable, we may need to repeat the refinement procedure indefinitely, but for many kinds of domains, particularly those arising from typed signatures (such as the even numbers example above), we do eventually find a saturating approximation that guarantees that the conjecture has no proof.

This algorithm has been implemented as part of the Mætning theorem prover explained in the section on Software above. We plan to extend it in the future with various automatic refinement heuristics.

7.9. Encoding Bigraph Structure with Subexponentials

Participants: Kaustuv Chaudhuri, Giselle Reis.

Bigraphs were proposed by Robin Milner as a model of ubiquitous computing, which is computation that is aware both of *location* and of *connections*. As a formalism it subsumes many other process calculi such as CCS and the π-calculus. However, it has a number of problems *qua* syntax because it is based on graphs and a complicated theory of composition. The biggest of these problems is how to implement it in a formal reasoning system.

In recent years, many members (and ex-members) of the Parsifal team have been experimenting with a variant of linear logic that has not just a single pair but an arbitrary family of exponential connectives that are arranged in a pre-order. Each such pair of *subexponentials* may admit or reject the structural properties of weakening and contraction. One benefit of subexponentials is that it allows for *querying* the *absence* of certain kinds of exponential formulas without requiring all non-exponential formulas to be deleted as a consequence, which is the issue with ordinary linear logic.

In [28] (LPAR'15), we show how to represent the structure of bigraphs in terms of a simple theory of linear logic with subexponentials (SEL). We show that our representation is adequate, i.e., that it respects the composition and juxtaposition operations on bigraphs. Moreover, we show how one can ask queries about the nesting of places in the representation without modifying it, which gives us a technical means of encoding bigraph reactions as well. Some of the details for bigraph reactions remain to be worked out in future work.

7.10. Encoding Additive Connectives with Multiplicatives and Subexponentials

Participant: Kaustuv Chaudhuri.

In a recent workshop on Linearity [55], we have published the formal proof (that was obtained in 2009) that linear logic with three subexponentials in a certain lattice is undecidable. An extended version of this paper was submitted to a special issue on Linearity in Mathematical Structures in Computer Science and was accepted in November 2015.

The preprint of that extended paper [41] gives a direct embedding of propositional MALL (multiplicative and additive linear logic) using only multiplicative connectives and five subexponentials. This means that the additive connectives are, in fact, redundant when we have multiplicatives and subexponentials. Moreover, in the first-order case this encoding is polynomial and focally adequate, which means that MALL can be simulated at the highest fidelity – at the level of individual inference rules.

7.11. Computation in Focused Intuitionistic Logic

Participants: Taus Brock-Nannestad, Nicolas Guenot, Daniel Gustafsson.

Focusing is a proof-theoretical tecnique for eliminating unnecessary nondeterminism in proofs. Because it cuts down on nondeterminism, focusing is particularly useful for directing proof search. Focusing thus plays a key role in explaining the meaning and behaviour of logic programs.

Despite this success in clarifying the operational semantics of logic programming, focusing has not been as widely studied in the Curry-Howard style "proofs as programs" interpretation. Early results in this area established that λ-calculi associated with the focused calculi LJT and LJQ had evaluation strategies corresponding to call-by-name and call-by-value respectively. For the LJF calculus — which contains both LJT and LJQ as fragments — no such correspondence was known.

In [27] (PPDP'15) we show how a proof-term assignment to (a variant of) Liang and Miller's focused sequent calculus LJF permits a uniform treatment of the call-by-value and call-by-name reduction strategies of the λ-calculus, as well as combinations of these strategies. Additionally, we show how to extract an abstract machine from LJF by considering machine states as certain configurations of instances of the cut rule. The aforementioned correspondence extends to this setting, and we show that well-known abstract machines for call-by-value and call-by-name are in fact exactly the abstract machines that one gets when considering certain fragments of LJF.

In the seminal work of Paul Blain Levy, the call-by-push-value language was introduced as a way of subsuming the call-by-value and call-by-name strategies of the λ-calculus. It was later on conjectured that call-by-push-value was simply implementing a notion of focusing, and indeed this turns out to be the case, as we show in the aforementioned paper.

7.12. Focused Linear Logic and the λ-calculus

Participants: Taus Brock-Nannestad, Nicolas Guenot.

Linear Logic enjoys strong symmetries inherited from classical logic while providing a constructive framework comparable to intuitionistic logic. However, the computational interpretation of sequent calculus presentations of linear logic remains problematic, mostly because of the many rule permutations allowed in the sequent calculus.

In focused variants of Linear Logic, most of these rule permutations are eliminated by the focusing restriction — during focusing, a single formula is decomposed eagerly, and the focus is passed down to its subformulas. Conversely, during inversion, all invertible connectives are decomposed. Moreover, this decomposition is made fully determinstic by keeping the connectives in question in a list, and only decomposing the first connective of this list.

The end result of this is that a focused proof in Linear Logic almost always has one particular formula singled out as the one that will be decomposed. Thus, somewhat curiously, focused Linear Logic behaves much more like an intuitionistic sequent calculus (where at all times there is a single "special" formula on the right hand side of the sequent) than a classical calculus.

In [26] (MFPS'15), we study a term assignment for a focused version of Multiplicative Exponential Linear Logic (MELL), and show how the focusing technique gives rise to a calculus that straightforwardly embeds both a linear variant of the λ-calculus, and a sequent-based formulation of Parigot's $\lambda\mu$-calculus.

7.13. There is no complete linear term rewriting system for propositional logic

Participant: Lutz Straßburger.

Recently, we observed that the set of all sound linear inference rules in propositional logic is already coNP-complete [84]. This means that every boolean tautology can be written as a (left-and right-) linear rewrite rule. This raises the question of whether there is a rewriting system on linear terms of propositional logic that is sound and complete for the set of all such rewrite rules. We have shown (in a joint work with Anupam Das) that, as long as reduction steps are polynomial-time decidable, such a rewriting system does not exist unless coNP=NP. This is published in [20].

7.14. A (Bi)linear Implementation of Strong Call-by-Value

Participant: Beniamino Accattoli.

The elegant theory of the call-by-value λ-calculus relies on closed terms and weak evaluation (i.e., not under abstractions) and it is well-known that the number of call-by-value β-steps is a reasonable cost model. When turning to open terms or strong evaluation—that are used for instance in the implementation of Coq—the operational theory breaks, and the call-by-value λ-calculus has to be extended with some additional rewriting rules. In a joint work with Sacerdoti Coen [18], a proposal for open/strong call-by-value, called *fireball calculus*, is studied from the point of view of cost models and abstract machines. First, it is shown that open terms introduce a new malicious behavior, making the study of cost models non-trivial. Second, it is shown that the number of β-steps in the fireball calculus is a reasonable cost model. Third, a new abstract machine is introduced and its overhead is shown to be linear with respect to the number of β-steps and the size of the initial term, providing a surprisingly efficient implementation scheme.

7.15. Implementations of Strong Call-by-Name, Revisited

Participant: Beniamino Accattoli.

The literature about abstract machines for the strong evaluation (i.e., possibly under abstraction) of the ordinary (i.e., call-by-name) λ-calculus is scarce. Essentially, there is a single, old work: Crégut's abstract machine [60] (1990), that is an extension of Krivine abstract machine to compute full normal forms. Crégut studies the correctness of the machine by means of an explicit substitutions calculus. In this joint work with Barenbaum and Mazza [17], Crégut's work is revisited and simplified in the extreme. An alternative, simpler machine is introduced, the *Strong Milner abastract machine*. Its correctness is studied via *linear substitution calculus*, a new approach to explicit substitutions developed by Accattoli and Kesner that is much simpler than Crégut's approach. Moreover, a complexity analysis of the machine is provided: its overhead is shown to be linear in the number of steps in the linear substitution calculus and in the size of the inital term.

7.16. Foundational Proof Certificates

We have continued to explore a number of new aspects of framework we call *Foundational Proof Certificates* (FPCs). Besides having defined and implemented prototype checkers for FPCs in classical and intuitionistic logic [37] we have also extended the proof theory underlying numerous modal logics so that FPCs can be applied to modal logics [35]. We have also extended the notion of FPC to work also in the model checking setting [31]. In both the modal logic and model checking domains, the key to getting FPCs to work is to have descriptions of *focused proof systems* available for those logics.

Given that FPCs are declarative and semantically simple structures, it has been possible to find numerous applications of them outside the problem of simply checking them. It was shown, for example, that FPCs can be used to help define the semantics of the output from traditional theorem provers [23]. We have also used FPCs as proof outlines in order to define high-level tactics to direct proof search [24].

7.17. Multi-level Delimited Control

There has been a great deal of interest in recent years to providing interesting functional programming primitives that are based on classical logic and not just intuitionistic logic. Unfortunately, the standard sequent calculus proof theory for classical logic is far too chaotic to provide such a foundation. We have recently proposed adding to classical (linear) logic an assortment of *subexponentials* and to provide a rigid structure for their placement within formulas. This new framework allows for sequent calculus proof theory to provide to the functional programming paradigm the feature often called *multi-level delimited control* [32]. The main result in that paper is also noteworthy in that it shows how to build certain complex synthetic connectives even though the standard approach (using focusing proof systems) cannot be used.

8. Partnerships and Cooperations

8.1. European Initiatives

8.1.1. FP7 & H2020 Projects

8.1.1.1. Proofcert

Title: ProofCert: Broad Spectrum Proof Certificates

Programm: FP7

Type: ERC

Duration: January 2012 - December 2016

Coordinator: Inria

Inria contact: Dale Miller

'There is little hope that the world will know secure software if we cannot make greater strides in the practice of formal methods: hardware and software devices with errors are routinely turned against their users. The ProofCert proposal aims at building a foundation that will allow a broad spectrum of formal methods—ranging from automatic model checkers to interactive theorem provers—to work together to establish formal properties of computer systems. This project starts with a wonderful gift to us from decades of work by logicians and proof theorist: their efforts on logic and proof has given us a universally accepted means of communicating proofs between people and computer systems. Logic can be used to state desirable security and correctness properties of software and hardware systems and proofs are uncontroversial evidence that statements are, in fact, true. The current state-of-the-art of formal methods used in academics and industry shows, however, that the notion of logic and proof is severely fractured: there is little or no communication between any two such systems. Thus any efforts on computer system correctness is needlessly repeated many time in the many different systems: sometimes this work is even redone when a given prover is upgraded.

In ProofCert, we will build on the bedrock of decades of research into logic and proof theory the notion of proof certificates. Such certificates will allow for a complete reshaping of the way that formal methods are employed. Given the infrastructure and tools envisioned in this proposal, the world of formal methods will become as dynamic and responsive as the world of computer viruses and hackers has become.'

8.2. International Research Visitors

8.2.1. Visits of International Scientists

Professor Chuck Liang visited the team from 25 May to 15 June 2015 in order to continue his collaborations with team members on basic questions of proof theory. This collaboration resulted in a paper that appears in LPAR 2015 on the topic of subexponentials and the Curry-Howard interpretation of logic.

8.2.1.1. Internships

Leonardo Lima is an intern funded by ProofCert during 1 Oct 2015 – 28 Feb 2016. He is a student of Prof. Vivek Nigam from Federal University of Paraíba, Brazil. He is working on formalizing the proof theory of linear logic within the Abella theorem prover.

8.2.2. Visits to International Teams

8.2.2.1. Research stays abroad

Graham-Lengrand spent 6 months, from March 2015 to August 2015 at SRI International, USA. This visit was to start a collaboration with N. Shankar and B. Dutertre on new algorithms and new architectures for automated and interactive theorem proving.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. Member of the organizing committees

L. Straßburger was on the Organization Committee for the International Workshop on Efficient and Natural Proof Systems, held at the University of Bath 14-16 December, 2015

9.1.2. Scientific events selection

9.1.2.1. Chair of conference program committees

K. Chaudhuri was the Program Committee chair of the following meeting.

LFMTP 2015: Logical Frameworks and Meta-Languages: Theory and Practice
Co-chaired with Iliano Cervesato, Carnegie Mellon University (Qatar)

9.1.2.2. Member of conference program committees

D. Miller was on the Program Committees of the following meetings.

TABLEAUX 2015: Automated Reasoning with Analytic Tableaux and Related Methods, Wroclaw, Poland, September.

PPDP 2015: 17th International Symposium on Principles and Practice of Declarative Programming, Siena, Italy, 13-16 July.

PxTP 2015: Fourth Workshop on Proof eXchange for Theorem Proving, Berlin, Germany, 2-3 August.

ICALP 2015 (42nd International Colloquium on Automata, Languages, and Programming), Track B, Kyoto, Japan, 6-10 July.

S. Graham-Lengrand was on the Program Committees of the following meetings.

TABLEAUX 2015: Automated Reasoning with Analytic Tableaux and Related Methods, Wroclaw, Poland, September.

CSL 2015: 24th EACSL Annual Conference on Computer Science Logic, Berlin, Germany, 7-10 September.

K. Chaudhuri was on the Program Committees of the following meetings.

CPP 2015: ACM-SIGPLAN Conference on Certified Programs and Proofs

FICS 2015: International Workshop on Fixed Points in Computer Science

WoF 2015: International Workshop on Focusing

L. Straßburger was on the Program Committees of the following meetings.

FoSSaCS 2015: 18th International Conference on Foundations of Software Science and Computation Structures

WENSP: International Workshop on Efficient and Natural Proof Systems

G. Reis was on the Program Committee for the following meeting.

LFMTP 2015: Logical Frameworks and Meta-Languages: Theory and Practice

9.1.2.3. Reviewer

T. Libal was a reviewer for the following workshops:

Fourth Workshop on Proof eXchange for Theorem Proving.

S. Graham-Lengrand was a reviewer for the following conferences:

LPAR 2015: 20th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Suva, Fiji, November.

LICS 2015: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, Kyoto, Japan, July.

D. Ilik was a reviewer for the following conferences:

ESOP 2015: 24th European Joint Conferences on Theory and Practice of Software

FoSSaCS 2015: 18th International Conference on Foundations of Software Science and Computation Structures

TLCA 2015: 13th International Conference on Typed Lambda Calculi and Applications

M. Volpe was a reviewer for the following conferences:

Tableaux 2015: 24th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, Wroclaw, Poland, September.

L. Straßburger was a reviewer for the following conferences:

TLCA 2015: 13th International Conference on Typed Lambda Calculi and Applications

LPAR 2015: 20th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Suva, Fiji, November.

FLOPS 2016: Thirteenth International Symposium on Functional and Logic Programming March 4-6, 2016, Kochi, Japan

B. Accattoli was a reviewer for the following conferences:

- CSL 2015: 24th EACSL Annual Conference on Computer Science Logic, Berlin, Germany, 7-10 September.
- TGC 2015: 10th International Symposium on Trustworthy Global Computing, Madrid, Spain, August 31 to September 1.
- FoSSaCS 2016: 19th International Conference on Foundations of Software Science and Computation Structures, to be held in Eindhoven, The Netherlands, 2-8 April 2016.

T. Brock-Nannestad was a reviewer for the following conferences:

- ITP 2015: 6th Conference on Interactive Theorem Proving, Nanjing, China, August 24–27 August, 2015.
- CPP 2016: The 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, Florida, USA, January 18–19, 2016.

9.1.3. Journal

9.1.3.1. Editor-in-chief

D. Miller was Editor-in-Chief of the ACM Transactions on Computational Logic. He completed a six year term on June 2015.

9.1.3.2. Member of the editorial boards

D. Miller is on the editorial board of the following journals: *ACM Transactions on Computational Logic*, *Journal of Automated Reasoning* (Springer), *Theory and Practice of Logic Programming* (Cambridge University Press), and *Journal of Applied Logic* (Elsevier).

D. Miller served on as an editor for Logical Methods in Computer Science for the production of a special issue for papers from CSL-LICS 2014.

K. Chaudhuri served as a guest editor for Mathematical Structures in Computer Science special issue on Logical Frameworks and Meta Languages, based on selected papers from LFMTP 2015

9.1.3.3. Reviewer - Reviewing activities

- T. Libal was a reviewer for the following journals:

  Journal of Automated Reasoning (Springer).
- S. Graham-Lengrand was a reviewer for the following journals: J. Appl. Logic, Theoret. Comput. Sci., Acta Inform.
- D. Ilik was a reviewer for the following journals:

  Mathematical Reviews (MathSciNet)

  Zentralblatt MATH
- M. Volpe was a reviewer for the following journals:

  Annals of Mathematics and Artificial Intelligence;

  Journal of Automated Reasoning;

  Logica Universalis.
- L. Straßburger was a reviewer for the following journals:

  Logical Methods in Computer Science (two articles)

  Theoretical Computer Science, Elsevier

  Ontos Mathematical Logic, Walter De Gruyter

  Logic Journal of the IGLP, Oxford University Press
- B. Accattoli was a reviewer for the following journals:
  - Logical Methods in Computer Science (two articles)
  - Theoretical Computer Science, Elsevier

9.1.4. Invited talks

D. Miller was a plenary speaker at the joint meeting of LOPSTR 2015 and PPDP 2015 (Siena, Italy) and was an invited speaker at LSFA 2015 (Natal Brazil) and in the Session on History and Philosophy of Computing at the 15th Congress of Logic, Methodology and Philosophy of Science, Helsinki.

D. Miller gave spoke at the ETH Zurick Department of Computer Science Distinguished Colloquium Series on 20 April.

S. Graham-Lengrand gave an invited talk at the IFIP Working Group 1.6 on Term Rewriting, on the occasion of its 2015 annual meeting in Warsaw, Poland.

B. Accattoli was an invited speaker at the 16th International Workshop on Logic and Computational Complexity (Kyoto, Japan, 5th of July).

9.1.5. Leadership within the scientific community

D. Miller was a member of the ACM SIGLOG Advisory Board, the LICS Organizing Board, the CPP Steering Committee, and the ACM SIGLOG Executive Committee Nominating Committee. He was also the Chair of the LICS 1995 Test-of-Time Award selection committee.

S. Graham-Lengrand is head of the French National Workgroup on *Logic, Algebra, and Computation* (part of GDR Informatique Mathématique, funded by CNRS).

K. Chaudhuri was elected to the Steering Committee of LFMTP (http://lfmtp.org) for a term of 5 years in 2015.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Licence: K. Chaudhuri, "*INF 321 : Les principes des langages de programmation*", 40 hours eq. TD, L3, École polytechnique, France

Licence: K. Chaudhuri, "*INF 431 : Programmation concurrente et distribuée*", 40 hours eq. TD, L3, École polytechnique, France

Licence: S. Graham-Lengrand, "*INF411: Les bases de la programmation et de l'algorithmique*", 32 hours eq. TD, L3, École Polytechnique, France.

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

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

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

Licence: T. Libal, "*INF 321 : Les principes des langages de programmation*", 40 hours eq. TD, L3, École polytechnique, France

Master: T. Libal, "*INF551: Computer-aided reasoning*", 20 hours eq. TD, M1, Êcole Polytechnique, France.

9.2.2. Supervision

PhD: Zakaria Chihani, "*Certification of first-order proofs in classical and intuitionistic logics*", École Polytechnique, 2 Nov 2015; supervised by Dale Miller.

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

PhD in progress: Roberto Blanco, Ulysse Gérard, and Quentin Heath, supervised by D. Miller

Internship: Horace Blanc, May-September 2015, supervised by B. Accattoli.

9.2.3. Juries

D. Miller was a reporter and president for the PhD jury of Ali Assaf, École Polytechnique, 28 September.

S. Graham-Lengrand was a member of the PhD thesis committee of Simon Cruanes, École Polytechnique, 10 September.

K. Chaudhuri was a member of the PhD thesis committee of Yuting Wang, University of Minnesota, 5 November.

9.3. Popularization

K. Chaudhuri organized a half-day tutorial on Abella as a satellite event of CADE 2015 in Berlin. The slides from the talk are available as [40], and the tutorial web-site is: http://abella-prover.org/tutorial. This event was co-organized with Gopalan Nadathur, professor of Computer Science at the University of Minnesota, USA.

G. Reis participated in “Fête de la Science” as an assistant to the programming activities.

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

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

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

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

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

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

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

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

[10] L. STRASSBURGER. *On the Axiomatisation of Boolean Categories with and without Medial*, in "Theory and Applications of Categories", 2007, vol. 18, n° 18, pp. 536–601, http://arxiv.org/abs/cs.LO/0512086

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

Publications of the year

Articles in International Peer-Reviewed Journals

[12] B. ACCATTOLI. *Proof nets and the call-by-value λ-calculus*, in "Journal of Theoretical Computer Science (TCS)", 2015 [*DOI :* 10.1016/J.TCS.2015.08.006], https://hal.archives-ouvertes.fr/hal-01244842

[13] R. ARISAKA, A. DAS, L. STRASSBURGER. *On Nested Sequents for Constructive Modal Logics*, in "Logical Methods in Computer Science", September 2015, vol. 11, n° 3, pp. 1-33 [*DOI :* 10.2168/LMCS-11(3:7)2015], https://hal.inria.fr/hal-01093143

[14] G. EDAN, M. FREEDMAN, X. MONTALBAN, D. MILLER, H. HARTUNG, B. HEMMER, E. FOX, F. BARKHOF, S. SCHIPPLING, A. SCHULZE, D. PLEIMES, C. POHL, R. SANDBRINK, G. SUAREZ, E.-M. WICKLEIN, L. KAPPOS. *Long-term Impact of Early MS Treatment with Interferon Beta-1b (IFNB-1b): Clinical, MRI, Employment, and Patient-Reported Outcomes (PROs) at the 11-Year Follow-up of BENEFIT (BENEFIT 11) (P7.012)*, in "Neurology", June 2015, vol. 84, n° 14 Supplement, P7.012 p. , https://hal-univ-rennes1.archives-ouvertes.fr/hal-01259391

[15] L. MAJUMDAR, P. GORAI, A. DAS, S. K. CHAKRABARTI. *Potential formation of three pyrimidine bases in interstellar regions*, in "Astrophysics and Space Science", December 2015, vol. 360, 14 p. [*DOI :* 10.1007/S10509-015-2567-1], https://hal.archives-ouvertes.fr/hal-01239142

[16] L. STRASSBURGER, N. NOVAKOVIC. *On the Power of Substitution in the Calculus of Structures*, in "ACM Transactions on Computational Logic", April 2015, vol. 16, n° 3 [*DOI :* 10.1145/2701424], https://hal.archives-ouvertes.fr/hal-00925707

International Conferences with Proceedings

[17] B. ACCATTOLI, P. BARENBAUM, D. MAZZA. *A Strong Distillery*, in "Programming Languages and Systems - 13th Asian Symposium, APLAS 2015", Pohang, South Korea, Lecture notes in computer science, November 2015, vol. 9458, pp. 231-250 [*DOI :* 10.1007/978-3-319-26529-2_13], https://hal.archives-ouvertes.fr/hal-01244838

[18] B. ACCATTOLI, C. SACERDOTI COEN. *On the Relative Usefulness of Fireballs*, in "30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015", Kyoto, Japan, July 2015 [*DOI :* 10.1109/LICS.2015.23], https://hal.archives-ouvertes.fr/hal-01244833

[19] K. CHAUDHURI, M. CIMINI, D. MILLER. *A Lightweight Formalization of the Metatheory of Bisimulation-Up-To*, in "4th ACM-SIGPLAN Conference on Certified Programs and Proofs (CPP)", Mumbai, India, X. LEROY, A. TIU (editors), ACM Proceedings, January 2015 [*DOI :* 10.1145/2676724.2693170], https://hal.inria.fr/hal-01091524

[20] A. DAS, L. STRASSBURGER. *No complete linear term rewriting system for propositional logic*, in "26th International Conference on Rewriting Techniques and Applications (RTA 2015)", Warsaw, Poland, June 2015 [*DOI :* 10.4230/LIPICS.RTA.2015.127], https://hal.inria.fr/hal-01236948

[21] S. GRAHAM-LENGRAND. *Realisability semantics of abstract focussing, formalised*, in "Proceedings of the First International Workshop on Focusing", Suva, Fiji, November 2015, vol. 197, 14 p. [*DOI :* 10.4204/EPTCS.197.3], https://hal.archives-ouvertes.fr/hal-01242866

[22] D. ROUHLING, M. FAROOQUE, S. GRAHAM-LENGRAND, J.-M. NOTIN, A. MAHBOUBI. *Axiomatic constraint systems for proof search modulo theories*, in "10th International Symposium on Frontiers of Combining Systems (FroCoS'15)", Wroclaw, Poland, C. LUTZ, S. RANISE (editors), LNAI, Springer, September 2015, vol. 9322 [*DOI :* 10.1007/978-3-319-24246-0_14], https://hal.inria.fr/hal-01107944

Conferences without Proceedings

[23] R. BLANCO, T. LIBAL, D. MILLER. *Defining the meaning of TPTP formatted proofs*, in "11th International Workshop on the Implementation of Logics", Suva, Fiji, November 2015, https://hal.inria.fr/hal-01238434

[24] R. BLANCO, D. MILLER. *Proof Outlines as Proof Certificates: A System Description*, in "First International Workshop on Focusing", Suva, Fiji, November 2015, https://hal.inria.fr/hal-01238436

[25] T. BROCK-NANNESTAD, K. CHAUDHURI. *Disproving Using the Inverse Method by Iterative Refinement of Finite Approximations*, in "Automated Reasoning with Analytic Tableaux and Related Methods", Wrocław, Poland, September 2015 [*DOI :* 10.1007/978-3-319-24312-2_11], https://hal.inria.fr/hal-01222592

[26] T. BROCK-NANNESTAD, N. GUENOT. *Focused Linear Logic and the λ-calculus*, in "Mathematical Foundations of Programming Semantics XXXI", Nijmegen, Netherlands, June 2015 [*DOI :* 10.1016/J.ENTCS.2015.12.008], https://hal.inria.fr/hal-01249220

[27] T. BROCK-NANNESTAD, N. GUENOT, D. GUSTAFSSON. *Computation in Focused Intuitionistic Logic*, in "17th International Symposium on Principles and Practice of Declarative Programming", Siena, Italy, July 2015 [*DOI :* 10.1145/2790449.2790528], https://hal.inria.fr/hal-01249216

[28] K. CHAUDHURI, G. REIS. *An adequate compositional encoding of bigraph structure in linear logic with subexponentials*, in "20th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)", Suva, Fiji, M. DAVIS, A. FEHNKER, A. MCIVER, A. VORONKOV (editors), Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, November 2015, vol. 9450, pp. 146–161 [*DOI :* 10.1007/978-3-662-48899-7_11], https://hal.inria.fr/hal-01208362

[29] Z. CHIHANI, T. LIBAL, G. REIS. *The Proof Certifier Checkers*, in "Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)", Wroclaw, Poland, September 2015, vol. 9323, pp. 201-210 [*DOI :* 10.1007/978-3-319-24312-2_14], https://hal.inria.fr/hal-01208333

[30] 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

[31] Q. HEATH, D. MILLER. *A framework for proof certificates in finite state exploration*, in "Proceedings of the Fourth Workshop on Proof eXchange for Theorem Proving", Berlin, Germany, August 2015 [*DOI :* 10.4204/EPTCS.186.4], https://hal.inria.fr/hal-01240172

[32] C. LIANG, D. MILLER. *On Subexponentials, Synthetic Connectives, and Multi-level Delimited Control*, in "LPAR 20 - International Conference on Logic for Programming, Artificial Intelligence and Reasoning", Suva, Fiji, M. DAVIS, A. FEHNKER, A. MCIVER, A. VORONKOV (editors), LNCS - Lecture Notes in Computer Science, Springer, November 2015, vol. 9450, pp. 297-312 [*DOI :* 10.1007/978-3-662-48899-7_21], https://hal.inria.fr/hal-01239753

[33] T. LIBAL. *Regular Patterns in Second-Order Unification*, in "Proceedings of CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany", Berlin, Germany, August 2015 [*DOI :* 10.1007/978-3-319-21401-6_38], https://hal.archives-ouvertes.fr/hal-01242215

[34] T. LIBAL. *Towards Deciding Second-order Unification Problems Using Regular Tree Automata*, in "29th International Workshop on Unification", Warsaw, Poland, June 2015, https://hal.inria.fr/hal-01242233

[35] D. MILLER, M. VOLPE. *Focused labeled proof systems for modal logic*, in "20th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)", Suva, Fiji, November 2015, https://hal.inria.fr/hal-01213858

[36] Y. WANG, K. CHAUDHURI. *A Proof-theoretic Characterization of Independence in Type Theory*, in "13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015)", Warsaw, Poland, T. ALTENKIRCH (editor), Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, July 2015, vol. 38, pp. 332–346 [*DOI :* 10.4230/LIPICS.TLCA.2015.332], https://hal.inria.fr/hal-01222743

Scientific Books (or Scientific Book chapters)

[37] D. MILLER. *Foundational Proof Certificates*, in "All about Proofs, Proofs for All", D. DELAHAYE, B. W. PALEO (editors), College Publications, January 2015, vol. Mathematical Logic and Foundations, n^{o} 55, pp. 150-163, https://hal.inria.fr/hal-01239733

Books or Proceedings Editing

[38] I. CERVESATO, K. CHAUDHURI (editors). *Proceedings Tenth International Workshop on Logical Frameworks and Meta Languages: Theory and Practice*, July 2015 [*DOI :* 10.4204/EPTCS.185], https://hal.inria.fr/hal-01222747

Research Reports

[39] K. CHAUDHURI, S. MARIN, L. STRASSBURGER. *Focused and Synthetic Nested Sequents (Extended Technical Report)*, Inria, April 2016, https://hal.inria.fr/hal-01251722

Scientific Popularization

[40] K. CHAUDHURI, G. NADATHUR. *Reasoning about Computational Systems using Abella*, August 2015, Abella Tutorial, https://hal.inria.fr/hal-01222774

Other Publications

[41] K. CHAUDHURI. *Expressing Additives Using Multiplicatives and Subexponentials*, August 2015, working paper or preprint, https://hal.inria.fr/hal-01222767

[42] D. ILIK. *On the exp-log normal form of types*, June 2015, working paper or preprint, https://hal.inria.fr/hal-01167162

[43] P.-A. MELLIÈS, N. ZEILBERGER. *A bifibrational reconstruction of Lawvere's presheaf hyperdoctrine*, January 2016, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01261955

[44] M. ROCCHETTO, L. VIGANÒ, M. VOLPE. *An interpolation-based method for the verification of security protocols*, December 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01245442

[45] L. VIGANÒ, M. VOLPE, M. ZORZI. *A Branching Distributed Temporal Logic for Reasoning about Quantum State Transformations*, October 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01213511

[46] N. ZEILBERGER. *Linear lambda terms as invariants of rooted trivalent maps*, December 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01247757

References in notes

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

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

[49] 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

[50] 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

[51] 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

[52] N. BJORNER, M. JANOTA. *Playing with Quantified Satisfaction*, in "Proc. of the 20th Intern. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'15)", M. DAVIS, A. FEHNKER, A. MCIVER, A. VORONKOV (editors), LNCS, Springer, 2015, vol. 9450 [*DOI :* 10.1007/978-3-662-48899-7]

[53] 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

[54] 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

[55] K. CHAUDHURI. *Undecidability of Multiplicative Subexponential Logic*, in "3rd International Workshop on Linearity", Vienna, Austria, S. ALVES, I. CERVESATO (editors), July 2014, https://hal.inria.fr/hal-00998753

[56] 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

[57] 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

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

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

[60] P. CRÉGUT. *An Abstract Machine for Lambda-Terms Normalization*, in "LISP and Functional Programming", 1990, pp. 333–340, http://doi.acm.org/10.1145/91556.91681

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

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

[63] 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

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

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

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

[67] 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

[68] 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

[69] 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 [*DOI :* 10.1016/J.TCS.2009.07.041], http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tcs09.pdf

[70] 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 [*DOI :* 10.1016/J.APAL.2011.01.012]

[71] 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

[72] 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

[73] 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

[74] S. MCLAUGHLIN, F. PFENNING. *Imogen: Focusing the Polarized Focused Inverse Method for Intuitionistic Propositional Logic*, in "15th International Conference on Logic, Programming, Artificial Intelligence and Reasoning (LPAR)", I. CERVESATO, H. VEITH, A. VORONKOV (editors), LNCS, November 2008, vol. 5330, pp. 174–181, http://www.cs.cmu.edu/~seanmcl/papers/McLaughlin-LPAR-2008.pdf

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

[76] D. MILLER, G. NADATHUR. *Programming with Higher-Order Logic*, Cambridge University Press, June 2012 [*DOI :* 10.1017/CBO9781139021326]

[77] 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

[78] 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

[79] 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

[80] G. MUNCH-MACCAGNONI. *Focalisation and Classical Realisability*, in "Proc. of the 18th Annual Conf. of the European Association for Computer Science Logic (CSL'09)", E. GRÄDEL, R. KAHLE (editors), LNCS, Springer, 2009, vol. 5771, pp. 409-423 [*DOI :* 10.1007/978-3-642-04027-6_30]

[81] 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

[82] 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

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

[84] L. STRASSBURGER. *Extension without Cut*, in "Annals of Pure and Applied Logic", 2012, vol. 163, n^{o} 12, pp. 1995–2007 [*DOI :* 10.1016/J.APAL.2012.07.004], http://hal.inria.fr/hal-00759215

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

[86] 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

[87] N. ZEILBERGER. *The Logical Basis of Evaluation Order and Pattern-Matching*, Carnegie Mellon University, April 2009

[88] 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