INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Parsifal

Proof search and reasoning with logic specifications

Saclay - Île-de-France

Theme : Programs, Verification and Proofs

Activity Report

2010

Table of contents

1. Team

Research Scientists

Dale Miller [Team Leader, Senior Researcher, INRIA Saclay – Île-de-France]
Kaustuv Chaudhuri [Junior Researcher, INRIA Saclay – Île-de-France]
Joëlle Despeyroux [Junior Researcher, INRIA Sophia Antipolis – Méditerranée]
Stéphane Lengrand [Junior Researcher, CNRS]
Lutz Strassburger [Junior Researcher, INRIA Saclay – Île-de-France]

PhD Students

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]
Anne-Laure Schneider [Allocation Internationale de thèse Gaspard Monge, started 1 September 2010]
Hernan-Pablo Vanzetto [Contrat doctoral INRIA via INRIA-MSR, started 1 November 2010]
Alexandre Viel [Allocataire École Normale Supérieure, Paris, started 1 October 2008]
François Wirion [Bourse INRIA, started 1 December 2008]

Post-Doctoral Fellows

Andrew Gacek [INRIA, 1 October 2009 - 30 September 2010]
Tom Gundersen [INRIA, 1 December 2009 - 30 November 2010]
Clément Houtmann [INRIA, 1 September 2010 - 31 August 2011]

Visiting Scientists

Kai Brünnler [Researcher, IAM, University of Bern, Switzerland, 1–30 September 2010]
Brigitte Pientka [Associate Professor, McGill University, Montreal, Canada, 1–28 February 2010]
Elaine Pimentel [Associate Professor, Universidade Federal de Minas Gerais, Brazil, 28 May – 10 July 2010]

Administrative Assistant

Marie-Jeanne Gaffard

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
- Probabilistic and stochastic reasoning systems commonly used in security, networking, and biological domains

2.2. Highlights

- Vivek Nigam (PhD from Parsifal September 2009) was awarded an Alexander von Humboldt scholarship to join Martin Hoffman’s group in LMU (Munich, Germany) for two years (2010-2012)

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 [49], and normalization in different logics can justify the design of new and different functional programming languages [28].
- *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 [53], and different proof search strategies can be used to justify the design of new and different logic programming languages [52].

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 ro 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 [29] 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 [64] that implements the Calculus of Inductive Constructions (CIC) is designed to facilitate the expression of computational features of proofs directly as executable functional programs, but general proof search techniques for Coq are rather primitive. In contrast, the Twelf system [57] that is based on the LF type theory (a subsystem of the CIC), is based on relational specifications in canonical form (*i.e.*, without redexes) for which there are sophisticated automated reasoning systems such as meta-theoretic analysis tools, logic programming engines, and inductive theorem provers. In recent years, there has been a push towards combining search and normalization in the same type-theoretic framework. The Beluga system [58], for example, is an extension of the LF type theory with a purely computational meta-framework where operations on inductively defined LF objects can be expressed as functional programs.

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

3.2. Focused proof systems

Focusing [29] is a general observation that proofs in sequent calculi can be organized into an alternating pair of dual *phases* – negative (sometimes called *asynchronous*) and positive (sometimes called *synchronous*). Each phase consists of a maximal chain of inferences of the the same *polarity*, *i.e.*, the phases represent *synthetic*, *macro*, or *"big step"* inference rules for clumps of connectives of the same polarity. For example, focusing tells us that the top level connective in the formula $A \otimes (B \oplus C)$, assuming A, B and C are negative, is the ternary connective $- \otimes (- \oplus -)$ instead of the composition of two binary connectives; in particular, $B \oplus C$ need not even be considered as a subformula. Indeed, focusing has proven to be crucial in controlling the search behavior of automated theorem provers [1], [51].

Focusing is very important for logical specifications of computations because synthetic inference rules have a direct correspondence with computational steps. When a system is encoded logically, we can identify at least three levels of *adequacy* of the encoding:

- *Global adequacy*, wherein the encoding does not necessarily respect the structure of the computation, but where soundness is ensured globally. For instance, an encoding of one proof system in another is globally adequate if it preserves truth and provability.
- *Full adequacy*, where the structure of the entire computation is preserved. For encodings of proof systems, full adequacy corresponds to a one-to-one correspondence between full proofs in the source and the target of the encoding.
- *Local adequacy*, where the structure of individual steps of a computation is preserved, *i.e.*, the encoding is a bi-simulation. An encoding of a proof system is locally adequate if the encoding preserves the individual inference rules of the proof system.

Locally adequate encodings give the best indication of the strength and generality of proof systems, but in all but the most trivial cases such encodings only exist if the target of the encoding is a focused proof system. The members of the Parsifal team have identified and proved a number of adequacy results in recent years [54], [48], [56], [18].

Focusing is therefore an important tool in the study of *universality* of proof systems. We already know that linear logic can serve as a uniform meta-language for a number of proof systems, both classical and intuitionistic, but these encoded systems generally are not able to communicate with each other. Liang and Miller have been building proof systems for combinations of classical, intuitionistic, and linear logics that allows proofs in these different systems to communicate via carefully chosen cuts.

3.3. Reasoning about logic specifications

A long term project of members of the Parsifal team has been the design of a powerful logic to reason about computational specifications written in logic. Coming up with the design of a logic that allows reasoning richly over relational specifications involving bindings in syntax has been a long standing problem, dating from at least the early papers by McDowell and Miller [50] [7]. In 2010, Gacek, Miller, and Nadathur (a colleague from the University of Minnesota) have completed the design the the logic $\mathcal{G}$ [39] that extends earlier work on this topic by including a novel generalization to syntactic identity. This extensions involved adding the "nominal abstraction" operator. With the addition of this predicate the resulting logic $\mathcal{G}$ gained enough expressive power to allow for natural and declarative descriptions of invariants over logic-based context.

The presence of nominal abstraction in $\mathcal{G}$ makes it possible for that logic to express predicates that strongly resemble those found in Pitt's "nominal logic" [59]. While $\mathcal{G}$ and nominal logic different in several ways, it is possible to find an interesting subset of both logics that do, in fact, correspond directly. In particular, Gacek showed [21] that αProlog [37] (a subset of nominal logic) can be directly translated into subset of $\mathcal{G}$.

The $\mathcal{G}$ logic is the logic that is implemented by the Abella prover of Andrew Gacek. This implementation has permitted a large number of example theorems and proofs to be done completely formally within $\mathcal{G}$. As a result, we have gained a great deal of confidence in the expressive strengths of his logic.

3.4. Deep inference and categorical axiomatizations

Deep inference [42], [43] 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.

One reason for using categories in proof theory is to give a precise algebraic meaning to the identity of proofs: two proofs are the same if and only if they give rise to the same morphism in the category. Finding the right axioms for the identity of proofs for classical propositional logic has for long been thought to be impossible, due to "Joyal's Paradox". For the same reasons, it was believed for a long time that it it not possible to have proof nets for classical logic. Nonetheless, Lutz Strassburger and François Lamarche provided proof nets for classical logic in [3], and analyzed the category theory behind them in [45]. In [10] and [62], one can find a deeper analysis of the category theoretical axioms for proof identification in classical logic. Particular focus is on the so-called *medial rule* which plays a central role in the deep inference deductive system for classical logic.

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.

- Use the results on deep inference to find new axiomatic description of categories of proofs for various logics. So far, this is well understood only for linear and intuitionistic logics. Already for classical logic there is no common accepted notion of proof category. How logics like LINC can be given a categorical axiomatisation is completely open.

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 due to the almost absolute monopoly of the sequent calculus, 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 [40] for linear logic but also in Robinson’s proof nets [60] for classical logic. In this kind of proof nets every link in the net corresponds to a rule application in the sequent calculus.

The concept of deep inference allows to design entirely new kinds of proof nets. The work by Lamarche and Strassburger [61], [46] have extended the theory of proof nets for multiplicative linear logic to multiplicative linear logic with units. This seemingly small step—just adding the units—had for long been an open problem, and the solution was found only by consequently exploiting the new insights coming from deep inference. A proof net no longer just mimics the sequent calculus proof tree, but rather an additional graph structure that is put on top of the formula tree (or sequent forest) of the conclusion. The work on proof nets within the team is focused on the following two directions:

- Extend the work of Lamarche and Strassburger to larger fragments of linear logic, containing the additives, the exponentials, and the quantifiers, as started in [63].
- 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 [3] this takes exponential time (in the size of the net). We hope that eventually deductive proof nets will provide a “bureaucracy-free” formalism for proof search.
- 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. A systematic approach to cut-elimination

One of the main problems of proof theory is to prove cut elimination for new logics. Usually, a cut elimination proof is a tedious case analysis, and, in general, it is very fragile and not modular [41]. That means that a minor change in the deductive system makes the cut elimination proof break down, and for every new system one has to start from scratch.

It is therefore an important research task, to find a more systematic approach to cut elimination proofs. That is to say, to find general guidelines that ensure the cut elimination property for large classes of systems, in a similar way as it has been done for display logics [32].

3.7. Proof search in type theory

Cross-fertilizing ideas between the proof search approach and the proof normalization approach, Lengrand has interacted with the TypiCal (INRIA Saclay) and the πr^2 (INRIA Rocquencourt) project-teams.

In proof assistants based on the proof normalization approach, or Type Theory, it is a hard challenge to design and understand their proof search mechanisms. Based on ideas from [47], a major effort has been spent on using concepts from the proof search approach, like *focused proof systems*, in order to rationalize the implemented mechanisms.

By doing so, we have helped improve the Coq system, by impacting the design of the new version of the tool's proof engine. One of these proof search mechanisms, known as *pattern unification*, has again become a hot topic of Coq's design, after Lengrand's use of Coq to specify a particular algorithm has revealed a drastic need for this missing feature.

It also emerged from Lengrand's interaction with these project-teams, that bridging Type Theory with the proof theory developed at Parsifal confirms the need for more extensionality on the functions programmed in Coq. Efforts to add such extensionality are ongoing.

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 (see Section. 5.3).

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

- The concept of *multi-focusing* [36] 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 [30] 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 progamming languages is the ability to reason about data structures with binding constructs upto 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. 3.3) 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.

Another important feature for the meta-theory of progamming languages is the ability to reason about inductive and co-inductive data structures and algorithms. While systems such as Coq [64] can represent such inductive proofs as fixpoints, there is only very primitive support for general automated reasoning over inductive definitions. The Tac system built in the Parsifal team [16] has been used to investigate automated inductive theorem proving from a more foundational perspective. Tac can already perform a number of sophisticated inductive proofs automatically.

Modern programming languages are increasingly incorporating distributed, concurrent, reactive, and real-time elements. In such languages, it is often necessary to reason not about executions but about *behaviors*, that is, it is necessary to compare the behavior of two different programs instead of characterizing all executions of a single program. The Bedwyr tool [31] built at Parsifal is a symbolic model checker that can automatically prove behavioral equivalence between π-calculus processes. It is a prototype of the kind of formal tools that will be necessary for the programming languages of the future.

4.3. Malleable proof languages

One of the benefits of focused proof systems (see Section.3.2) is the ability to treat computational steps as single synthetic rules. If the computational steps belong to a particular proof search strategy, then it becomes possible to represent, precisely, the traces of that strategy as synthetic proofs.

Recently, 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 [55], [54] or focused intuitionistic logic [44] as the meta-language. In the presence of induction and co-induction, arbitrary finite computations can be embedded into single synthetic steps [16].

It seems clear that a suitably general focused proof system can serve as a universal proof language for a large variety of proof systems. We can identify at least the following major challenges:

- Can focused proof systems serve as a framework for broad spectrum proof certificates for such domains as proof-carrying code or proof-carrying authorization?
- Can one design a proof language based on focused proofs that allows for a variable amount of verbosity in terms of a tunable trade off between simplicity of the proof checker (or, equivalently, the amount of search that the proof checker is allowed to perform) and the size of proof certificates?
- Can one design a generic universal proof checker for a large variety of proof systems?
- How does (co)induction in a focused proof system compare to type systems such as Deduction Modulo [38] or Superdeduction [34] that are parameterized on rewrite systems?

5. Software

5.1. Abella

Participants: Andrew Gacek, Dale Miller.

The earliest versions of the Abella theorem prover was written while Gacek was a PhD student at the University of Minnesota. During Gacek's post doc in the Parsifal team, he and Miller have designed and implemented more features into this prover. One such feature involves simplifying dependences among variables. More explicitly, for Abella to correctly capture the relationship between "meta-variables" and binding, the prover takes several *raising* steps that are responsible for explicitly accounting for the dependence between these two kinds of variables. Given the typing discipline of Abella it is possible to statically determine that certain possible dependences are, in fact, vacuous. Abella is now able to remove these vacuous dependences: such a simplification makes it possible to make simplify many Abella proofs. Gacek, Miller, and Nadathur have also published two papers describing the underlying theory of Abella.

For more information, see the Abella home page.

5.2. Tac

Participants: David Baelde, Dale Miller, Zachary Snow, Alexandre Viel.

Given the team's expertise with the structure of proofs and techniques for automation, we have taken on the implementation of the TAC prover. This prover, written in OCaml, has been used to prove a range of inductive theorems in a completely automatic fashion. The architecture, which is the subject of the conference paper [16], is based on recent work by the team on the structure of focused proofs for induction and co-induction. A goal of this prover is to completely automate a large number of shallow theorems within an inductive and co-inductive setting: proofs of more significant theorems would then be organized as being simple lists of lemmas. While the automatic tactic of this prover will not likely prove particularly hard and deep theorems, the evidence we have gathered so far is that it is useful for automatically proving a great number of routine and shallow (but possibly tedious) theorems.

For more information, see the Tac home page.

5.3. TLAPS

Participants: Kaustuv Chaudhuri, Denis Cousineau [INRIA-MSR], Damien Doligez [INRIA Paris-Rocquencourt, EPI Gallium], Leslie Lamport [Microsoft Research Silicon Valley], Stephan Merz [INRIA Nancy Grand-Est, EPI Veridis], Hernán Vanzetto [Masters Student, Université Henri Poincaré, Nancy].

The TLA+ Proof System (TLAPS) [20] is a formal proof system for Leslie Lamport's *Temporal Logic of Actions* (TLA), a specification language for distributed, concurrent, reactive, and real time algorithms. It has been in planning and prototype stages since 2006, and active development started in 2008. The first public release was made in May 2010. Although the software is still young, it already has both industrial and academic users.

The TLAPS is based on translation of a high level proof language into proof obligations for various backend reasoning systems, which include both off the shelf automated reasoning systems (theorem provers, SMT solvers, etc.) and systems that have been particularly engineered for the TLA language. The backend systems may produce proofs that are then formally checked in Isabelle/TLA+, an axiomatization of TLA in the Isabelle proof system. Some backend systems, such as decision procedures for arithmetic, do not currently produce proofs.

Also in 2010, the TLAPS was integrated into the TLA Toolbox, an integrated development environment (IDE) for many TLA-related tools, including the TLC model checker. The next public release will include support for reasoning about liveness, which will involve implementing a new temporal reasoning framework.

For more information, see the TLAPS home page.

6. New Results

6.1. Proof normalization via atomic flows

Participants: Tom Gundersen, Lutz Strassburger.

In a joint work with Alessio Guglielmi (University of Bath), Tom Gundersen and Lutz Strassburger have given a novel method for normalizing proofs in classical logic, which is based on atomic flows. These are purely graphical devices that abstract away from much of the typical bureaucracy of proofs. We make crucial use of the *path breaker*, an atomic-flow construction that avoids some unpleasant termination problems, and that can be used in any proof system with sufficient symmetry. We also give an original 2-dimensional-diagram exposition of atomic flows, which helps us to connect atomic flows with other known formalisms. This work has been published in LICS 2010 [24].

6.2. Typing lambda-terms with deep inference

Participants: Nicolas Guenot, Lutz Strassburger.

Nicolas Guenot and Lutz Strassburger have been working on extending the Curry-Howard correspondence to the notion of deep inference. The result is a deductive system for intuitionistic logic within deep inference, together with a cut elimination procedure on one side, and a term calculus on the other side. The terms are a variation of lambda terms, and the normalization can simulate beta-reduction, and the rewrite rules are in one-to-one correspondence to cut elimination reduction rules in the deductive system.

6.3. Subexponential logic

Participants: Kaustuv Chaudhuri, Dale Miller.

Subexponential logics are a family of refinements of classical logic that are each parameterized by a collection of *subexponential* connectives arranged in a (pre)order. Although the concept is quite old, Miller and Nigam have shown in 2009 that focused derivations in subexponential logics can adequately capture computations in a programming language consisting of loops, iteration, and loads and stores in *locations*. More generally, Miller has argued that subexponential logics have the potential to be the building blocks of future specification languages for logical and computational systems [25].

Chaudhuri has shown that the classical and intuitionistic dialects of polarized subexponential logics have the same expressive power, in the sense that partial derivations of sequents in one dialect are in one-to-one correspondence with the partial derivations of the encoding of the sequents in the other dialect. This result generalizes several known adequacy theorems between particular subexponential logics, and gives a new adequacy result for encodings of intuitionistic logic in classical linear logic. This result was published in CSL 2010 [18].

6.4. Dynamic polarity assignment

Participant: Kaustuv Chaudhuri.

It is well known that the *polarity* of the atomic propositions in linear logic is ambiguous – each atom may be *assigned* a polarity arbitrarily without destroying the completeness of focused derivations. If the atoms are assigned their polarity statically, that is, before proof search, then it is possible to obtain either forward chaining (also known as program-directed reasoning) or backward chaining (also known as goal-directed reasoning) semantics for logic programs based on the chosen assignments. However, it is not apparently possible to get the intersection of these two strategies with purely static assignments. Chaudhuri has shown that delaying the assignment of atomic polarities until the proof search context can justify a particular assignment can produce such a combined strategy. This dynamic assignment strategy enjoys all the benefits of forward chaining (locality, sharing, concurrency), but terminates and gives the same set of answers as backward chaining on terminating programs. Indeed, we obtain the same semantics as the so-called *magic sets transformation* of logic programs that is based on rewriting the original program and queries, but, since polarity assignment only constrains the proofs and does not alter the programs, we avoid the non-compositional overhead of program transformation. This work was published in LPAR 2010 [19].

6.5. Heterogeneous verification for distributed algorithms

Participants: Kaustuv Chaudhuri, Damien Doligez [INRIA Paris-Rocquencourt, EPI Gallium], Leslie Lamport [Microsoft Research Silicon Valley], Stephan Merz [INRIA Nancy Grand-Est, EPI Veridis].

The TLA+ Proof System (TLAPS), developed at the INRIA-MSR joint centre in association with a number of other INRIA teams, was released publicly. Chaudhuri designed and implemented the *proof manager* component of the tool that interprets high level TLA+ proofs and delegates sub-problems to various backend verifiers. The TLAPS was described in a paper published in IJCAR 2010 [20] and its construction was elaborated and demonstrated further in an invited talk (given by Merz) at ICTAC 2010 [14]. Merz has also demonstrated the TLAPS at a tutorial during IFM 2010.

One of the goals of this project was to give fully formalized correctness proofs of well known distributed algorithms such as Lamport's Bakery Algorithm for distributed mutual exclusion and the Paxos algorithm for distributed consensus. In 2009, the safety part of the correctness of the Bakery algorithm was successfully proved; this proof, initially several thousands of lines long, has been used as a benchmark for further development of the TLAPS and has now shrunk to less than 100 lines after improvements to backend reasoning; in the near future, we will be able to eliminate essentially the entire proof by using a new SMT backend, now in development.

In 2010, significant progress has also been made in proving a number of Paxos algorithms correct, including a novel unpublished version of Paxos that is Byzantine fault tolerant.

6.6. Complexity of λ-terms and intersection types

Participants: Stéphane Lengrand, Alexis Bernadet.

New complexity results have emerged from the study, in the framework of non-idempotent type systems, of how lambda-calculi manage resources.

It is well-known that strongly normalising terms of the λ-calculus can be characterized by a type system using intersections. A term is of type $A \cap B$ if it is both of type A and type B.

Intersections are usually considered idempotent in that $A \cap A = A$. But recent studies have suggested that dropping this property would enrich types with quantitative information.

The first result is a refinement, with quantitative information, of the characterisation of strongly normalising terms : the relevant complexity measure here λ-terms is the worst-case complexity (the length of longest reduction sequences). This quantitative information can now be read directly from typing trees. This result is accepted for publication at the FOSSACS'2011 conference [33].

This unveiled a tight connection between non-idempotent intersections and the way data are erased and duplicated in λ-calculus, suggesting a move to the framework with explicit substitutions. These allow a finer-grained control of erasure and duplication in λ-calculus. The second result is a lift of the first one to the explicit substitution framework, where our analysis of resource management becomes finer-grained and more interesting.

The third result is a new set of semantical tools for λ-calculi, based on filters and orthogonality techniques. These tools simplify previously known proofs of strong normalisation for traditional typing systems like System F. λ-terms typable in System F are strongly normalising and therefore typable with intersection types. In other words, the infinite polymorphism given by System F is in practice always reduced to the finite polymorphism given by intersection types. Our tools explain how.

6.7. A focused sequent calculus interacting with decision procedures

Participants: Stéphane Lengrand, Mahfuza Farooque, Clément Houtmann.

This line of research pertains to the ANR-project PSI (described below).

Inspired by techniques from SAT-modulo-theory, we have designed a focused sequent calculus that can interact with theory-specific decision procedures. Such a procedure is assumed to be able to decide the consistency, with respect to a particular theory, of conjunctions of atomic propositions (with free variables) written in the theory's syntax.

Our sequent calculus organises an interplay between a syntactic equality and a semantical / theory-based equality. It is modular over the theories considered and the decision procedures plugged-in.

We conjecture that the instance of our modular sequent calculus with the procedure known as Congruence Closure is sound and complete with respect to classical logic with Leibniz's equality.

We have a candidate system for making this sequent calculus suitable for proof-search using meta-variables and syntactic unification, whose cohabitation with theories is difficult (and is therefore absent from SAT-modulo-theory solvers).

6.8. Towards a stochastic linear logic for biological computation

Participants: Kaustuv Chaudhuri, Joëlle Despeyroux.

In previous work, Joëlle Despeyroux and Kaustuv Chaudhuri have given an encoding of the synchronous stochastic π-calculus in a hybrid extension of intuitionistic linear logic (called HyLL). Precisely, they have shown that focused partial sequent derivations in the encoding are in bijection with stochastic traces. The modal worlds are used to represent the rates of stochastic interactions, and the connectives of hybrid logic are used to represent the constraints in the stochastic transition rules. These results were presented in an extended report, available from HAL [27].

One of the most successful applications of the stochastic π-calculus has been in representing signal transduction networks in cellular biology. An interesting application of this work would therefore be the direct representations of biological processes in HyLL, the original motivation for this line of investigation. Furthermore, other stochastic systems can, at least in principle, be similarly encoded in HyLL, giving us the linguistic ability to compare and combine systems represented using different stochastic formalisms.

This year, a new definition of the stochastic constraints part of the logic was given. While the new definition is more general than the previous one, it is still not yet satisfactory; More work is needed to provide hyll with the expressiveness of the traditional temporal logic used to reason on biological computations.

7. Other Grants and Activities

7.1. National initiatives

7.1.1. INFER: ANR on the Theory and Application of Deep Inference

Participants: Dale Miller, Lutz Straßburger.

The ANR-project Blanc titled "INFER: Theory and Application of Deep Inference" that is coordinated by Lutz Straßburger has been accepted in September 2006. Besides Parsifal, the teams associated with this effort are represented by François Lamarche (INRIA-Loria) and Michel Parigot (CNRS-PPS). Among the list of theoretical problems there is the fundamental need for a theory of correct identification of proofs, and its corollary, the development of a really general and flexible approach to proof nets. A closely related problem is the extension of the Curry-Howard isomorphism to these new representations. Among the list of more practical problems to be consider is the question of strategy and complexity in proof search, in particular for higher order systems. These questions are intimately related to how proofs themselves are formulated in these systems. Given their common grounding in rewriting theory, the proposal plans to deepen the relationship between deep inference and well established techniques like deduction modulo and unification for quantifiers. The proposal also plans to explore the formulation and use of more "exotic" logical systems, for example, non-commutative logics, that have interesting applications, such as in linguistics and quantum computing.

7.1.2. PSI: ANR on Proof Search control in Interaction with domain-specific methods

Participant: Stéphane Lengrand.

Stephane Lengrand is the scientific leader of the ANR-project Jeunes chercheurs entitled "Proof Search control in Interaction with domain-specific methods", which started in September 2009. Other founding members are among the INRIA project-team "TypiCal" : G. Faure and A. Mahboubi. Since the project started, a Ph.D. student has joined the project's research effort, and another one (Mahfuza Farooque) has been recruited on the project's funds for three years, starting on 1 October 2010. A one-year post-doc position, funded by the project, has been offered to a candidate who should be joining the team in 2011. The project aims at organising the interaction between generic proof-search techniques as developed at Parsifal with decision procedures for specific theories. The project has set up a regular workgroup with experts on SAT-modulo-theory and developers of the Alt-Ergo solver at the INRIA project-team "Proval". Importing techniques from SAT-modulo-theory (or automated reasoning) to a framework where proof objects are being dynamically constructed by proof-search is the desired objective of this collaboration for the PSI-project. This objective converges with theirs in their efforts to extend the capabilities of the Alt-Ergo solver.

7.1.3. CPP: ANR on Confidence, Proofs, and Probabilities

Participants: Ivan Gazeau, Dale Miller.

The ANR Blanc titled "CPP: Confidence, Proofs, and Probabilities" has started 1 October 2009. This grant brings together the following institutions and individuals: LSV (Jean Goubault-Larrecq), CEA LIST (Eric Goubault, Olivier Bouissou, and Sylvie Putot), INRIA Saclay (Catuscia Palamidessi, Dale Miller, and Stephane Gaubert), Supelec L2S (Michel Kieffer and Eric Walter), and Supelec SSE (Gilles Fleury and Daniel Poulton). This project proposes to study the joint use of probabilistic and formal (deterministic) semantics and analysis methods, in a way to improve the applicability and precision of static analysis methods on numerical programs. The specific long-term focus is on control programs, e.g., PID (proportional-integral-derivative) controllers or possibly more sophisticated controllers, which are heavy users of floating-point arithmetic and present challenges of their own. To this end, we shall benefit from case studies and counsel from Hispano-Suiza and Dassault Aviation, who will participate in this project, but preferred to remain formally non-members, for administrative reasons.

7.1.4. Panda: ANR on Parallelism and Distribution Analysis

Participant: Dale Miller.

The ANR Blanc titled "Panda: Parallelism and Distribution Analysis" has started 1 October 2009. This project brings together researchers from INRIA Saclay (Comète and Parsifal), CEA LIST, MeASI as well labs in Paris (LIPN, PPS, LSV, LIP, LAMA), and on the Mediterranean (LIF, IML, Airbus). Scientifically, this proposal deals with the validation of concurrent and distributed programs, which is difficult because the number of its accessible states is too large to be enumerated, and even the number of control points, on which any abstract collecting semantics is based, explodes. This is due to the great number of distinct scheduling of actions in legal executions. This adds up to the important size of the codes, which, because they are less critical, are more often bigger. The objective of this project is to develop theories and tools for tackling this combinatorial explosion, in order to validate concurrent and distributed programs by static analysis, in an efficient manner. Our primary interest lies in multithreaded shared memory systems. But we want to consider a number of other paradigms of computations, encompassing most of the classical ones (message-passing for instance as in POSIX or VXWORKS) as well as more recent ones.

7.2. European initiatives

7.2.1. Structural and Computational Proof Theory

Participants: Kaustuv Chaudhuri, Nicolas Guenot, Dale Miller, Lutz Straßburger.

Structural is an ANR-FWF project submitted to the "Programme Blanc International Annexe projets franco-autrichiens." This project, which involves Parsifal, the University of Paris 7, and the University of Vienna, was accepted in December 2010 and will start in 2011.

7.3. International initiatives

7.3.1. REDO: Redesigning logical syntax

Participants: Nicolas Guenot, Dale Miller, Lutz Straßburger, François Wirion.

The REDO project is an INRIA funded ARC between INRIA Nancy Grand Est, the University of Bath, and INRIA Saclay – Île-de-France. It started in January 2009 and lasts 2 years. Coordinator is Lutz Straßburger.

7.3.2. Slimmer: an INRIA and NSF funded international team

Participants: David Baelde, Andrew Gacek, Dale Miller.

Slimmer stands for *Sophisticated logic implementations for modeling and mechanical reasoning* is an "Equipe Associée" with seed money from INRIA. This project is initially designed to bring together the Parsifal personnel and Gopalan Nadathur's Teyjus team at the University of Minnesota (USA). Separate NSF funding for this effort has also been awards to the University of Minnesota. We are planning to expand the scope of this project to include other French and non-French sites, in particular, Alwen Tiu (Australian National University), Elaine Pimentel (Universidade Federal de Minas Gerais, Brazil) and Brigitte Pientka (McGill University, Canada).

8. Dissemination

8.1. Scientific co-ordination

- Dale Miller has served on the following programme committees:
  1. LPAR-17: 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, Yogyakarta, Indonesia, 11–15 October
  2. IFIP-TCS 2010: International Conference on Theoretical Computer Science, part of the World Computer Congress in Brisbane, Australia, 20–23 September
  3. Workshop on Proof Systems for Program Logics, FLoC 2010, Edinburgh, United Kingdom, 10 July
  4. Workshop on Logics for Agents and Mobility, FLoC 2010, Edinburgh, United Kingdom, 15 July
  5. Workshop on Proof-Search in Type Theories, FLoC 2010, Edinburgh, United Kingdom, 15 July
  6. Workshop on Programming Languages for Mechanized Mathematics Systems (PLMMS), 5 July

  He also serves on the editorial board of the following journals:
  - ACM Transactions on Computational Logic (ToCL). Editor-in-Chief since 1 June 2009; Area editor for Proof Theory since 1999
  - Journal of Automated Reasoning, published by Springer. Member of Editorial Board since 2010
  - Journal of Applied Logic, published by Elsevier. Area editor for "Type Theory for Theorem Proving Systems" since 2003
  - Journal of Logic and Computation, published by Oxford University Press. Associate editor since 1989

  - Journal of Functional and Logic Programming, published by European Association for Programming Languages and Systems (EAPLS). Permanent member of the Editorial Board. 1996–2010

  His other professional duties include:

  - Member of the “comité d’enseignement et recherche du Département d’Informatique de l’École Polytechnique (DIX)”, from October 2010
  - Member of the “comité de sélection sur le poste 27PR90 ‘Logique et vérification”’ at Rennes 1, May 2010.
  - Member of the “comité de programmes”, Digiteo, during 2010

- Stéphane Lengrand has served on the following programme committees:
  1. LICS-25: Twenty-Fifth Annual IEEE Symposium on Logic In Computer Science (LICS 2010), Edinburgh, United Kingdom, 11–14 July
  2. Workshop on Proof-Search in Type Theories, FLoC 2010, Edinburgh, United Kingdom, 15 July
  3. Workshop on Classical Logic and Computation, colocated with MFCS’2010 and CSL’2010, Brno, Czech Republic, 21–22 August
- Lutz Straßburger organized the third REDO meeting in Bath, UK, 14–16 September
- Kaustuv Chaudhuri serves on the “commission développement technologique (CDT)” for INRIA Saclay – Île-de-France since February 2010

Parsifal has also hosted the following short term scientific visitors:

- Murdoch J. Gabbay, Lecturer, Heriott-Watt University, Edinburgh, United Kingdom, 2–5 January & 4–9 October & 27-31 October & 11–18 November & 22–28 December,
- Dan R. Ghica, Senior Lecturer, University of Birmingham, United Kingdom, 14–25 November
- Chuck Liang, Associate Professor, Hofstra University, NY, USA, 22 June – 2 July & 9–18 December
- Gopalan Nadathur, Professor, University of Minnesota, MN, USA, 17–30 May and 3–9 December 2010.

8.2. Teaching

- 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.
- Dale Miller taught 10 hours in a graduate course at the University of Milan for one week in March 2010.
- Lutz Straßburger taught a course *Introduction to Proof Theory* at ESSLLI 2010 in Copenhagen in August 2010.

9. Bibliography

Major publications by the team in recent years

[1] K. CHAUDHURI, F. PFENNING, G. PRICE. *A Logical Characterization of Forward and Backward Chaining in the Inverse Method*, in "J. of Automated Reasoning", March 2008, vol. 40, n^{o} 2-3, p. 133–177 [*DOI :* s10817-007-9091-0], http://www.springerlink.com/content/b335r17728862pm2/fulltext.pdf.

[2] R. DYCKHOFF, S. LENGRAND. *Call-by-Value λ-calculus and LJQ*, in "Journal of Logic and Computation", 2007, vol. 17, n^o 6, p. 1109–1134.

[3] 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.

[4] F. LAMARCHE, L. STRASSBURGER. *From Proof Nets to the Free *-Autonomous Category*, in "Logical Methods in Computer Science", 2006, vol. 2, n^{O} 4:3, p. 1–44, http://arxiv.org/pdf/cs.LO/0605054.

[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] 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.

[8] R. MCDOWELL, D. MILLER, C. PALAMIDESSI. *Encoding transition systems in sequent calculus*, in "Theoretical Computer Science", 2003, vol. 294, n^{O} 3, p. 411–437, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tcs97.pdf.

[9] D. MILLER, A. TIU. *A proof theory for generic judgments*, in "ACM Trans. on Computational Logic", October 2005, vol. 6, n^{O} 4, p. 749–783, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tocl-nabla.pdf.

[10] 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.

Publications of the year

Articles in International Peer-Reviewed Journal

[11] O. DELANDE, D. MILLER, A. SAURIN. *Proof and refutation in MALL as a game*, in "Annals of Pure and Applied Logic", February 2010, vol. 161, n^{O} 5, p. 654–672 [*DOI :* 10.1016/J.APAL.2009.07.017], http://hal.inria.fr/hal-00527922.

[12] V. NIGAM, D. MILLER. *A framework for proof systems*, in "Journal of Automated Reasoning", 2010, vol. 45, n^{O} 2, p. 157–188 [*DOI :* 10.1007/S10817-010-9182-1], http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/nigam-ijcar.pdf.

[13] A. TIU, D. MILLER. *Proof Search Specifications of Bisimulation and Modal Logics for the π-calculus*, in "ACM Transactions on Computattional Logic", 2010, vol. 11, n^{O} 2 [*DOI :* 10.1145/1656242.1656248], http://arxiv.org/abs/0805.2785.

Invited Conferences

[14] K. Chaudhuri, D. Doligez, S. Merz, L. Lamport. *The TLA+ Proof System: Building a Heterogeneous Verification Platform*, in "Proceedings of the 7th International Colloquium on Theoretical Aspects of Computing (ICTAC)", Natal, Brazil, A. Cavalcanti, D. Déharbe, M.-C. Gaudel, J. Woodcock (editors), LNCS, Springer, September 2010, vol. 6256, 44 [*DOI :* 10.1007/978-3-642-14808-8_3], http://hal.inria.fr/inria-00521886.

[15] D. Miller. *Reasoning about Computations Using Two-Levels of Logic*, in "Proceedings of the 8th Asian Symposium on Programming Languages and Systems (APLAS 2010)", K. Ueda (editor), LNCS, 2010, n^o 6461, p. 34–46 [*DOI :* 10.1007/978-3-642-17164-2_4], http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/aplas10.pdf.

International Peer-Reviewed Conference/Proceedings

[16] 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.

[17] P. Bruscoli, A. Guglielmi, T. Gundersen, M. Parigot. *A Quasipolynomial Cut-Elimination Procedure in Deep Inference via Atomic Flows and Threshold Formulae*, in "Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2009)", Dakar, Senegal, E. Clarke, A. Voronkov (editors), LNAI, Springer, April 2010, vol. 6355, http://cs.bath.ac.uk/ag/p/QPNDI.pdf.

[18] K. Chaudhuri. *Classical and Intuitionistic Subexponential Logics are Equally Expressive*, in "19th Annual EACSL Conference on Computer Science Logic (CSL 2010)", Brno, Czech Republic, A. Dawar, H. Veith (editors), LNCS, Springer, August 2010, vol. 6247, p. 185–199 [*DOI :* 10.1007/978-3-642-15205-4_17], http://hal.inria.fr/inria-00534865.

[19] K. Chaudhuri. *Magically Constraining the Inverse Method Using Dynamic Polarity Assignment*, in "Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2010)", Yogyakarta, Indonesia, C. Fermüller, A. Voronkov (editors), LNCS, Springer, October 2010, vol. 6397, p. 202–216 [*DOI :* 10.1007/978-3-642-16242-8_15], http://hal.inria.fr/inria-00535948.

[20] 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.

[21] A. Gacek. *Relating Nominal and Higher-order Abstract Syntax Specifications*, in "Proceedings of the 2010 Symposium on Principles and Practice of Declarative Programming", ACM, July 2010, p. 177–186 [*DOI :* 10.1145/1836089.1836112], http://arxiv.org/pdf/1003.5447.pdf.

[22] N. Guenot. *Focused Proof Search for Linear Logic in the Calculus of Structures*, in "Technical Communications of the 26th International Conference on Logic Programming (ICLP 2010)", Edinburgh, United Kingdom, M. V. Hermenegildo, T. Schaub (editors), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, July 2010, vol. 7, p. 84–93 [*DOI :* 10.4230/LIPIcs.ICLP.2010.84], http://www.lix.polytechnique.fr/~nguenot/pub/iclp10.pdf.

[23] A. Guglielmi, T. Gundersen, M. Parigot. *A Proof Calculus Which Reduces Syntactic Bureaucracy*, in "Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010)", Edinburgh, United Kingdom, C. Lynch (editor), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, July 2010, vol. 6, p. 135–150 [*DOI :* 10.4230/LIPIcs.RTA.2010.135], http://drops.dagstuhl.de/opus/volltexte/2010/2649.

[24] 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.

[25] D. Miller. *Finding unity in computational logic*, in "ACM-BCS-Visions of Computer Science", University of Edinburgh, Edinburgh, United Kingdom, British Informatics Society, April 2010, p. 1–13, http://www.bcs.org/server.php?show=conMediaFile.15102.

[26] L. Strassburger. *What Is the Problem with Proof Nets for Classical Logic?*, in "Programs, Proofs, Processes, 6th Conference on Computability in Europe (CiE 2010)", Ponta Delgada, Azores, Portugal, F. Ferreira, B. Löwe, E. Mayordomo, L. M. Gomes (editors), LNCS, Springer, June 2010, vol. 6158, p. 406–416 [*DOI :* 10.1007/978-3-642-13962-8_45], http://www.lix.polytechnique.fr/~lutz/papers/CiE10.pdf.

Research Reports

[27] K. Chaudhuri, J. Despeyroux. *A Hybrid Linear Logic for Constrained Transition Systems with Applications to Molecular Biology*, INRIA, 2010, http://hal.inria.fr/inria-00402942/.

References in notes

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

[29] 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.

[30] 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.

[31] D. Baelde, A. Gacek, D. Miller, G. Nadathur, A. Tiu. *A User Guide to Bedwyr*, November 2006, http://gforge.inria.fr/docman/view.php/367/705/userguide.pdf.

[32] N. D. Belnap, Jr.. *Display Logic*, in "Journal of Philosophical Logic", 1982, vol. 11, p. 375–417.

[33] A. Bernadet, S. Lengrand. *Complexity of strongly normalising λ-terms via non-idempotent intersection types*, in "14th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2011)", Saarbrücken, Germany, M. Hofmann (editor), LNCS, Springer, March 2011, To appear.

[34] P. BRAUNER, C. HOUTMANN, C. KIRCHNER. *Principles of Superdeduction*, in "22nd IEEE Symposium on Logic in Computer Science (LICS 2007)", Wroclaw, Poland, IEEE Computer Society, July 2007, p. 41–50 [*DOI :* 10.1109/LICS.2007.37], http://hal.inria.fr/inria-00133557.

[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. 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.

[37] J. CHENEY, C. URBAN. *Alpha-Prolog: A Logic Programming Language with Names, Binding, and Alpha-Equivalence*, in "Logic Programming, 20th International Conference", B. DEMOEN, V. LIFSCHITZ (editors), LNCS, Springer, 2004, vol. 3132, p. 269–283.

[38] G. DOWEK, T. HARDIN, C. KIRCHNER. *Theorem Proving Modulo*, INRIA, April 1998, n^{o} 3400.

[39] 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.

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

[41] J.-Y. GIRARD. *Proof Theory and Logical Complexity, Volume I*, Studies in Proof Theory, Bibliopolis, edizioni di filosofia e scienze, 1987, vol. 1.

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

[43] 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.

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

[45] F. LAMARCHE, L. STRASSBURGER. *Constructing free Boolean categories*, in "Proceedings of the Twentieth Annual IEEE Symposium on Logic in Computer Science (LICS'05)", 2005, p. 209–218.

[46] F. LAMARCHE, L. STRASSBURGER. *From Proof Nets to the Free *-Autonomous Category*, in "Logical Methods in Computer Science", 2006, vol. 2, n^{o} 4:3, p. 1–44, http://arxiv.org/pdf/cs.LO/0605054.

[47] STÉPHANE. LENGRAND, R. DYCKHOFF, J. MCKINNA. *A sequent calculus for Type Theory*, in "Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic (CSL'06)", Z. ESIK (editor), Lecture Notes in Computer Science, Springer-Verlag, September 2006, vol. 4207, p. 441–455.

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

[49] 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.

[50] 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.

[51] 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, p. 174–181, http://www.cs.cmu.edu/~seanmcl/papers/McLaughlin-LPAR-2008.pdf.

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

[53] 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.

[54] 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.

[55] V. NIGAM, D. MILLER. *A framework for proof systems*, March 2009, Extended version of IJCAR08 paper. Submitted.

[56] V. NIGAM, D. MILLER. *Algorithmic specifications in linear logic with subexponentials*, in "ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP)", 2009, p. 129–140, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/ppdp09.pdf.

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

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

[59] A. M. PITTS. *Nominal Logic, A First Order Theory of Names and Binding*, in "Information and Computation", 2003, vol. 186, n^{o} 2, p. 165–193.

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

[61] L. STRASSBURGER, F. LAMARCHE. *On Proof Nets for Multiplicative Linear Logic with Units*, in "Computer Science Logic, CSL 2004", J. MARCINKOWSKI, A. TARLECKI (editors), LNCS, Springer-Verlag, 2004, vol. 3210, p. 145–159.

[62] L. STRASSBURGER. *What could a Boolean category be?*, in "Classical Logic and Computation 2006 (Satellite Workshop of ICALP'06)", S. VAN BAKEL (editor), 2006, http://www.lix.polytechnique.fr/~lutz/papers/medial-kurz.pdf.

[63] L. STRASSBURGER. *Some Observations on the Proof Theory of Second Order Propositional Multiplicative Linear Logic*, in "Typed Lambda Calculi and Applications, TLCA'09", P.-L. CURIEN (editor), LNCS, Springer, 2009, vol. 5608, p. 309–324 [*DOI :* 10.1007/978-3-642-02273-9_23], http://www.lix.polytechnique.fr/~lutz/papers/ObsPT-MLL2-finalforTLCA09.pdf.

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