INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Parsifal

Preuves Automatiques et Raisonnement sur des SpécIFicAtions Logiques

Saclay - Île-de-France

THEME SYM

Activity Report

2008

Table of contents

1. Team

Research Scientist

Joëlle Despeyroux [CR INRIA-Sophia]
Stéphane Lengrand [CR CNRS]
Dale Miller [DR INRIA-Saclay, Team Leader]
Lutz Straßburger [CR INRIA-Saclay]

PhD Student

David Baelde [Allocataire École Normale Supérieure, Lyon, since 1 September 2005]
Olivier Delande [Bourse Monge École Polytechnique, since 1 October 2006]
Nicolas Guenot [Allocataire Ministère de la Recherche, since 1 October 2008]
Vivek Nigam [Allocataire Mobius, since 1 October 2006]
Alexis Saurin [Allocataire École Normale Supérieure, Paris, since 1 October 2004]
Alexandre Viel [Allocataire École Normale Supérieure, Paris, since 1 October 2008]
François Wirion [Bourse INRIA, since 1 December 2008]

Post-Doctoral Fellow

Matteo Capelletti [Post Doctorant INRIA, 1 Dec 2007 - 30 November 2008]
Stefan Hetzl [Post Doctorant INRIA, 1 Sep 2008 - 30 Aug 2009]
Laurent Méhats [Post Doctorant INRIA, 1 Sep 2007 - 31 Aug 2008]

Visiting Scientist

Chuck Liang [Associate Professor, Hofstra University, NY, USA, 21 May - 18 Jun 2008]
Elaine Pimentel [Associate Professor, Universidade Federal de Minas Gerais, Brazil, 16 Oct - 15 Nov 2008]

Administrative Assistant

Isabelle Biercewicz

2. Overall Objectives

2.1. Main Themes

The aim of the Parsifal team is to develop and exploit the theories of proofs and types to support the specification and verification of computer systems. To achieve these goals, the team works on several level.

- The team has expertise in *proof theory* and *type theory* and conducts basic research in these fields: in particular, the team is developing results that help with the automation of deduction and with the formal manipulation and communication of proofs.
- Based on experience with computational systems and theoretical results, the team *designs* new logical principles, new proof systems, and new theorem proving environments.
- Some of these new designs are appropriate for *implementation* and the team works at developing prototype systems to help validate basic research results.
- By using the implemented tools, the team can develop examples of specifications and verification to test the success of the design and to help suggest new logical and proof theoretic principles that need to be developed in order to improve ones ability to specify and verify.

The foundational work of the team focuses on the proof theory of classical, intuitionistic, and linear logics making use, primarily, of sequent calculus and deep inference formalisms. A major challenge for the team is the reasoning about computational specifications that are written in a relational style: this challenge is being addressed with the introduction of some new approaches to dealing with induction, co-induction, and generic judgments. Another important challenge for the team is the development of normal forms of deduction: such normal forms can be used to greatly enhance the automation of search (one only needs to search for normal forms) and for communicating proofs (and proof certificates) for validation.

The principle application areas of concern for the team currently are in functional programming (e.g., λ-calculus), concurrent computation (e.g., π-calculus), interactive computations (e.g., games), and biological systems.

2.2. Highlights of the year

The team has developed the theme of *focused proof search* to broaden its proof theoretical and game theoretic basis as well as to support a number of computer science applications. Additional advances were made on an approach to reasoning with bindings in syntactic expressions. Together with a team at the University of Minneapolis, we have implemented two prototype provers, Abella and Taci, which have been used to explore the effectiveness of both these theoretical threads.

3. Scientific Foundations

3.1. General Overview

Keywords: *proof normalization*, *proof search*.

In the specification of computational systems, logics are generally used in one of two approaches. In the *computation-as-model* approach, computations are encoded as mathematical structures, containing such items as nodes, transitions, and state. Logic is used in an external sense to make statements *about* those structures. That is, 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. This use of logic to represent and reason about computation is probably the oldest and most broadly successful use of logic in computation.

The *computation-as-deduction* approach, uses directly pieces of logic's syntax (such as formulas, terms, types, and proofs) as elements of the specified computation. In this much more rarefied setting, there are two rather different approaches to how computation is modeled.

The *proof normalization* approach views the state of a computation as a proof term and the process of computing as normalization (know variously as β-reduction or cut-elimination). Functional programming can be explained using proof-normalization as its theoretical basis [38] and has been used to justify the design of new functional programming languages [20].

The *proof search* approach views the state of a computation as a sequent (a structured collection of formulas) and the process of computing as the process of searching for a proof of a sequent: the changes that take place in sequents capture the dynamics of computation. Logic programming can be explained using proof search as its theoretical basis [43] and has been used to justify the design of new logic programming languages [42].

The divisions proposed above are informal and suggestive: such a classification is helpful in pointing out different sets of concerns represented by these two broad approaches (reductions, confluence, etc, versus unification, backtracking search, etc). Of course, a real advance in computation logic might allow us merge or reorganize this classification.

Although type theory has been essentially designed to fill the gap between these two kinds of approaches, it appears that each system implementing type theory up to now only follows one of the approaches. For example, the Coq system implementing the Calculus of Inductive Constructions (CIC) uses proof normalization while the Twelf system [50], implementing the Edinburgh Logical Framework (LF, a sub-system of CIC), follows the proof search approach (normalization appears in LF, but it is much weaker than in, say, CIC).

The Parsifal team works on both the proof normalization and proof search approaches to the specification of computation.

3.2. Reasoning about logic specifications

Keywords: *LINC*, *definitions*, *fixed points*, *generic judgments*, *higher-order abstract syntax*, *lambda-tree syntax*.

Once a computational system (e.g., a programming language, a specification language, a type system) is given a logic (relational) specifications, how do we reason about the formal properties of such specifications? New results in proof theory are being developed to help answer this question.

The traditional architecture for systems designed to help reasoning about the formal correctness of specification and programming languages can generally be characterized at a high-level as follows: **First: Implement mathematics.** This often involves choosing between a classical or constructive (intuitionistic) foundation, as well as a choosing abstraction mechanism (eg, sets or functions). The Coq and NuPRL systems, for example, have chosen intuitionistically typed λ-calculus for their approach to the formalization of mathematics. Systems such as HOL [32] use classical higher-order logic while systems such as Isabelle/ZF [49] use classical logic. **Second: Reduce programming correctness problems to mathematics.** Thus, data structures, states, stacks, heaps, invariants, etc, all are represented as various kinds of mathematical objects. One then reasons directly on these objects using standard mathematical techniques (induction, primitive recursion, fixed points, well-founded orders, etc).

Such an approach to formal methods is, of course, powerful and successful. There is, however, growing evidence that many of the proof search specifications that rely on such intensional aspects of logic as bindings and resource management (as in linear logic) are not served well by encoding them into the traditional data structures found in such systems. In particular, the resulting encoding can often be complicated enough that the *essential logical character* of a problem is obfuscated.

Despeyroux, Pfenning, Leleu, and Schürmann proposed two different type theories [2], [1] based on modal logic in which expressions (possibly with binding) live in the functional space $A \rightarrow B$ while general functions (for case and iteration reasoning) live in the full functional space $\Box A \rightarrow B$. These works give a possible answer to the problem of extending the Edinburgh Logical Framework, well suited for describing expressions with binding, with recursion and induction principles internalized in the logic (as done in the Calculus of Inductive Constructions). However, extending these systems to dependent types seems to be difficult (see [28] where an initial attempt was given).

The LINC logic of [57] appears to be a good meta-logical setting for proving theorems about such logical specifications The three key ingredients of LINC can be described as follows.

First, LINC is an intuitionistic logic for which provability is described similarly to Gentzen's LJ calculus [29]. Quantification at higher-order types (but not predicate types) is allowed and terms are simply typed λ-terms over $\beta\eta$-equivalence. This core logic provides support for λ-*tree syntax*, a particular approach to *higher-order abstract syntax*. Considering a classical logic extension of LINC is also of some interest, as is an extension allowing for quantification at predicate type.

Second, LINC incorporates the proof-theoretical notion of *definition* (also called *fixed points*), a simple and elegant device for extending a logic with the if-and-only-if closure of a logic specification and for supporting inductive and co-inductive reasoning over such specifications. This notion of definition was developed by Hallnäs and Schroeder-Heister [35] and, independently, by Girard [31]. Later McDowell, Miller, and Tiu have made substantial extensions to our understanding of this concept [40], [7], [45]. Tiu and Momigliano [46], [57] have also shown how to modify the notion of definition to support induction and co-induction in the sequent calculus.

Third, LINC contains a new (third) logical quantifier ∇ (nabla). After several attempts to reason about logic specifications without using this new quantifier [39], [41], [7], it became clear that when the object-logic supports λ-tree syntax, the *generic judgment* [45], [9] and its associated quantifier could provide a strong and declarative role in reasoning. This new quantifier is able to help capture internal (intensional) reasons for judgment to hold generically instead of the universal judgment that holds for external (extensional) reasons. Another important observation to make about ∇ is that if given a logic specification that is essentially a collection of Horn clauses (that is, there is no uses of negation in the specification), there is no distinctions to be made between $\forall$ or ∇ in the premise (body) of semantic definitions. In the presence of negations and implications, a difference between these two quantifiers does arise [9].

3.3. Focused proof systems

Keywords: *focused proof systems*.

There is a great deal of non-determinism that is present in the search for proofs (in the sense of automated deduction). The non-determinism involved with generating lemmas is one extreme: when attempting to prove one formula it is possible to generate a potential lemma and then attempt to prove it and to use it to prove the original formula. In general, there are no clues as to what is a useful lemma to construct. The famous "cut-elimination" theorem say that it is possible to prove theorems without using lemmas (that is, by restricting to *cut-free* proofs). Of course, cut-free proofs are not appropriate for all domains of computation logic since they can be vastly larger than proofs containing cuts. Even when restricting to cut-free proofs make sense (as in logic programming, model checking, and some areas of automated reasoning), the construction of cut-free proofs still contains a great deal of non-determinism.

Structuring the non-deterministic choices within the search for cut-free proofs has received increasing attention in recent years with the development of *focusing proofs systems*. In such proof systems, there is a clear separation between non-deterministic choices for which no backtracking is required ("don't care non-determinism") and choices were backtracking may be required ("don't know non-determinism"). Furthermore, when a backtrackable choice is required, that choice actually extends over a series of inference rules, representing a "focus" during the construction of a proof. One focusing-style proof systems was developed within the early work of providing a proof-theoretic foundations for logic programming via *uniform proofs* [43]. The first comprehensive analysis of focusing proofs was done in linear logic by Andreoli [22]. There it was shown that proofs are constructed in two alternating phases: a *negative phase* in which don't-care-non-determinism is done and a *positive phase* in which a focused sequence of don't-know-non-determinism choices is applied.

Since a great deal of automated deduction (in the sense of logic programming, type inference, and model checking) is done in intuitionistic and classical logic, there is a strong need to have comprehensive focusing results for these logics as well. In linear logic, the assignment of inference rules to the positive and negative phases is canonical (only the treatment of atomic formulas is left as a non-canonical choice). Within intuitionistic and classical logic, a number of inference rules do not have canonical treatments. Instead, several focusing-style proof systems have been developed one-by-one for these logics. A general scheme for putting all of these choices together has recently been developed within the team and will be described below.

3.4. Proof structure and proof certificates

Keywords: *proof certificates*.

There has been a good deal of concern in the proof theory literature on the nature of proofs as objects. An example of such a concern is the question as to whether or not two proofs should be considered equal. Such considerations were largely of interest to philosophers and logicians. Computer scientists started to get involved with the structure of proofs more generally in essentially two ways. The first is the extraction of algorithms from constructive proofs. The second is the use of proof-like objects to help make theorem proving systems more sophisticated (proofs could be stored, edited, and replayed, for example).

It was not until the development of the topic of *proof carrying code (PCC)* that computer scientists from outside the theorem proving discipline took a particular interesting in having proofs as actual data structure within computations. In the PCC setting, proofs of safety properties need to be communicated from one host to another: the existence of the proof means that one does not need to trust the correctness of a piece of software (for example, that it is not a virus). Given this need to produce, communicate, and check proofs, the actual structure and nature of proofs-as-objects becomes increasingly important.

Often the term *proof certificate* (or just *certificate*) is used to refer to some data structure that can be used to communicate a proof so that it can be checked. A number of proposals have been made for possible structure of such certificates: for examples, proof scripts in theorem provers such as Coq are frequently used. Other notions include oracle [48] and fixed points [21].

The earliest papers on PCC made use of logic programming systems Twelf [47] and λProlog [23]. It seems that the setting of logic programming is a natural one for exploring the structure of proofs and the trade-offs between proof size and the need for run-time proof search. For example, there is a general trade-off between the size of a proof object and the amount of search one must do to verify that an object does, in fact, describe a proof. Exploring such trade-off should be easy and natural in the proof search setting where such search is automated. In particular, focused proof systems should be a large component of such an analysis.

3.5. Deep Inference and Categorical Axiomatizations

Keywords: *category theory*, *deep inference*.

Deep inference [33], [34] 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 counterintuitive 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 [4], and analyzed the category theory behind them in [36]. In [10] and [54], 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.

3.6. Proof Nets and Combinatorial Characterization of Proofs

Keywords: *cographs*, *correctness criteria*, *proof nets*, *relation webs*.

Proof nets are abstract (graph-like) presentations of proofs such that all "trivial rule permutations" are quotiented away. More generally, we investigate combinatoric objects and correctness criteria for studying proofs independently from syntax. 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 [30] for linear logic but also in Robinson’s proof nets [52] 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. Recent work by Lamarche and Straßburger [53] and [5] 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.
- 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). We hope that eventually deductive proof nets will provide a “bureaucracy-free” formalism for proof search.

3.7. A logic for systems biology

Keywords: *biology*, *hybrid logic*, *linear logic*, *stochastic pi-calculus*, *stochastic transition systems*, *systems biology*, *temporal logic*.

Systems in molecular biology, such as those for regulatory gene networks or protein-protein interactions, can be seen as state transition systems that have an additional notion of *rate* of change. Methods for specifying such systems is an active research area. However, to our knowledge, no logic (more powerful than the boolean logic) have been proposed so far to both specify and reason about these systems.

One current and prominent method uses process calculi, such as the stochastic π-calculus, that has a built in notion of rate [26]. Process calculi, however, have the deficiency that reasoning about the specifications is external to the specifications themselves, usually depending on simulations and trace analysis.

Kaustuv Chaudhuri and Joëlle Despeyroux are considering the problem of giving a *logical* instead of a *process-based* treatment both to specify and to reason about biological systems in a uniform linguistic framework. The logic they have proposed, called HyLL, is an extension of (intuitionistic) linear logic with a modal situated truth that may be reified by means of the $\downarrow$ operator from *hybrid logic*. A variety of semantic interpretation can be given to this logic, including the rates and the delay of formation.

The expressiveness of the logic has been demonstrated on small examples and first meta-theoretical properties of the logic have been proven. Considerable work needs to be done before this proposal succeeds as a natural logical framework for systems biology. Remaining work mainly includes the description of larger examples (requiring more specifications of usual biological notions), and automating reasoning about the specifications. It also includes further studies of the meta-theoretical properties of the logic, and of course eventual extensions of the logic (for example to get branching semantics).

4. Application Domains

4.1. Model checking operational semantics

Keywords: *model checking*, *operational semantics*, *process calculus*, *software correctness*.

When operational semantics is presented as inference rules, it can often be encoded naturally as a logic program, which means that it is usually easy to animate such semantic specifications in direct and natural ways. Given the natural duality between finite success and finite failure (given a proof theoretic foundations in papers such as [8] and [44]) it is also possible to describe model checking systems from a proof theoretic setting.

One application area for this work is, thus, the development of model checking software that can work on linguistic expressions that may contain bound variables. Specific applications could be towards checking bisimulation of λ-calculus and π-calculus expressions.

More about a prototype model checker in this area is described below.

4.2. Mechanized metatheory

Keywords: *mechanized metatheory*, *reasoning about software*.

There has been increasing interest in the international community with the use of formal methods to provide proofs of properties of programs and entire programming languages. The example of proof carrying code is one such example. Two more examples for which the team's efforts should have important applications are the following two challenges.

Tony Hoare's Grand Challenge titled "Verified Software: Theories, Tools, Experiments" has as a goal the construction of "verifying compilers" to support a vision of 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 [24] 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." There is clearly a strong overlap in the goals of this challenge and those of part of the Parsifal team.

4.3. Biological systems

Keywords: *biology*, *systems biology*.

When one looks at systems of biochemical reactions in molecular biology, such as gene-protein and protein-protein interaction systems, one observes two basic phenomena: state change (where, for example, two or more molecules interact to form other molecules) and delay. Each of the state changes has an associated delay before the state change is observed, or, more precisely, a probability distribution over possible delays: the rate of the change. A system of biochemical reactions can therefore be seen as a stochastic computation.

The HyLL logic proposed by Kaustuv Chaudhuri and Joëlle Despeyroux is a first attempt at providing a logical framework for both specifying and reasoning about such computations.

5. Software

5.1. Bedwyr

Participants: David Baelde, Andrew Gacek, Dale Miller.

In order to provide some practical validation of the formal results mentioned above regarding the logic LINC and the quantifier ∇, we picked a small but expressive subset of that logic for implementation. While that subset did not involve the proof rules for induction and co-induction (which are difficult to automate) the subset did allow for model-checking style computation. During 2006 and 2007, the Parsifal team, with contributions from our close colleagues at the University of Minnesota and the Australian National University, designed and implemented the Bedwyr system for doing proof search in that fragment of LINC. This system is organized as an open source project and is hosted on INRIA's GForge server. It has been described in the conference papers [56] and [25]. This systems, which is implemented in OCaml, has been download about 200 times since it was first released.

Bedwyr is a generalization of logic programming that allows model checking directly on syntactic expressions possibly containing bindings. This system, written in OCaml, is a direct implementation of two recent advances in the theory of proof search.

1. It is possible to capture both finite success and finite failure in a sequent calculus. Proof search in such a proof system can capture both may and must behavior in operational semantics.
2. Higher-order abstract syntax is directly supported using term-level λ-binders, the ∇-quantifier, higher-order pattern unification, and explicit substitutions. These features allow reasoning directly on expressions containing bound variables.

Bedwyr has served well to validate the underlying theoretical considerations while at the same time providing a useful tool for exploring some applications. The distributed system comes with several example applications, including the finite π-calculus (operational semantics, bisimulation, trace analysis, and modal logics), the spi-calculus (operational semantics), value-passing CCS, the λ-calculus, winning strategies for games, and various other model checking problems.

5.2. Taci

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

During the summer of 2007, Baelde (LIX PhD student) and visiting intern Zachery Snow (PhD student from the University of Minnesota) built a prototype theorem prover, called *Taci*, that we are currently using "in-house" to experiment in a number of large examples and a few different logics. During the summer of 2008, Snow visited the team again: he and Baelde and Viel developed Taci in a new direction: they took one logic (an intuitionistic logic of fixed points) and developed an automated prover for it based on all of the recent focused proof search techniques that the team and others have been developing in recent years. This new prototype is now a focus point for further implementations.

6. New Results

6.1. Formalizing operational semantic specifications in logic

Participant: Dale Miller.

An important application area for some of the proof theory results of the team is in reasoning about the operational semantics of programs and specifications. Miller provided a survey of various approaches to encoding such semantic specifications in the survey article [13]. Miller and the former team member Alwen Tiu recent submitted a paper [55] that provides, in their opinion, the definitive treatment of the (finite) π-calculus within the λ-tree syntax approach to encoding. In that paper, the syntax and transition semantics for the π-calculus are presented as simple logic programs. Using those specifications, they developed the notions of open and late bisimulation. All of these specifications were declarative and without side conditions. A result of maintaining a high-level of declarativeness in these specifications, it was possible to describe a novel characterization of the differences between open and late bisimulations. Full adequacy results were provided, there by showing a precise match between the "standard" techniques for the specification of the π-calculus and the more abstract, proof-theoretically inspired approach based on λ-trees syntax.

6.2. New insights into ∇-quantification

Participants: David Baelde, Dale Miller.

The team has been actively extending the scope of effectiveness ∇-quantifications. As Tiu and Miller have shown in [55], the ∇ quantifier (developed in previous years within the team) provides a completely satisfactory treatment of binding structures in the *finite* π-calculus. Moving this quantifier to treat infinite behaviors via induction and co-induction, required new advances in the underlying proof theory of ∇-quantification.

The team has explored two different approaches to this problem. David Baelde [11], [14] has developed a minimalist generalization of previous work by Miller and Tiu: he has found what seems to be the simplest extension that earlier work that allows ∇ to interact properly with fixed points and their inference rules (namely, induction and co-induction). His logical approach allows for a rather careful and rigid understanding of scope in the treatment of the meta-theory of logics and computational specifications.

Another angle has been developed as a result of our close international collaborations. Alwen Tiu, now at the Australian National University, has developed a logic, called LG^{ω} which extends the earlier, "minimal" approach by introducing the structural rules of strengthening and exchange into the context of generic variables. As a result, the behavior of bindings becomes much more like the behavior of names more generally, while still maintaining much of the status as being binders. In combination with our close colleagues at the University of Minnesota, we have extended this work to include a new definitional principle, called *nabla-in-the-head*, that strengthens our ability to declaratively describe the structure of contexts and proof invariants. This new definitional principle was first presented in [17] and examples of it were presented in [18]. Our colleague, Andrew Gacek (a PhD student at the University of Minnesota and former intern with Parsifal) has also built the Abella proof editor that allows for the direct implementation of this new definitional principle. His system is in distribution and has been used by a number of people to develop examples in this logic.

6.3. Foundational aspects of focusing proof systems

Participants: David Baelde, Dale Miller, Alexis Saurin.

Since focusing proof systems seem to be behind much of our computational logic framework, the team has spent some energies developing further some foundational aspects of this approach to proof systems.

Chuck Liang and Miller have recently finished the paper [6] in which a comprehensive approach to focusing in intuitionistic and classical logic was developed.

Given the team's ambitious to automate logics that require induction and co-induction, we have also looked in detail at the proof theory of fixed points. In particular, David Baelde's recent PhD thesis [11] contains a number of important, foundational theorems regarding focusing and fixed points. In particular, he has examined the logic MALL (multiplicative and additive linear logic). To strengthen this decidable logic into a more general logic, Girard added the exponentials, which allowed for modeling unbounded ("infinite") behavior. Baelde considers, however, the addition of fixed points instead and he has developed the proof theory of the resulting logic. We see this logic as being behind much of the work that the team will be doing in the coming few years.

Alexis Saurin's recent PhD [12] also contains a wealth of new material concerning focused proof system. In particular, he provides a new and modular approach to proving the completeness of focused proof systems as well as develops the theme of multifocusing.

A particular outcome of our work on focused proof search is the use of *maximally multifocused proofs* to help provide sequent calculus proofs a canonicity. In particular, Chaudhuri (a former Parsifal post doc), Miller, and Saurin have shown in [15] that it is possible to show that maximally multifocused sequent proofs can be placed in one-to-one correspondence with more traditional proof net structures for subsets of MALL.

6.4. A neutral approach to proof and refutation in MALL

Participants: Olivier Delande, Dale Miller, Alexis Saurin.

Given the team's previous efforts at building automated deduction systems that viewed finite success and finite failure as duals within a proof theory setting, we were intrigued to see if that duality could be extended beyond the simple, Horn clause setting. This past year, Olivier Delande and Miller described in the paper [16] a way to extend that notion of proving and refuting to the richer setting of MALL. They showed how it is possible in that logic to view moves in a game as contributing simultaneously to both a proof and a refutation of a given formula. Of course, only one of the players of such a game can be a winner. Delande, Miller, and Saurin have completed a comprehensive approach to this style of game in [27].

6.5. Meta-level focusing and object-level proof system

Participants: Vivek Nigam, Dale Miller.

It is well known how to use an intuitionistic meta-logic to specify natural deduction systems. It is also possible to use linear logic as a meta-logic for the specification of a variety of sequent calculus proof systems [51]. Nigam and Miller have shown that adopting different *focusing* annotations for such linear logic specifications, a range of other proof systems can also be specified. In particular, they have shown that natural deduction (normal and non-normal), sequent proofs (with and without cut), tableaux, and proof systems using general elimination and general introduction rules can all be derived from essentially the same linear logic specification by altering focusing annotations. By using elementary linear logic equivalences and the completeness of focused proofs, we are able to derive new and modular proofs of the soundness and completeness of these various proofs systems for intuitionistic and classical logics.

6.6. Proof complexity

Participant: Lutz Straßburger.

In proof complexity one usually distinguishes between proofs "with extension" and proofs "without extension", whereas in other areas of proof theory one distinguishes between proofs "with cut" and proofs "without cut". We have shown that with the use of deep inference it is possible to provide a uniform treatment for both classifications. This allows, in particular, to study cut-free proofs with extension, which is not possible with other formalisms. By using deep inference we could also give a new and simpler proof for the well-known theorem saying that extended Frege-systems p-simulate Frege-systems with substitution, and vice versa.

6.7. Proof normalization in sequent calculus

Participant: Stéphane Lengrand.

One way of obtaining cut-free proofs is to normalize a proof that contains cuts. Stéphane Lengrand has studied the termination of normalization procedures in sequent calculi.

The methodology uses proof-terms and rewrite systems on these objects, which have to be proved terminating.

Lengrand and Kikuchi in [19] have designed such a cut-elimination system for one of the simplest, and non-focused, sequent calculus for intuitionistic logic. Yet it provides a computation model that simulates the λ-calculus in a strong sense. Thus, termination of the system is at least as strong as that of the (simply-typed) λ-calculus.

Lengrand in [37] has shown the termination of a cut-elimination system presented in [3] for a focused sequent calculus in relation to call-by-value semantics.

The proof term approach could provide a systematic method to obtain cut-elimination results in focused sequent calculi such as LJF and LKF [6].

6.8. Computation with meta-variables

Participant: Stéphane Lengrand.

Meta-variables are central in proof search mechanisms to represent incomplete proofs and incomplete objects. They are used in almost all implementations of proof software, yet their meta-theory remains less explored than that of complete proofs and objects such as the λ-calculus.

Stéphane Lengrand and Jamie Murdoch Gabbay have studied an extension of λ-calculus with a particular kind of meta-variables originating from nominal logic. A paper on this study is in revision phase for a publication in Information and Computation.

The highlight of 2008 is the design of a typing system with polymorphism *à la* Hindley-Milner, as used in programming languages like CaML. The extension of λ-calculus with meta-variables creates particular difficulties for the typing algorithm to be implemented in the compiler. A deciding algorithm for typing is being developed using constraints and graph theory.

6.9. A Stochastic Linear Logic for Biological Computation

Participants: Kaustuv Chaudhuri, Joëlle Despeyroux.

Kaustuv Chaudhuri and Joëlle Despeyroux have proposed a logic, called HyLL (for Hybrid Linear Logic), for defining stochastic transition systems using linear implications for the state transitions and a modal logic with the worlds representing the rates of transitions. The worlds (rates) are reified in the propositional syntax using the connectives of hybrid logic, which allows directly encoding and reasoning about biological reaction systems, the intended use of this logic. This year, they have demonstrated the expressivity of the logic by giving an adequate encoding of the stochastic pi-calculus using a focused sequent calculus. A paper has been submitted and a technical report is in preparation.

7. Other Grants and Activities

7.1. Actions nationales

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).

7.2. Actions internationales

7.2.1. *Slimmer: INRIA funded international team*

Participants: David Baelde, Dale Miller.

Slimmer stands for *Sophisticated logic implementations for modeling and mechanical reasoning* is an “Equipes Associées” 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).

7.2.2. *Mobius: an EU Integrated Project on Proof Carrying Code*

Participants: Vivek Nigam, Olivier Delande, Joëlle Despeyroux, Dale Miller.

Mobius stands for “Mobility, Ubiquity and Security” and is a Proposal for an Integrated Project in response to the call FP6-2004-IST-FETPI. This proposal involve numerous sites in Europe and was awarded in September 2005. This large, European based project is coordinated via INRIA-Sophia.

7.2.3. *TYPES: coordinated action from IST*

Participants: Dale Miller, Joëlle Despeyroux.

TYPES has been an European project (a coordination action from the IST program) aiming at developing the technology of formal reasoning based on type theory. The project brought together 36 universities and research centers from 8 European countries (France, Italy, Germany, Netherlands, United Kingdom, Sweden, Poland and Estonia). It was the continuation of a number of European projects since 1992. The funding from the last project enabled the maintaining of collaboration within the community by supporting an annual workshop, a few smaller thematic workshops, one summer school, and visits of researchers to one another's labs. The funding of the project ended in April 2008.

7.2.4. *Hurbert Curien: PAI Amadeus on the "Realm of Cut Elimination"*

Participants: Dale Miller, Lutz Straßburger.

The "PAI" Amadeus for collaboration between France and Austria has approved the grant "The Realm of Cut Elimination" in November 2006. This proposal has allowed for collaborations between the Parsifal team and the groups of Agata Ciabattoni at Technische Universität Wien (Austria) and Michel Parigot at CNRS-PPS.

7.2.5. *Hurbert Curien: PAI Germaine De Stael "Deep Inference and the Essence of Proofs"*

Participants: Dale Miller, Lutz Straßburger.

This collaboration between Paris and Bern (Germany) aims at exploring some questions in the structure and identity of proofs. People involved in the Paris area are Lutz Straßburger, Dale Miller, Alexis Saurin, David Baelde, Michel Parigot, Stéphane Lengrand, and Séverine Maingaud. People involved in Bern are Kai Brünnler, Richard McKinley, and Phiniki Stouppa.

8. Dissemination

8.1. Services to the Scientific Community

8.1.1. *Organization of Conferences and Workshops*

Lutz Straßburger co-organized (with Pascal Manoury, Michel Parigot, and Paul Roziere from PPS) a workshop on "Structural Proof Theory" from 19–21 November 2008 in Paris. The workshop was partially supported by the projects ANR INFER, PHC "Realm of Cut Elimination", and PHC "Deep Inference and the Essence of Proofs".

Stéphane Lengrand organized two workshops on "Proof Search in Type Theories" at the École Polytechnique on 10th April and 5th-6th June 2006.

8.1.2. *Editorial activity*

Dale Miller has the following editorial duties.

- *Theory and Practice of Logic Programming*. Member of Advisory Board since 1999. Cambridge University Press.
- *ACM Transactions on Computational Logic (ToCL)*. Area editor for *Proof Theory* since 1999. Published by ACM.
- *Journal of Functional and Logic Programming*. Permanent member of the Editorial Board since 1996. MIT Press.
- *Journal of Logic and Computation*. Associate editor since 1989. Oxford University Press.

8.1.3. Participation in program committees

Dale Miller was a program committee member for the following conferences.

- PPDP 2008: 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, Valencia, 15-17 July.
- LSFA08: Third Workshop on Logical and Semantic Frameworks with Applications, Brazil.
- CSL08: 17th Annual Conference of the European Association for Computer Science Logic, 15-20 September, Bertinoro, Italy.
- TCS 2008: 5th IFIP International Conference on Theoretical Computer Science, August, Milano, Italy.
- FOSSACS 2008: Foundations of Software Science and Computation Structures. Budapest, Spring.
- FLOPS 2008: Ninth International Symposium on Functional and Logic Programming, 14-16 April, Ise, Japan.

8.1.4. Invited talks at Conferences or Workshops

Stéphane Lengrand has been invited to speak at the Workshop on Classical Logic and Computation 2008 (affiliated to IJCAR08), Reykjavik, 13 July 2008.

Dale Miller has been invited to speak at the following meetings.

- Journées du projet PEPS-Relations, University of Paris XIII, 15 - 16 December 2008.
- APS: 4th International Workshop on Analytic Proof Systems, part of LPAR 2008, Doha, Qatar, 22 November 2008.
- SOS 2008: Structural Operational Semantics, an affiliated workshop of ICALP 2008, Reykjavik, Iceland, 6 July 2008.
- WFLP 2008: 17th International Workshop on Functional and (Constraint) Logic Programming, Siena, 3-4 July 2008.
- LFMTP 2008: International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice (affiliated with LICS08), Pittsburgh, 23 June 2008.
- Dimostrazioni, Polaritá e Cognizione, Facoltà di Lettere e Filosofia, Università di Roma Tre, 18 April 2008.

8.2. Teaching

Dale Miller co-teaches the course “Logique Linéaire et paradigmes logiques du calcul” in the masters program MPRI (“Master Parisien de Recherche en Informatique”) (2004-2008).

Stéphane Lengrand teaches the course “Logique formelle et Programmation Logique” at the École d’ingénieur ESIEA (2008).

8.3. Evaluation

Dale Miller was an examinator for the PhD thesis of Samuel Mimram (Université Paris VII, 1 Dec 2008) and of Paolo Di Giamberardino (University of Rome 3 and University of the Mediterranean, 18 April 2008).

8.4. Internship supervision

From March to August 2008, Edlira Nano (MPRI) was writing her Master’s thesis on “Star-autonomous categories without units” under the supervision of Lutz Straßburger.

Also from March to August 2008, both Ivan Gazeau and Alexandre Viel conducted their MPRI internships at LIX under the supervision of Dale Miller.

9. Bibliography

Major publications by the team in recent years

[1] J. DESPEYROUX, P. LELEU. *Recursion over Objects of Functional Type*, in "Special issue of MSCS on 'Modalities in Type Theory'", vol. 11, n^{o} 4, August 2001.

[2] J. DESPEYROUX, F. PFENNING, C. SCHÜRMANN. *Primitive Recursion for Higher-Order Abstract Syntax*, in "Theoretical Computer Science (TCS)", vol. 266, n^{o} 1-2, September 2001, p. 1–57.

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

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

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

[6] C. LIANG, D. MILLER. *Focusing and Polarization in Linear, Intuitionistic, and Classical Logics*, Accepted to Theoretical Computer Science, 2008.

[7] R. MCDOWELL, D. MILLER. *Reasoning with Higher-Order Abstract Syntax in a Logical Framework*, in "ACM Trans. on Computational Logic", vol. 3, n^{o} 1, 2002, 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", vol. 294, n^{o} 3, 2003, 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", vol. 6, n^{o} 4, October 2005, 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", vol. 18, n^{o} 18, 2007, p. 536–601, http://arxiv.org/abs/cs.LO/0512086.

Year Publications

Doctoral Dissertations and Habilitation Theses

[11] D. BAELDE. *A linear approach to the proof-theory of least and greatest fixed points*, Ph. D. Thesis, Ecole Polytechnique, December 2008.

[12] A. SAURIN. *Une étude logique du contrôle (appliquée à la programmation fonctionnelle et logique)*, Ph. D. Thesis, Ecole Polytechnique, September 2008.

Articles in Non Peer-Reviewed Journal

[13] D. MILLER. *Formalizing operational semantic specifications in logic*, in "Concurrency Column of the Bulletin of the EATCS", October 2008, http://www.ru.is/faculty/luca/BEATCS/miller.pdf.

International Peer-Reviewed Conference/Proceedings

[14] D. BAELDE. *On the Expressivity of Minimal Generic Quantification*, in "LFMTP 2008: International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice", A. ABEL, C. URBAN (editors), 2008, p. 16–31, http://hal.inria.fr/inria-00284186/en/.

[15] 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, vol. 273, Boston: Springer, September 2008, p. 383–396, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/tcs08trackb.pdf.

[16] O. DELANDE, D. MILLER. *A neutral approach to proof and refutation in MALL*, in "23th Symp. on Logic in Computer Science", F. PFENNING (editor), IEEE Computer Society Press, 2008, p. 498–508, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lics08b.pdf.

[17] 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.

[18] A. GACEK, D. MILLER, G. NADATHUR. *Reasoning in Abella about Structural Operational Semantics Specifications*, in "LFMTP 2008: International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice", A. ABEL, C. URBAN (editors), 2008, p. 75–89, http://arxiv.org/pdf/0804.3914.pdf.

[19] K. KIKUCHI, S. LENGRAND. *Strong Normalisation of Cut-Elimination that Simulates β-Reduction*, in "11th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'08), Budapest, Hungary", R. AMADIO (editor), LNCS, vol. 4962, Springer-Verlag, March 2008, p. 380–394.

References in notes

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

[21] E. ALBERT, G. PUEBLA, M. V. HERMENEGILDO. *Abstraction-Carrying Code*, in "LPAR 2004: Logic for Programming, Artificial Intelligence, and Reasoning", Lecture Notes in Computer Science, vol. 3452, Springer, 2004, p. 380–397.

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

[23] A. W. APPEL, A. P. FELTY. *Lightweight Lemmas in Lambda Prolog*, in "16th International Conference on Logic Programming", MIT Press, November 1999, p. 411–425.

[24] 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.

[25] D. BAELDE, A. GACEK, D. MILLER, G. NADATHUR, A. TIU. *The Bedwyr system for model checking over syntactic expressions*, in "21th Conference on Automated Deduction (CADE)", F. PFENNING (editor), Lecture Notes in Artificial Intelligence, n^o 4603, Springer, 2007, p. 391–397, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/cade2007.pdf.

[26] R. BLOSSEY, L. CARDELLI, A. PHILLIPS. *A Compositional Approach to the Stochastic Dynamics of Gene Networks*, in "Transactions on Computational Systems Biology", vol. IV, n^o 3939, 2006, p. 99–122.

[27] O. DELANDE, D. MILLER, A. SAURIN. *Proof and refutation in MALL as a game*, Submitted for publication, October 2008.

[28] J. DESPEYROUX, P. LELEU. *Primitive recursion for higher-order abstract syntax with dependant types*, in "Informal proceedings of the FLoC'99 IMLA Workshop on Intuitionistic Modal Logics and Applications", June 1999, http://www-sop.inria.fr/parsifal/Joelle.Despeyroux/papers/imla99.ps.

[29] G. GENTZEN. *Investigations into Logical Deductions*, in "The Collected Papers of Gerhard Gentzen, Amsterdam", M. E. SZABO (editor), North-Holland, 1969, p. 68–131.

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

[31] J.-Y. GIRARD. *A Fixpoint Theorem in Linear Logic*, An email posting to the mailing list linear@cs.stanford.edu, February 1992.

[32] M. GORDON. *HOL: A Machine Oriented Formulation of Higher-Order Logic*, Technical report, n^o 68, University of Cambridge, July 1985.

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

[34] A. GUGLIELMI, L. STRASSBURGER. *Non-commutativity and MELL in the Calculus of Structures*, in "Computer Science Logic, CSL 2001", L. FRIBOURG (editor), LNCS, vol. 2142, Springer-Verlag, 2001, p. 54–68.

[35] L. HALLNÄS, P. SCHROEDER-HEISTER. *A Proof-Theoretic Approach to Logic Programming. II. Programs as definitions*, in "Journal of Logic and Computation", vol. 1, n^o 5, October 1991, p. 635–660.

[36] 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.

[37] S. LENGRAND. *Termination of lambda-calculus with the extra Call-By-Value rule known as assoc*, Technical report, LIX, 2007, http://hal.inria.fr/inria-00292029.

[38] 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.

[39] R. MCDOWELL. *Reasoning in a Logic with Definitions and Induction*, Ph. D. Thesis, University of Pennsylvania, December 1997.

[40] R. MCDOWELL, D. MILLER. *Cut-elimination for a logic with definitions and induction*, in "Theoretical Computer Science", vol. 232, 2000, p. 91–119.

[41] 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.

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

[43] D. MILLER, G. NADATHUR, F. PFENNING, A. SCEDROV. *Uniform Proofs as a Foundation for Logic Programming*, in "Annals of Pure and Applied Logic", vol. 51, 1991, p. 125–157.

[44] D. MILLER, A. SAURIN. *A game semantics for proof search: Preliminary results*, in "Proceedings of the Mathematical Foundations of Programming Semantics (MFPS05)", Electr. Notes Theor. Comput. Sci, n^o 155, 2006, p. 543–563, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/mfps05.pdf.

[45] 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.

[46] A. MOMIGLIANO, A. TIU. *Induction and Co-induction in Sequent Calculus*, in "Post-proceedings of TYPES 2003", M. C. S. BERARDI, F. DAMIANI (editors), LNCS, n^o 3085, January 2003, p. 293–308, http://www.lix.polytechnique.fr/Labo/Alwen.Tiu/linc.pdf.

[47] G. C. NECULA. *Proof-carrying code*, in "Conference Record of the 24th Symposium on Principles of Programming Languages 97, Paris, France", ACM Press, 1997, p. 106–119.

[48] G. C. NECULA, S. P. RAHUL. *Oracle-based checking of untrusted software.*, in "POPL", 2001, p. 142–154.

[49] L. C. PAULSON. *Isabelle: The Next 700 Theorem Provers*, in "Logic and Computer Science", P. ODIFREDDI (editor), Academic Press, 1990, p. 361–386.

[50] 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, n^o 1632, Springer, 1999, p. 202–206.

[51] E. PIMENTEL, D. MILLER. *On the specification of sequent systems*, in "LPAR 2005: 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning", Lecture Notes in Artificial Intelligence, n^o 3835, 2005, p. 352–366, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/lpar05.pdf.

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

[53] 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, vol. 3210, Springer-Verlag, 2004, p. 145–159.

[54] 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.

[55] A. TIU, D. MILLER. *Proof Search Specifications of Bisimulation and Modal Logic for the π-calculus*, Submitted May 2008..

[56] A. TIU, G. NADATHUR, D. MILLER. *Mixing Finite Success and Finite Failure in an Automated Prover*, in "Proceedings of ESHOL'05: Empirically Successful Automated Reasoning in Higher-Order Logics", December 2005, p. 79–98, http://www.lix.polytechnique.fr/Labo/Dale.Miller/papers/eshol05.pdf.

[57] A. TIU. *A Logical Framework for Reasoning about Logical Specifications*, Ph. D. Thesis, Pennsylvania State University, May 2004, http://www.lix.polytechnique.fr/Labo/Alwen.Tiu/etd.pdf.