IN PARTNERSHIP WITH:
CNRS

Ecole Polytechnique

Activity Report 2012

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
Programs, Verification and Proofs

Table of contents

Project-Team PARSIFAL

Keywords: Proof Theory, Automated Theorem Proving, Programming Languages, Logics

Creation of the Project-Team: July 01, 2007 .

1. Members

Research Scientists
Dale Miller [Team Leader, Senior Researcher, Inria Saclay – Île-de-France]
Kaustuv Chaudhuri [Junior Researcher, Inria Saclay – Île-de-France]
Stéphane Graham-Lengrand [Junior Researcher, CNRS]
François Lamarche [Senior Researcher, Inria Saclay – Île-de-France, Started January 1]
Lutz Straßburger [Junior Researcher, Inria Saclay – Île-de-France, HdR]

Engineer
Quentin Heath [ADT BATT, Started 1 September 2011]

PhD Students
Zakaria Chihani [Contrat Doctoral Inria, Started Fall 2012]
Mahfuza Farooque [ANR Jeunes Chercheurs PSI, Started 1 October 2010]
Ivan Gazeau [ANR Blanc CPP, Started 1 October 2009]
Nicolas Guenot [Allocataire Ministère de la Recherche, Started 1 October 2008]
Hernán-Pablo Vanzetto [Contrat doctoral Inria via Inria-MSR, Started 1 November 2010]

Post-Doctoral Fellows
Beniamino Accattoli [One Year, Ended August 31]
Matteo Cimini [One Year, Started October 1]
Willem Heijltjes [One Year, Ended October 31]
Stefan Hetzl [Fourteen Months, Started July 1]
Novak Novakovic [One Year, Started November 26]
Revantha Ramanayake [One Year, Ended August 31]
Fabien Renaud [One Year, Started September 3]

Administrative Assistant
Christelle Liévin

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 and verification 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.

2.2. Highlights of the Year

- Stefan Hetzl received his Habilitation 5 November 2012 from the Technical University of Vienna.
- Kaustuv Chaudhuri and Stefan Hetzl organized "Collegium Logicum 2012: Structural Proof Theory" at Inria-Saclay.
- Dale Miller and Gopalan Nadathur (Professor at the University of Minnesota) published a book title "Programming with higher-order logic" (June 2012, Cambridge University Press).

3. Scientific Foundations

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 [44], 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 [49], and different proof search strategies can be used to justify the design of new and different logic programming languages [48].

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 [56] 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 [53] 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 [54], 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. Design of two level-logic systems

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 has been published is a series of papers by McDowell and Miller [46] [45], Tiu and Miller [51] [57], and Gacek, Miller, and Nadathur [2] [38]. Besides presenting formal properties of these logic, these papers also documented a number of examples where this logic demonstrated superior approaches to reasoning about a number of complex formal systems, ranging from programming languages to the λ-calculus and π-calculus.

The team has also been working on three different prototype theorem proving system that are all related to this stronger logic. These 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.
- Tac, which is a sophisticate tactic for automatically completing simple proofs involving induction and unfolding.

We are now in the process of attempting to make all of these system communicate properly. Given that these systems have been authored by different team members at different times and for different reasons, they do not formally share the same notions of syntax and proof. We are now working to revisit all of these systems and revise them so that they all work on the *same* logic and so that they can share their proofs with each other.

Currently, Chaudhuri, Miller, and Accattoli are working with our technical staff member, Heath, to redesign and restructure these systems so that they can cooperate in building proofs.

3.3. Making the case for proof certificates

The team has been considering how it might be possible to define a universal format for proofs so that any existing theorem provers can have its proofs trusted by any other prover. This is a rather ambitious project and involves a great deal of work at the infrastructure level of computational logic. As a result, we have put significant energies into considering the high-level objectives and consequences of deploying such proof certificates.

Our current thinking on this point is roughly the following. Proofs, both formal and informal, are documents that are intended to circulate within societies of humans and machines distributed across time and space in order to provide trust. Such trust might lead a mathematician to accept a certain statement as true or it might help convince a consumer that a certain software system is secure. Using this general definition of proof, we have re-examined a range of perspectives about proofs and their roles within mathematics and computer science that often appears contradictory.

Given this view of proofs as both document and object, that need to be communicated and checked, we have attempted to define a particular approach to a *broad spectrum proof certificate* format that is intended as a universal language for communicating formal proofs among computational logic systems. We identify four desiderata for such proof certificates: they must be

1. checkable by simple proof checkers,
2. flexible enough that existing provers can conveniently produce such certificates from their internal evidence of proof,
3. directly related to proof formalisms used within the structural proof theory literature, and
4. permit certificates to elide some proof information with the expectation that a proof checker can reconstruct the missing information using bounded and structured proof search.

We consider various consequences of these desiderata, including how they can mix computation and deduction and what they mean for the establishment of marketplaces and libraries of proofs. More specifics can be found in Miller's papers [8] and [47].

3.4. Combining Classical and Intuitionistic Proof Systems

In order to develop an approach to proof certificates that is as comprehensive as possible, one needs to handle theorems and proofs in both classical logic and intuitionistic logic. Yet, building two separate libraries, one for each logic, can be inconvenient and error-prone. An ideal approach would be to design a single proof system in which both classical and intuitionistic proofs can exist together. Such a proof system should allow cut-elimination to take place and should have a sensible semantic framework.

Liang and Miller have recently been working on exactly that problem. In their paper [7], they showed how to describe a general setting for specifying proofs in intuitionistic and classical logic and to achieve one framework for describing initial-elimination and cut-elimination for such these two logics. That framework allowed for some mixing of classical and intuitionistic features in one logic. A more ambitious merging of these logics was provided in their work on "polarized intuitionistic logic" in which classical and intuitionistic connectives can be used within the same formulas [14].

3.5. Deep inference

Deep inference [40], [42] 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.

3.6. 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 [39] for linear logic but also in Robinson’s proof nets [55] 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., [4], [3] and [41]. 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 [4] 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.

4. Application Domains

4.1. Automated Theorem Proving

Automated theorem proving has traditionally focused on classical first-order logic, but non-classical logics are increasingly becoming important in the specification and analysis of software. Most type systems are based on (possibly second-order) propositional intuitionistic logic, for example, while resource-sensitive and concurrent systems are most naturally expressed in linear logic.

The members of the Parsifal team have a strong expertise in the design and implementation of performant automated reasoning systems for such non-classical logics. In particular, the Linprover suite of provers [35] continue to be the fastest automated theorem provers for propositional and first-order linear logic.

Any non-trivial specification, of course, will involve theorems that are simply too complicated to prove automatically. It is therefore important to design semi-automated systems that allow the user to give high level guidance, while at the same time not having to write every detail of the formal proofs. High level proof languages in fact serve a dual function – they are more readily comprehended by human readers, and they tend to be more robust with respect to maintenance and continued evolution of the systems. Members of the Parsifal team, in association with other Inria teams and Microsoft Research, have been building a heterogeneous semi-automatic proof system for verifying distributed algorithms [36].

On a more foundational level, the team has been developing many new insights into the structure of proofs and the proof search spaces. Two directions, in particular, present tantalizing possibilities:

- The concept of *multi-focusing* [37] can be used to expose concurrency in computational behavior, which can in turn be exploited to prune areas of the proof search space that explore irrelevant interleavings of concurrent actions.
- The use of *bounded search*, where the bounds can be shown to be complete by meta-theoretic analysis, can be used to circumvent much of the non-determinism inherent in resource-sensitive logics such as linear logic. The lack of proofs of a certain bound can then be used to justify the presence or absence of properties of the encoded computations.

Much of the theoretical work on automated reasoning has been motivated by examples and implementations, and the Parsifal team intends to continue to devote significant effort in these directions.

4.2. Mechanized Metatheory

There has been increasing interest in the use of formal methods to provide proofs of properties of programs and programming languages. Tony Hoare's Grand Challenge titled "Verified Software: Theories, Tools, Experiments" has as a goal the construction of "verifying compilers" for a world where programs would only be produced with machine-verified guarantees of adherence to specified behavior. Guarantees could be given in a number of ways: proof certificates being one possibility.

The POPLMark challenge [33] envisions "a world in which mechanically verified software is commonplace: a world in which theorem proving technology is used routinely by both software developers and programming language researchers alike." The proposers of this challenge go on to say that a "crucial step towards achieving these goals is mechanized reasoning about language metatheory."

The Parsifal team has developed several tools and techniques for reasoning about the meta-theory of programming languages. One of the most important requirements for programming languages is the ability to reason about data structures with binding constructs up to α-equivalence. The use of higher-order syntax and nominal techniques for such data structures was pioneered by Miller, Nadathur and Tiu. The Abella system (see Section 3.2) implements a refinement of a number of these ideas and has been used to give full solutions to sections of the POPLMark challenge in addition to fully formal proofs of a number of other theorems in the meta-theory of the λ-calculus.

Now that the Abella system has been in circulation among colleagues during the past couple of years, there are many aspects of the methodology that now need to be addressed. During the summer of 2011, the team employed three interns Carnegie Mellon University and McGill University to work on different aspects of Abella. Particular focus was given to better ways to manipulate specification-logic contexts in the reasoning-logic and with finding ways to have Abella output a proper proof object (different from the scripts that are used to find a proof).

Our colleague Alwen Tiu from the Australian National University has also been building on our Bedwyr model checking tool so that we can build on top of it his SPEC system for doing model checking of spi-calculus expressions. We have adopted his enhancements to Bedwyr and are developing further improvements within the context of the BATT project (see Section 5.2).

4.3. Proof Certificates

Members of the Parsifal team have shown how to specify a large variety of proof systems—including natural deduction, the sequent calculus, and various tableau and free deduction systems—uniformly using either focused linear logic [52], [50] or focused intuitionistic logic [43] as the meta-language. In the presence of induction and co-induction, arbitrary finite computations can be embedded into single synthetic steps [34]. Additional work [8] shows that this same framework can also capture resolution refutations as well as Pratt primality certificates.

An important application then of this work in designing synthetic inference systems based on classical and intuitionistic logic is that of designing a *broad spectrum proof certificate*. The definition of proof certificates can be remarkably flexible within the simple setting of focused proofs.

The most important implications of such a certificate format would be that most of the worlds theorem provers should be able to print out their proofs and communication them to other provers: these other provers could then check such certificates by expanding the synthetic connectives they contain down into a small and fixed set of "micro" inference rules.

4.4. Automated reasoning and SMT solving

Automated reasoning uses a broad range of techniques whose soundness and completeness relate to the existence of proofs. The research programme of the ANR PSI project at Parsifal is to build a finer-grained connection by specifying automated reasoning techniques as the step-by-step construction of proofs, as we know it from proof theory and logic programming. The goal is to do this in a unifying framework, namely proof-search in a polarized and focussed logic. One of the advantages of this is to combine those techniques more easily. Another one is to envisage extending those techniques.

For instance, SAT-modulo-Theory problems require the combination of logical reasoning with domain-specific decision procedures. So in the PSI project we study how to incorporate the call to decision procedures in proof-theoretical framework like the focussed sequent calculus, and the proof-search mechanisms that are related to it.

In the same spirit we also study how to handle existential variables and equality, for which specific automated reasoning techniques have been designed (superposition / paramodulation calculi).

5. Software

5.1. Abella

Participants: Kaustuv Chaudhuri [correspondant], Matteo Cimini, Dale Miller.

Abella is an interactive theorem prover based on the two-level logic logic approach. It consists of a sophisticated reasoning logic that supports induction, co-induction, and generic reasoning, and a specification logic that is based on logic programming. Abella was initially designed to reason about simple second-order Lambda Prolog programs, which is sufficient for the computational specifications.

During 2012, as part of the RAPT Associated Team, Chaudhuri and Yuting Wang (intern from Univ. Minnesota) have been working on extending the expressive power of both levels of the Abella system. The following modifications have been made.

- We have extended the specification logic to support the full Lambda Prolog, which can be used to provide succinct higher-order specifications that tend to be unnatural and difficult to reason about with only second-order Lambda Prolog programs.
- We have extended the type system of Abella from simple types to parametrically polymorphic types. This is a significant improvement in the user-friendliness of the system as a lot of code does not have to be manually monomorphised and duplicated any more.
- We have experimented with extending the type system of Abella even further to higher-order predicate quantification. The theoretical basis of this work is a part of ongoing research, although we already have a number of examples of practical benefits of this extension.
- Finally, several improvements have been made to Abella's proof language to make the proofs more robust and reusable. We intend to make a more drastic change to the proof language in the future that will make proofs more declarative and high level.

The core development of Abella has also been centralized, with a single canonical repository and a new webpage: http://abella-prover.org. These resources are managed jointly by members of Parsifal and our colleagues at the University of Minnesota.

The next version of Abella, version 2.0, is in beta testing with expected release early in 2013.

5.2. Bedwyr

Participants: Quentin Heath, Dale Miller [correspondant].

During 2012, Quentin Heath has made the following important improvements to the Bedwyr model checking system.

- The concrete syntaxs for Bedwyr and Abella have been unified. Now, both systems can load the definitions and theorems developed in the other system. Eventually, we expect to have our model checker (Bedwyr) and interactive theorem prover (Abella) share theories and proofs.
- The documentation, distribution, and testing of Bedwyr were all improved, greatly increasing the usability of this system.
- The underlying support for logic has also been increased. In particular, the Bedwyr system contains a *tabling* mechanism which is capable of remembering past successful proofs (it can even support a finite failure as a successful proof of a negation). The most recent version of Bedwyr allows one to actually program the table in rather sophisticated ways. For example, simple lemmas can be loaded into the table and these lemmas can be used to greatly extend the range of what is tabled (remembered). We are currently examining different trade-offs between different styles of reasoning in the table (backchaining vs forwardchaining).

The work of Heath is being done in the context of the BATT ADJ project funded by Inria.

See also the web page http://slimmer.gforge.inria.fr/bedwyr/.

5.3. Psyche

Participants: Mahfuza Farooque, Stéphane Graham-Lengrand [correspondant].

Psyche (*Proof-Search factorY for Collaborative HEuristics*) is a modular programme for universal proof-search in classical logic. The motivation is twofold:

On the one hand, prove some mathematics of the broadest range while making the most of problem-specific techniques; On the other hand, gain high confidence about the correctness of the proofs produced without having to rely on a proof-checker.

The architecture is that of an interaction between a trusted universal kernel and smart plugins that are meant be efficient at solving certain kinds of problems:

The kernel contains the mechanisms for exploring the proof-search space in a sound and complete way, taking into account branching and backtracking. The output of Psyche comes from the (trusted) kernel and is therefore correct by construction. The plugins then drive the kernel by specifying how the branches of the search space should be explored, depending on the kind of problem that is being treated. The quality of the plugin is then measured by how fast it drives the kernel towards the final answer.

Version 1.0 of Psyche (released 4/9/2012) handles classical propositional logic, and its proof-search mechanism is simply the incremental construction of proof-trees in the polarised and focussed sequent calculus. The mechanism is driven by a plugin that emulates the behaviour of a SAT-solver (DPLL), with non-trivial features such as the eager application of the Unit Propagation rule, conflict analysis, backjumping and clause learning.

Psyche's input for that kind of SAT-problem is a file given in the standard DIMACS format.

See also the web page http://www.lix.polytechnique.fr/~lengrand/PSI/index.php?page=Psyche/index.

6. New Results

6.1. Recovering Proof Structures in the Sequent Calculus

Participants: Kaustuv Chaudhuri, Stefan Hetzl, Dale Miller.

The *sequent calculus* is often criticized as a proof syntax because it contains a lot of noise. It records the precise minute sequence of operations that was used to construct a proof, even when the order of some proof steps in the sequence is irrelevant and when some of the steps are unnecessary or involve detours. These features lead to several technical problems: for example, cut-elimination in the classical sequent calculus LK, as originally developed by Gentzen, is not confluent, and hence proof composition in LK is not associative. Many people choose to discard the sequent calculus when attempting to design a better proof syntax with the desired properties.

In recent years, there has been a project at Parsifal to recover some of these alternative proof syntaxes by imposing a certain abstraction over sequent proofs. The earliest example of this was in [37], where we showed a class of sequent proofs that were isomorphic to proof nets for multiplicative linear logic. In 2012, we were able to obtain a similar result for first-order classical logic, wherein we defined a class of sequent proofs that are isomorphic to expansion trees, a generalization of Herbrand disjunctions that is in some sense a minimalistic notion of proof for classical logic. This result was published at the CSL 2012 conference [22] and a journal version is in preparation.

Our technique for recovering these dramatically different proof structures directly in the sequent calculus involves the use of *maximal multi-focusing* which gives a syntactic characterization of those sequent proofs that: (1) have a “don’t care” ordering of proof steps where the order does not matter, and (2) groups larger logical steps, called *actions*, into a maximally parallel form where only important orderings of actions are recorded. This technique was pioneered at Parsifal, and we have barely scratched the surface of its applications.

6.2. Compact Proof Certificates By Bounded Contractions

Participant: Kaustuv Chaudhuri.

An important engineering question in the ProofCert project is that of communicating, manipulating, and storing formal proof certificates. A fully detailed proof certificate, especially one generated by proof search, can be very large. Using such proofs would require a high bandwidth interface between the proof producer and consumer, which limits the scalability of the *ensemble of proving systems* approach. It is therefore natural to ask if there are more compact formats for proof certificates. The ideal format would have a tunable level of detail, so that the size of the certificates can be tailored to the application domain.

Suppose the proof consumer is equipped with some proof search capabilities. What then needs to be transmitted to the consumer to guarantee that it can check a proof within desired complexity bounds? It turns out that there is a systematic and general answer to this problem: use *focusing* and record only the “decision” rules of focusing in the proof certificate. From a high level perspective, this answer is equivalent to designing a proof system where the contraction rules are carefully bounded.

A proposal along these lines was published at the CPP 2012 conference [21]. In fact, this paper solves a harder than necessary problem by building proof certificates for linear logic, where unconstrained proof search has very high complexity even in the propositional fragment. The proposed solution is a spectrum of certificates that trades off the size of the certificate for the complexity of checking the certificate. At one end we have a very compact certificate that essentially amounts to a maximum depth of the proof, but reconstructing a proof with only a depth bound tends to be infeasible as the search space grows super-exponentially with the depth. Certificates at other end of the spectrum contain information about all the contractions in the proof; these certificates can be checked deterministically, in time proportional to the size of the certificate. Moreover, there is a simple abstraction mechanism between different levels of detail in this spectrum that allows for a *proof elaborator* to alter the level of detail in the certificate.

6.3. A Two-level Approach to Reasoning about Computation

Participant: Dale Miller.

In a paper that appeared in the J. of Automated Reasoning, Gacek, Miller, and Nadathur [12] described the foundations and architecture of a new interactive theorem prover capable of reasoning with rich collections of inductive and coinductive relations. This prover, called Abella, also contains the "generic" quantifier ∇ that provides a direct and elegant treatment of term-level binding.

A novel aspect of Abella is that it can define provability in various simple logics and can also reason about provability in such logics. The current system includes a *specification logic* that is a (restricted) intuitionistic logic programming language (a sublanguage of λProlog). The main logic of Abella is then the second logic, called the *reasoning logic*, and it is capable of reasoning about provability in the specification language.

This approach to reasoning about computation has interesting applications. For example, the reasoning logic is aware of the fact that the cut and substitution rules can be eliminated in the specification logic. As a consequence, the notoriously difficult "substitution lemmas" that occur repeated in the study of operational semantics are proved essentially for free (that is, they are an immediate consequence of cut-elimination).

In [17], Accattoli showed that when one reasons about the *untyped* λ-calculus, the specification logic is often not needed. In particular, Accattoli reinterpreted the formalization by G. Huet of the meta-theory of λ-calculus residuals in Abella and showed that the resulting meta-theory had a much more elegant and natural specification than the one presented early by Huet in Coq. While the use of two-levels of logic was not important for this particular (untyped) example, other aspects of Abella—relation specifications, ∇-quantification, and strong induction principles—were critical for improving the expressivity of this prover.

6.4. A Non-local Method for Robustness Analysis of Floating Point Programs

Participants: Dale Miller, Ivan Gazeau.

Programs that must deal with floating point programs and their associate errors can have erratic behavior. In particular, a program that yields outputs that depend continuously on their inputs (in an idealized arithmetic setting) can behave non-continuously when using floating point arithmetic. There are few tools for reasoning about program correctness in a setting that allows for such discontinuous operators.

In [23], Gazeau, Miller, and Palamidessi provide an approach to reason about some programs that are not continuous. In that paper, they introduce the notion of "robustness", which intuitively means that if the input to the program changes less than a fixed small amount then the output changes only slightly. This notion is useful in the analysis of rounding error for floating point programs because it helps to establish bounds on output errors introduced by both measurement errors and by floating point computation. Compositional methods often do not work since key constructs—like the conditional and the while-loop—are not robust. The authors proposed a method for proving the robustness of a while-loop. This method is non-local in the sense that instead of breaking the analysis down to single lines of code, it checks certain global properties of its structure. This paper shows that both the CORDIC computation of the cosine and Dijkstra's shortest path algorithm are robust.

6.5. Herbrand Confluence

Participants: Stefan Hetzl, Lutz Straßburger.

It is well-known that cut-elimination in the sequent calculus for classical first-order logic is in its most general form, is neither confluent nor strongly normalizing. But if one takes a coarser (and mathematically more realistic) look at cut-free proofs, one can analyze which witnesses they choose for which quantifiers, or in other words: one can only consider the Herbrand-disjunction of a cut-free proof. This yields a surprising confluence result for a natural class of proofs: all (possibly infinitely many) normal forms of the non-erasing cut reduction lead to the same Herbrand-disjunction. This result has been presented at CSL 2012 [25].

6.6. Semi-Star-Autonomous Categories

Participants: Willem Heijltjes, Lutz Straßburger.

A curious aspect of Girard's proof nets for multiplicative linear logic without units is that, despite being a canonical representation of proof, their categorical semantics is not obvious—this in contrast to the situation *with* units, where star-autonomous categories form a natural semantics, but no canonical proof nets are known.

In the middle of the past decade several proposals for a categorical semantics of proof nets, a notion of *semi-star-autonomous* categories, were investigated: by Robin Houston and Dominic Hughes, by Kosta Došen, and by François Lamarche and Lutz Straßburger.

The present effort by Willem Heijltjes and Lutz Straßburger completes the notion in such a way that proof nets constitute the *free* semi-star-autonomous category.

6.7. Foundations and applications of explicit substitutions

Participant: Beniamino Accattoli.

Starting from the study of Linear Logic proof nets, a new approach to explicit substitutions for ł-calculus has recently been introduced by Accattoli and D. Kesner [31]. This approach has been systematically explored by Accattoli and his co-authors.

The rewriting theory of these new explicit substitutions *at a distance* has been studied in [11] and [16]. In [11] Accattoli and Kesner study the preservation of λ-calculus strong normalization (PSN) when explicit substitutions are extended with permutative axioms allowing to swap constructors in the term, generalizing considerably the already difficult case of PSN with composition of substitutions. In [16] Accattoli developed an abstract technique for proving factorizations theorems for generic explicit substitution calculi. The factorization theorem for λ-calculus says that any reduction can be re-organized as an *head* reduction followed by a non-head reduction.

In [16] it is shown how to prove this theorem in an uniform way for many explicit subsitutions calculi. The technique emerged as a generalization of the proofs for explicit substitutions at a distance, which are simpler than usual explicit substitutions and thus lead to cleaner and more compact arguments, easier to generalize.

Applications of explicit substitutions at a distance have been studied in [19], [18], [20]. In [19] Accattoli and Dal Lago show that the length of the head reduction in calculi at a distance is a measure of time complexity. More precisely, they show that such a quantity is polynomially related (in both directions) to the cost of evaluating with Turing Machines. This result is an important step forward towards the solution of the long-standing open problem of finding a time cost model for ł-calculus.

In [20] Accattoli and Paolini apply substitutions at a distance in a call-by-value setting. They show that in this new framework there is a natural characterization of *solvability*, an important notion related to denotational semantics and the representation of partial recursive functions. In [26] (a work presented to a workshop and currently submitted to the post-proceedings of the workshop) Accattoli shows the tight relations between the framework in [20] and linear logic proof nets, providing a new characterization of the proof nets representing the call-by-value λ-calculus.

Finally, in [18] Accattoli and Kesner introduce a calculus generalizing many different extensions of λ-calculus with permutations, appeared in various contexts (studies about call-by-value, postponing of reductions, monadic languages, etc) and prove confluence and preservation of strong normalization, exploiting and extending their own results in [11].

6.8. Sequent Calculus with Calls to a Decision Procedure

Participants: Mahfuza Farooque, Stéphane Lengrand.

In the PSI project, we have designed a version of the focussed sequent calculus (for first-order classical logic) that can call external decision procedures. Since the last Activity Report, we have finished proving the essential meta-theory for it: soundness, invertibility of asynchronous rules, cut-elimination, the fact that polarities do not affect provability but only the shape of proofs, and finally completeness.

The first properties are the object of [27], while the latter ones have been obtained later in 2012.

A side-product of this meta-theory is a technical device that could be used to encode other techniques from automated reasoning like *connection tableaux*.

Secondly, we have encoded the SMT-solving algorithm DPLL(T) as the incremental construction of proof-trees in that sequent calculus [29], [28]. A very interesting aspect of the encodings is that the basic rules of DPLL(T) makes use of cuts on atoms in sequent calculus, while the advanced jrules (e.g. backjumping) makes use of general cuts. This sheds a new light on the computational speed-ups that those advanced rules provide.

We have done the encoding for two distinct presentations of DPLL(T) in the literature, and we have formalised the connection between those two descriptions [29].

6.9. Martin-Löf Identity Type in the Category of Small Categories

Participant: François Lamarche.

For the last five or six years there has been a surge of interest in finding models for the identity type in Martin-Löf type theory, and it has been clear for some time that there was a tight connection with path objects in abstract homotopy theory. A lot of proposals have been made, but there are very few semantics that fit the necessary requirements of having dependent products and also an identity type which is fully stable under substitution. The most famous model of the sort is the one proposed by Voevodsky, in his Univalent Foundations project, which uses for base category the category of simplicial sets and models dependent types by the means of Kan Fibrations. In [13] François Lamarche proposes another such model, where the base category is the category of small categories, and dependent types are modelled with Grothendieck bifibrations (maps between categories that are Grothendieck fibrations as well as their duals between the opposite categories). The full requirements of modelling Martin-Löf type theory are met. Calculations show that the model shows some amount of degeneracy "in dimensions above 2" for the associativity of equality (which should not be strict in any dimension), which is a great improvement over the models on strict groupoids and strict ω-groupoids. The construction that models the identity type is a concrete path functor for categories. It is showing itself to be very useful in homotopy theory.

7. Partnerships and Cooperations

7.1. European Initiatives

7.1.1. FP7 Projects

7.1.1.1. Proofcert

Title: ProofCert: Broad Spectrum Proof Certificates

Type: IDEAS

Instrument: ERC Advanced Grant (Advanced)

Duration: January 2012 - December 2016

Coordinator: Inria (France)

See also: https://team.inria.fr/parsifal/proofcert/

Abstract: 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 times 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.

7.1.2. Collaborations in European Programs, except FP7

7.1.2.1. STRUCTURAL: ANR blanc International

Participants: Kaustuv Chaudhuri, Nicolas Guenot, Willem Heijltjes, François Lamarche, Dale Miller, Lutz Straßburger.

Title: Structural and computational proof theory

Duration: 01/01/2011 – 31/12/2013

Partners:

University Paris VII, PPS (PI: Michel Parigot)

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

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

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

Total funding by the ANR: 242 390,00 EUR (including 12 000 EUR pôle de compétivité: SYSTEMTIC Paris région)

This project is a consortium of four partners, two French and two Austrian, who are all internationally recognized for their work on structural proof theory, but each coming from a different tradition. One of the objective of the project is build a bridge between these traditions and develop new proof-theoretic tools and techniques of structural proof theory having a strong potential of applications in computer science, in particular at the level of the models of computation and the extraction of programs and effective bounds from proofs.

On one side, there is the tradition coming from mathematics, which is mainly concerned with first-order logic, and studies, e.g., Herbrand's theorem, Hilbert's epsilon-calculus, and Goedel's Dialectica interpretation. On the other side, there is the tradition coming from computer science, which is mainly concerned with propositional systems, and studies, e.g., Curry-Howard isomorphism, algebraic semantics, linear logic, proof nets, and deep inference. A common ground of both traditions is the paramount role played by analytic proofs and the notion of cut elimination. We will study the inter-connections of these different traditions, in particular we focus on different aspects and developments in deep inference, the Curry-Howard correspondence, term-rewriting, and Hilbert's epsilon calculus. As a byproduct this project will yield a mutual exchange between the two communities starting from this common ground, and investigate, for example, the relationship between Herbrand expansions and the computational interpretations of proofs, or the impact of the epsilon calculus on proof complexity.

Besides the old, but not fully exploited, tools of proof theory, like the epsilon-calculus or Dialectica interpretation, the main tool for our research will be deep inference. Deep inference means that inference rules are allowed to modify formulas deep inside an arbitrary context. This change in the application of inference rules has drastic effects on the most basic proof theoretical properties of the systems, like cut elimination. Thus, much of the early research on deep inference went into reestablishing these fundamental results of logical systems. Now, deep inference is a mature paradigm, and enough theoretical tools are available to think to applications. Deep inference provides new properties, not available in shallow deduction systems, namely full symmetry and atomicity, which open new possibilities at the computing level that we intend to investigate in this project. We intend to investigate the precise relation between deep inference and term rewriting, and hope to develop a general theory of analytic calculi in deep inference. In this way, this project is a natural continuation of the ANR project INFER which ended in May 2010.

7.1.2.2. PHC Procope: From Proofs to Counterexamples for Programming

Participants: Kaustuv Chaudhuri, Nicolas Guenot, Willem Heijltjes, Lutz Straßburger.

Title: From Proofs to Counterexamples for Programming

Duration: 01/01/2012 – 31/12/2013

German Partner: University of Bonn, Institute for Computer Science (Department III)

Finding counterexamples is an endeavor which is as important as proving theorems. But while the latter has seen a huge amount of research effort—we have nowadays a large quantity of tools for automated and interactive theorem proving—the former has mainly been neglegted by proof theorists. One of the reasons is that finding counterexamples or countermodels has been considered a model theoretical activity, rather than a proof theoretical one. Only recently, researchers have begun to explore the well-known duality between "proof search" and "search for countermodels" in a purely proof theoretical way. The main objective of this collaboration is to develop the necessary proof theory for automatically generating such counterexamples in a more general setting.

7.1.2.3. PHC Germaine de Staël: Extending the Realm of the Curry-Howard-Correspondence

Participants: Nicolas Guenot, Willem Heijltjes, Lutz Straßburger.

Title: Extending the Realm of the Curry-Howard-Correspondence

Duration: 01/01/2011 – 31/12/2012

Swiss Partner: University of Bern, Institut für Informatik und angewandte Mathematik (IAM)

The Curry-Howard correspondence between proofs and programs is probably the most interesting and surprising connection between mathematics and computer science. It was discovered in the 1960s, but its main development started in the 1980s. The basis of the correspondence is a correspondence between intuitionistic proofs and typed functional programs (written as terms of lambda-calculus).

Our goal is to develop such a correspondence for new formalisms, like hypersequents, nested sequents and deep inference, in order to better understand their proofs and, we hope, either to discover new programming constructs or to give a new logical interpretation to existing ones.

7.2. International Initiatives

7.2.1. Inria Associate Teams

7.2.1.1. RAPT

Participants: Beniamino Accattoli, Kaustuv Chaudhuri, Quentin Heath, Dale Miller, Yuting Wang.

Title: Applying Recent Advances in Proof Theory for Specification and Reasoning

Inria principal investigator: Kaustuv Chaudhuri

International Partner:

Institution: McGill University (Canada)

Laboratory: School of Computer Science

Researcher: Prof. Brigitte Pientka

International Partner:

Institution: Carnegie Mellon University (United States)

Laboratory: Department of Computer Science

Researcher: Prof. Frank Pfenning

International Partner:

Institution: University of Minnesota (United States)

Laboratory: Department of Computer Science and Engineering

Researcher: Prof. Gopalan Nadathur

Duration: 2011 - 2013

See also: http://www.lix.polytechnique.fr/~kaustuv/rapt/

Many aspects of computation systems, ranging from operational semantics, interaction, and various forms of static analysis, are commonly specified using inference rules, which themselves are formalized as theories in a logical framework. While such a use of logic can yield sophisticated, compact, and elegant specifications, formal reasoning about these logic specifications presents a number of difficulties. The RAPT project will address the problem of reasoning about logic specifications by bringing together three different research teams, combining their backgrounds in type theory, proof theory, and the building of computational logic systems. We plan to develop new methods for specifying computation that allow for a range of specification logics (eg, intuitionistic, linear, ordered) as well as new means to reason inductively and co-inductively with such specifications. New implementations of reasoning systems are planned that use interactive techniques for deep meta-theoretic reasoning and fully automated procedures for a range of useful theorems.

7.2.2. Inria International Partners

7.2.2.1. Eternal: Inria ARC

Participants: Kaustuv Chaudhuri, Dale Miller, Lutz Straßburger.

Title: Interactive Resource Analysis

webpage: http://eternal.cs.unibo.it/

Inria principal investigator: Dale Miller

Inria Partner:

Institution: Inria

Team: FOCUS

Researcher: Ugo Dal Lago

Inria Partner:

Institution: Inria

Team: pi.r2

Researcher: Pierre-Louis Curien

Duration: 2011 - 2013

This project aims at putting together ideas from Implicit Computational Complexity and Interactive Theorem Proving, in order to develop new methodologies for handling quantitative properties related to program resource consumption, like execution time and space. The task of verifying and certifying quantitative properties is undecidable as soon as the considered programming language gets close to a general purpose language. So, full-automatic techniques in general cannot help in classifying programs in a precise way with respect to the amount of resources used and moreover in several cases the programmer will not gain any relevant information on his programs. In particular, this is the case for all the techniques based on the study of structural constraints on the shape of programs, like many of those actually proposed in the field of implicit computational complexity. To overcome these limitations, we aim at combining the ideas developed in the linear logic approach to implicit computational complexity with the ones of interactive theorem proving, getting rid of the intrinsic limitations of the automatic techniques. In the obtained framework, undecidability will be handled through the system's user, who is asked not only to write the code, but also to drive the semi-automatic system in finding a proof for the quantitative properties of interest. In order to reduce the user effort and allow him to focus only on the critical points of the analysis, our framework will integrate implicit computational complexity techniques as automatic decision procedures for particular scenarios. Moreover, in order to be widely applicable, the modularity of the framework will permit to deal with programs written in different languages and to consider different computational resources. The kind of study proposed by this project has been almost neglected so far. Here, we aim at providing such a framework for both theoretic investigations and for testing in practice the effectiveness of the approach.

7.3. International Research Visitors

7.3.1. Visits of International Scientists

Brigitte Pientka, Associate Professor, McGill University
February 21 – 24

Gopalan Nadathur, Professor, University of Minnesota
July 10 – 12

Elaine Pimentel, Associate Professor, Universidade Federal de Minas Gerais
June 6 – July 17

Chuck Liang, Professor, Hofstra University
March 6 – May 6 and December 17 – 24

7.3.2. Internships

Yuting WANG (May – August)

Subject: Development of the Abella theorem prover.

Institution: University of Minnesota (United States)

Florence Clerc (March – July)

Subject: Relating double-negation translations and focused proof systems

Institution: Master Parisien de Recherche en Informatique

Zakaria Chihani (April – September)

Subject: Proof certificates for some basic proof systems in classical logic

Institution: Master Parisien de Recherche en Informatique

7.3.3. Visits to International Teams

Stefan Hetzl has visited the Vienna University of Technology four times, for a total of 36 days, within the framework of the FWF/ANR Structural project.

8. Dissemination

8.1. Scientific Animation

8.1.1. Organization

Dale Miller served as Program Committee Chair for the following three meetings: CPP 2012: Second International Conference on Certified Programs and Proofs, 13-15 December, Kyoto, Japan; IJCAR 2012: International Joint Conference on Automated Reasoning, Manchester, UK, June; FICS 2012: Fixed Points in Computer Science (a satellite workshop of ETAPS 2012), Tallinn, Estonia, 24 March.

Dale Miller served on the Program Committees of the following three meetings: LFMTP'12: Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, 9 September, Copenhagen, Denmark; LAM 2012: Fifth International Workshop on Logics, Agents, and Mobility, June, Hamburg, Germany; LPAR-18: The 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning. Merida, Venezuela, 11-15 March.

Dale Miller organized the Special Session on Structural Proof Theory and Computing at the 2012 Annual Meeting of the Association of Symbolic Logic (ASL), Madison, Wisconsin, 31 March - 3 April.

Kaustuv Chaudhuri and Stefan Hetzl organized the Collegium Logicum 2012 workshop at Inria Saclay during 15–16 November

Kaustuv Chaudhuri and Brigitte Pientka organized the RAPT/PROMIS workshop in Montreal, Canada, during 14–18 October

8.1.2. Invited Talks

Dale Miller gave an invited talk at the Journees nationales du GDR-IM, 25-26 January 2012, University of Paris 7.

Dale Miller gave an invited talk at the ANR-DFG Hypothese Workshop titled Different Aspects of Proof Theory, ENS Paris, 3 May 2012.

Dale Miller gave research talks at the following venues: Summer Workshop on Proof Theory, Pisa, Italy, 12-15 June; CHocCoLa seminar, ENS Lyon, 15 March; Workshop on Certificates and Computation, ITU Copenhagen, 12 March; Department of Mathematics, Carnegie Mellon University, 5 April.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Licence : Stéphane Graham-Lengrand teaches 50 hours (eq. TD) in L3 at Ecole Polytechnique in the course “INF431: Algorithmique et programmation”.

Master : Dale Miller taught 12 hours at MPRI (Master Parisien de Recherche en Informatique) in the Course 2-1: Logique linéaire et paradigmes logiques du calcul.

Master : Stéphane Graham-Lengrand teaches 36 hours (eq. TD) in M1 at Ecole Polytechnique in the course “INF551: Computer-aided reasoning”, and 15 hours (eq. TD) in M2 at Master Parisien de Recherche en Informatique (MPRI) on “Curry-howard correspondence for classical logic”.

8.2.2. Supervision

PhD in progress :

1. Zakaria Chihani, since October 2012, supervisor: Dale Miller
2. Mahfuza Farooque, since October 2010, co-supervisor: Stéphane Graham-Lengrand

3. Ivan Gazeau, since October 2009, co-supervisor: Dale Miller
4. Nicolas Guenot, since October 2008, supervisor: Lutz Straßburger
5. Hernán-Pablo Vanzetto, since October 2010, co-supervisor: Kaustuv Chaudhuri

8.2.3. Juries

Dale Miller served as a reporter for the habilitation of Stefan Hetzl (Fall 2012) and was the president of the jury for the habilitation of Frédéric Blanqui, 13 July 2012.

Dale Miller served as a member of the PhD jury for Robert Simmons, Carnegie Mellon University, 22 October 2012. He was also “rapporteur” for the PhD of Nicolas Pouillard, University of Paris 7, 13 January 2012.

9. Bibliography

Major publications by the team in recent years

[1] 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, p. 159–173 [*DOI :* 10.4230/LIPIcs.CSL.2011.159].

[2] A. Gacek, D. Miller, G. Nadathur. *Nominal abstraction*, in "Information and Computation", 2011, vol. 209, n^{o} 1, p. 48–73, http://arxiv.org/abs/0908.1390.

[3] 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, p. 284–293 [*DOI :* 10.1109/LICS.2010.12], http://www.lix.polytechnique.fr/~lutz/papers/AFII.pdf.

[4] 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, p. 246–261.

[5] S. Lengrand, R. Dyckhoff, J. McKinna. *A focused sequent calculus framework for proof search in Pure Type Systems*, in "Logical Methods in Computer Science", 2010, To appear.

[6] C. Liang, D. Miller. *Focusing and Polarization in Linear, Intuitionistic, and Classical Logics*, in "Theoretical Computer Science", 2009, vol. 410, n^{o} 46, p. 4747–4768.

[7] C. Liang, D. Miller. *A Focused Approach to Combining Logics*, in "Annals of Pure and Applied Logic", 2011, vol. 162, n^{o} 9, p. 679–697 [*DOI :* 10.1016/j.apal.2011.01.012].

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

[9] L. Strassburger. *On the Axiomatisation of Boolean Categories with and without Medial*, in "Theory and Applications of Categories", 2007, vol. 18, n^{o} 18, p. 536–601, http://arxiv.org/abs/cs.LO/0512086.

[10] 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.

Publications of the year

Articles in International Peer-Reviewed Journals

[11] B. ACCATTOLI, D. KESNER. *Preservation of Strong Normalisation modulo permutations for the structural lambda-calculus*, in "Logical Methods in Computer Science", 2012, vol. 8, n^o 1 [*DOI :* 10.2168/LMCS-8(1:28)2012].

[12] A. GACEK, D. MILLER, G. NADATHUR. *A two-level logic approach to reasoning about computations*, in "J. of Automated Reasoning", 2012, vol. 49, n^o 2, p. 241–273 [*DOI :* 10.1007/S10817-011-9218-1], http://arxiv.org/abs/0911.2993.

[13] F. LAMARCHE. *Modelling Martin Löf Type Theory in Categories*, in "Journal of Applied Logic", June 2012, conference "Logic, Categories, Semantics, Bordeaux, France", scheduled to appear in June 2013, http://hal.inria.fr/hal-00706562.

[14] C. LIANG, D. MILLER. *Kripke Semantics and Proof Systems for Combining Intuitionistic Logic and Classical Logic*, in "Annals of Pure and Applied Logic", February 2013, vol. 164, n^o 2, p. 86–111 [*DOI :* 10.1016/J.APAL.2012.09.005].

[15] L. STRASSBURGER. *Extension without cut*, in "Ann. Pure Appl. Logic", 2012, vol. 163, n^o 12, p. 1995–2007.

International Conferences with Proceedings

[16] B. ACCATTOLI. *An Abstract Factorization Theorem for Explicit Substitutions*, in "23rd International Conference on Rewriting Techniques and Applications (RTA)", Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2012, vol. 15, p. 6–21 [*DOI :* 10.4230/LIPICS.RTA.2012.6].

[17] B. ACCATTOLI. *Proof pearl: Abella formalization of lambda calculus cube property*, in "Second International Conference on Certified Programs and Proofs (CPP)", Kyoto, Japan, LNCS, Springer, December 2012, vol. 7679, p. 173–187 [*DOI :* 10.1007/978-3-642-35308-6_15].

[18] B. ACCATTOLI, D. KESNER. *The Permutative λ-Calculus*, in "18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)", LNCS, Springer, 2012, vol. 7180, p. 23–36 [*DOI :* 10.1007/978-3-642-28717-6_5].

[19] B. ACCATTOLI, U. D. LAGO. *On the Invariance of the Unitary Cost Model for Head Reduction*, in "23rd International Conference on Rewriting Techniques and Applications (RTA)", Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2012, vol. 15, p. 22–37 [*DOI :* 10.4230/LIPICS.RTA.2012.22].

[20] B. ACCATTOLI, L. PAOLINI. *Call-by-Value Solvability, Revisited*, in "Eleventh International Symposium on Functional and Logic Programming (FLOPS)", LNCS, Springer, 2012, vol. 7294, p. 4–16 [*DOI :* 10.1007/978-3-642-29822-6_4].

[21] K. CHAUDHURI. *Compact Proof Certificates for Linear Logic*, in "Proceedings of the Second Internatinal Conference on Certified Programs and Proofs (CPP)", Kyoto, Japan, C. HAWBLITZEL, D. MILLER (editors), LNCS, Springer, December 2012, vol. 7679, p. 208–223 [*DOI :* 10.1007/978-3-642-35308-6_17].

[22] K. CHAUDHURI, S. HETZL, D. MILLER. *A Systematic Approach to Canonicity in the Classical Sequent Calculus*, in "Computer Science Logic (CSL'12): 21st Annual Conference of the EACSL", P. CÉGIELSKI, A. DURAND (editors), Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, September 2012, vol. 16, p. 183–197 [*DOI :* 10.4230/LIPICS.CSL.2012.183].

[23] I. GAZEAU, D. MILLER, C. PALAMIDESSI. *A non-local method for robustness analysis of floating point programs*, in "QAPL - Tenth Workshop on Quantitative Aspects of Programming Languages", Tallinn, Estonia, M. MASSINK, H. WIKLICKY (editors), Electronic Proceedings in Theoretical Computer Science, March 2012, vol. 85, p. 63–76 [*DOI :* 10.4204/EPTCS.85.5], http://hal.inria.fr/hal-00665995.

[24] S. HETZL. *Project Presentation: Algorithmic Structuring and Compression of Proofs (ASCOP)*, in "Conference on Intelligent Computer Mathematics (CICM) 2012", J. JEURING, ET AL. (editors), Lecture Notes in Artificial Intelligence, Springer, 2012, vol. 7362, p. 437–441.

[25] S. HETZL, L. STRASSBURGER. *Herbrand-Confluence for Cut Elimination in Classical First Order Logic*, in "Computer Science Logic (CSL'12) - 26th International Workshop/21st Annual Conference of the EACSL, CSL 2012", Fontainebleau, France, P. CÉGIELSKI, A. DURAND (editors), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, September 3-6 2012, vol. 16, p. 320–334.

Conferences without Proceedings

[26] B. ACCATTOLI. *Proof nets and the call-by-vlaue lambda calculus*, in "7th Workshop on Logical and Semantic Frameworks, with Applications (LSFA)", 2012.

Research Reports

[27] M. FAROOQUE, S. LENGRAND. *A sequent calculus with procedure calls*, Inria, December 2012, http://hal.archives-ouvertes.fr/hal-00690577.

[28] M. FAROOQUE, S. LENGRAND, A. MAHBOUBI. *Simulating the DPLL(T) procedure in a sequent calculus with focusing*, Inria, April 2012, http://hal.inria.fr/hal-00690392.

[29] M. FAROOQUE, S. LENGRAND, A. MAHBOUBI. *Two simulations about DPLL(T)*, Inria, March 2012, http://hal.archives-ouvertes.fr/hal-00690044.

References in notes

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

[31] B. ACCATTOLI, D. KESNER. *The Structural λ-Calculus*, in "CSL", 2010, p. 381–395 [*DOI :* 10.1007/978-3-642-15205-4_30].

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

[33] B. E. AYDEMIR, A. BOHANNON, M. FAIRBAIRN, J. N. FOSTER, B. C. PIERCE, P. SEWELL, D. VYTINIOTIS, G. WASHBURN, S. WEIRICH, S. ZDANCEWIC. *Mechanized Metatheory for the Masses: The PoplMark Challenge*, in "Theorem Proving in Higher Order Logics: 18th International Conference", LNCS, Springer-Verlag, 2005, p. 50–65.

[34] 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, p. 278–292 [*DOI :* 10.1007/978-3-642-14203-1], http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/ijcar10.pdf.

[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, D. DOLIGEZ, L. LAMPORT, S. MERZ. *Verifying Safety Properties With the TLA^{+} Proof System*, in "Fifth International Joint Conference on Automated Reasoning (IJCAR 2010)", Edinburgh, United Kingdom, J. GIESL, R. HÄHNLE (editors), LNAI, Springer, July 2010, vol. 6173, p. 142–148 [*DOI :* 10.1007/978-3-642-14203-1_12], http://hal.inria.fr/inria-00534821.

[37] K. CHAUDHURI, D. MILLER, A. SAURIN. *Canonical Sequent Proofs via Multi-Focusing*, in "Fifth IFIP International Conference on Theoretical Computer Science", G. AUSIELLO, J. KARHUMÄKI, G. MAURI, L. ONG (editors), IFIP International Federation for Information Processing, Boston: Springer, September 2008, vol. 273, p. 383–396, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tcs08trackb.pdf.

[38] 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, p. 33–44, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lics08a.pdf.

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

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

[41] 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, p. 1–36, http://arxiv.org/abs/0709.1205.

[42] 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, p. 54–68.

[43] A. S. HENRIKSEN. *Using LJF as a Framework for Proof Systems*, University of Copenhagen, 2009, http://hal.inria.fr/inria-00442159/en/.

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

[45] 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, p. 80–136, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/mcdowell01.pdf.

[46] 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, p. 434–445.

[47] D. Miller. *Communicating and trusting proofs: The case for broad spectrum proof certificates*, June 2011, Available from author's website.

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

[49] 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, p. 125–157.

[50] D. Miller, E. Pimentel. *Using linear logic to reason about sequent systems*, in "International Conference on Automated Reasoning with Analytic Tableaux and Related Methods", U. Egly, C. Fermüller (editors), LNCS, Springer, 2002, vol. 2381, p. 2–23, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tableaux02.pdf.

[51] 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, p. 118–127, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lics03.pdf.

[52] V. Nigam, D. Miller. *A framework for proof systems*, March 2009, Extended version of IJCAR08 paper. Submitted.

[53] 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, p. 202–206.

[54] 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, p. 15–21.

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

[56] The Coq Development Team. *The Coq Proof Assistant Version 8.3 Reference Manual*, Inria, October 2010.

[57] 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.