Collected Abstracts of the 2013 LIX Colloquium on the

Theory and Application of Formal Proofs

November 5–7, 2013
Laboratoire d'Informatique de l'Ecole Polytechnique
Palaiseau, France

THEME

Formal proofs are becoming increasingly important in a number of domains in computer science and mathematics. The topic of the colloquium is structural proof theory, broadly construed. Some examples of relevant topics:

Structure
Sequential and parallel structure in proofs; sharing and duplication of proofs; permutation of proof steps; canonical forms; focusing; polarities; graphical proof syntax; proof complexity; cut-eliminiation strategies; CERes

Proof Checking
Generating, transmitting, translating, and checking proof objects; universal proof languages; proof certificates; proof compression; cut-introduction; certification of high-performance systems (SMT, resolution, etc.)

Proof Search
Automated and interactive proof search in constrained logics (linear, temporal, bunched, probabilistic, etc.); mixing deduction and computation; induction and co-induction; cyclic proofs; computational interpretations

List of talks (in presentation order)

1. **Alessio Guglielmi**

   Proof Composition Mechanisms and Their Geometric Interpretation

2. **Richard McKinley**

   Higher-order sequent proofs, higher-order nets

3. **Willem Heijltjes** and **Robin Houston**

   The proof equivalence problem for multiplicative linear logic is PSPACE-complete

4. **Roman Kuznets**

   Craig Interpolation, Proof-Theoretically via Nested Sequents

5. **Sorin Stratulat**

   Reductive-free cyclic induction reasoning

6. **Richard Moot**

   First-order linear logic as a general framework for logic-based computational linguistics

7. **Thomas Studer**

   Towards syntactic cut-elimination for temporal logics

8. **James Brotherston**

   Cyclic Abduction of Inductive Safety & Termination Preconditions

9. **Roberto Maieli**

   Construction of Bipolar Focussing Proof Structures

10. **Sara Negri**

    Extending the scope of labelled sequent calculi: the case of classical counterfactuals

11. **Roy Dyckhoff**

    Automating LPO Reasoning about Termination

12. **Anupam Das**

    Some ideas on bounded arithmetics for systems of monotone proofs

13. **George Metcalfe**

    Proof Theory for Lattice-Ordered Groups

14. **Dominic Hughes**

    First-order proofs without syntax

15. **Stefan Hetzl** and **Daniel Weller**

    Expansion Trees with Cut

16. **Matthias Baaz**

    Interpolation in finitely-valued first-order logics

17. **Sebastian Eberhard** and **Stefan Hetzl**

    Guessing induction formulas for proofs of universal statements

18. **Agata Ciabattoni**

    Power and Limits of Structural Rules

19. **Nicolas Guenot**

    Linear Session Types for Solos

20. **Iliano Cervesato** and **Jorge Luis Sacchini**

    Meta-Reasoning in the Concurrent Logical Framework CLF

21. **Amy Felty**, **Alberto Momigliano** and **Brigitte Pientka**

    Toward a Theory of Contexts of Assumptions in Logical Frameworks

22. **Fabien Renaud**, **Dale Miller** and **Zakaria Chihani**

    A Semantics for Proof Evidence

23. **Herman Geuvers**

    Pure Type Systems revisited

24. **Danko Ilik**

    Type isomorphisms in presence of strong sums: axiomatizability, practical decidability and subtyping

25. **Noam Zeilberger**

    Proofs in monoidal closed bifibrations

26. **José Espírito Santo**, **Ralph Matthes**, **Koji Nakazawa** and **Luís Pinto**

    A classical theory of values and computations

27. **Chad E. Brown**

    Distributed Mathematics

Proof Composition Mechanisms and Their Geometric Interpretation

Alessio Guglielmi
University of Bath

Given two proofs $A \Rightarrow B$ and $C \Rightarrow D$, where formulae A and C are premisses and B and D are conclusions, we consider the following three proof-composition mechanisms, of which the first two define deep inference:

1) by a connective $*$: $(A \Rightarrow B) * (C \Rightarrow D)$;

2) by an inference rule B/C: $(A \Rightarrow B)/(C \Rightarrow D)$;

3) by substitution for an atom a: $(A \Rightarrow B)\{a \leftarrow (C \Rightarrow D)\}$.

We call atomic flows certain directed graphs obtained from proofs by tracing their atom occurrences. Atomic flows retain all the structural information of proofs and lose all their logical information. The three compositions above correspond to natural operations on atomic flows and they yield certain geometric invariants.

In the first part of the talk I will show some of the many uses of the geometric properties of atomic flows, in particular when applied to the classical problem of normalising proofs. I will argue that cut admissibility is an instance of a much more general geometric problem and that doing structural proof theory with shapes instead of formulae and sequents might be much more effective and natural.

In the second part of the talk I will illustrate, with many examples, the current effort in defining a formalism allowing proof composition by substitution. Again, atomic flows are the guiding principle. This way we can obtain an exponential speed-up in the size of proofs and capture correspondingly big equivalence classes of proofs in a way that does not adversely impact normalisation.

The talk with all the references will appear at `http://cs.bath.ac.uk/ag/t/PCMTGI.pdf`.

Higher-order sequent proofs, higher-order nets

Richard McKinley

IAM, Universität Bern, Switzerland

Abstract. In this talk, we present a new syntax for Linear Logic, inspired by natural deduction. In the sequent-calculus formulation of this syntax, the leaves of sequent trees can be labelled not only with axioms, but also with assumptions. The usual linear-logic operations of dereliction, contraction and weakening are replaced by a single abstraction operation, which introduces a ?-formula in the conclusion and discharges any number of appropriate assumptions. This yields a kind of "sequent-calculus in natural deduction style". The usual promotion rule of linear logic is replaced by a much simpler ! rule which allows to pass from a (parametric) proof of $\vdash A$ to a (identically paramaterised) proof of $\vdash !A$.

A cost of this simplified presentation is that cut-elimination does not hold as in standard sequent systems: instead, as with natural deduction, we define a natural notion of *redex* and accompanying notion of *normality*; a simple translation from the standard sequent calculus for LL yields soundness for the full calculus with cut and completeness for the calculus of normal proofs.

Unlike the standard calculus for LL, the new rules for the exponentials define them uniquely: if we use these rules to add two sets of exponentials to MALL, they will be logically equivalent, in contrast to the traditional Linear Logic rules for exponentials. This is not in contradiction to the existence of sub-exponentials; given an ordinary sequent calculus with sub-exponentials, adding exponential rules of our calculus yields a canonical "super-exponential" above the existing subexponentials.

From the perspective of dynamics, this calculus bears the same relation to the usual calculus as ordinary natural deduction bears to resource-sensitive calculi of explicit substitutions. The cut-reduction dynamics of the exponentials in this calculus are as one would expect, given the inspiration of natural deduction, if we treat the !-rule as creating a *value*: given a cut between a ?-abstraction and a !-value, the cut is reduced by replacing each formula discharged by the ?-rule with a copy of the proof contained in the value. This inspires the name *higher-order* sequent-proofs

This idea of representing exponentials using abstraction extends to a proof-net calculus for MELL. This proof-net calculus has the aesthetic advantage that each box has only a single output, and the technical advantage that each proof has an easily computed canonical form, while the usual proof-net calculus for MELL has non-canonicity owing to the interactions between the auxilliary outputs of boxes and contractions.

If time permits, we will look at some further directions/questions for this work, for example:

- Linear logic proofs as higher-order processes.
- Nested Linear Logic
- Differential linear logic
- The linear modal cube

The proof equivalence problem for multiplicative linear logic is PSPACE-complete

Abstract

Willem Heijltjes and Robin Houston

October 1, 2013

MLL *proof equivalence* is the problem of deciding whether one proof in multiplicative linear logic may be turned into another by a series of commuting conversions. It is also the *word problem* for $\star$-autonomous categories (Barr, 1991): the problem of deciding whether two term representations denote the same morphism in any $\star$-autonomous category. In MLL$^-$, without the two units, proof equivalence corresponds to syntactic equality on *proof nets* (Girard, 1987; Danos and Regnier, 1989), and is linear-time decidable. For full MLL, thanks to many years of work on proof nets (Trimble, 1994; Blute et al., 1996; Straßburger and Lamarche, 2004; Hughes, 2012) we know that the proof equivalence problem corresponds to a reasonably simple graph-rewiring problem.

On the other side of the fence, thanks to many years of work on combinatorial games and complexity theory (Flake and Baum, 2002; Hearn and Demaine, 2005, 2009; Gopalan et al., 2006; Ito et al., 2011) we know many examples of rewiring problems on graphs that are PSPACE-complete. We will give a reduction from the configuration-to-configuration problem for *Nondeterministic Constraint Logic* (Hearn and Demaine, 2005, 2009), a graphical formalism specifically designed for easy problem reduction.

The PSPACE-completeness result rules out a satisfactory notion of proof net for MLL with units, and in particular for this reason it may come as a surprise. However, PSPACE-completeness is not unusual for graph rewiring problems, nor for *reconfiguration* problems (Ito et al., 2011) of the kind to which MLL proof equivalence belongs. A reconfiguration problem is the problem of finding a path across related solutions to a given decision problem—in this case MLL proof search. Reconfiguration problems whose associated decision problem is NP-complete, as is MLL proof search, frequently are PSPACE-complete; an important example, *satisfiability-reconfiguration* (Gopalan et al., 2006) is the problem of finding a path across satisfying assignments to a boolean formula by changing the value of one atom at a time.

References

Michael Barr. *-Autonomous categories and linear logic. *Mathematical Structures in Computer Science*, 1:159–178, 1991.

Richard Blute, Robin Cockett, Robert Seely, and Todd Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, 113:220–296, 1996.

Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

Gary William Flake and Eric B. Baum. Rush hour is PSPACE-complete, or, "why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1–2):895–911, 2002.

Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. The connectivity of boolean satisfiability: Computational and structural dichotomies. In *Automata, Languages and Programming*, volume 4051 of *LNCS*, pages 346–357. Springer Berlin Heidelberg, 2006.

Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1–2):72–96, 2005.

Robert A Hearn and Erik D Demaine. *Games, puzzles, and computation*. AK Peters, Ltd., 2009.

Dominic J.D. Hughes. Simple free star-autonomous categories and full coherence. *Journal of Pure and Applied Algebra*, 216(11):2386–2410, 2012.

Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12–14):1054–1065, 2011.

Lutz Straßburger and François Lamarche. On proof nets for multiplicative linear logic with units. *CSL*, pages 145–159, 2004.

Todd Trimble. *Linear logic, bimodules, and full coherence for autonomous categories*. PhD thesis, Rutgers University, 1994.

Craig Interpolation Proof-Theoretically via Nested Sequents

Roman Kuznets*
(joint work with Melvin Fitting)

Institute of Computer Science and Applied Mathematics
University of Bern
kuznets@iam.unibe.ch

Craig interpolation is a standard property that logics are tested for. A logic L is said to enjoy the *Craig interpolation property* (CIP) if for every implication such that $\mathsf{L} \vdash A \to B$, there exists a formula C, called an *interpolant* of A and B, such that $\mathsf{L} \vdash A \to C$, $\mathsf{L} \vdash C \to B$, and C only uses the "common language" of the formulas A and B. For propositional monomodal logics, the "common language" means propositional atoms common to A and B.

The proof-theoretic method of proving the Craig interpolation property consists in constructing an interpolant by induction on the depth of a given cut-free derivation in a suitable sequent calculus. This method is constructive in that it yields an algorithm for constructing an interpolant rather than simply demonstrates its existence. However, the applicability of this method is limited to the logics that possess a cut-free sequent calculus.

Various generalizations of sequent calculi have been developed over the years: among them, hypersequent, nested sequent, and labelled sequent calculi. These calculi provide cut-free descriptions of a wider range of logics. For instance, among modal logics without a known cut-free sequent calculus, $\mathsf{S5}$ has a cut-free hypersequent representation and $\mathsf{K5}$ has a cut-free nested sequent representation. Thus, to apply the proof-theoretic method to such logics, one of the above-mentioned generalizations of sequents needs to be used as a proof system. Unfortunately, adapting the proof-theoretic method to the more general sequent-like systems is far from being trivial.

We present such an adaptation of the proof-theoretic method of proving CIP to nested sequent calculi and apply our method to obtain a uniform constructive proof of CIP for all extensions of the minimal normal modal logic K with any combination of the modal axioms d, t, b, $\mathsf{4}$, and $\mathsf{5}$. For three of these 15 modal logics, namely, $\mathsf{B5}$, $\mathsf{K45}$, and $\mathsf{D45}$, the Craig interpolation property has not been previously known, to the best of our knowledge. Besides the conjunction and disjunction of interpolants, also used in the constructive interpolation via ordinary sequents, our method only requires three additional operations on interpolants, one of which simply permutes structural boxes, for all 15 modal logics. The presence of structural boxes in nested sequents being interpolated necessitates additional structure for interpolants, in the general case making them more than just formulas.

* Supported by Swiss National Science Foundation grant PZ00P2–131706.

Reductive-free cyclic induction reasoning

- abstract -

Sorin STRATULAT

Université de Lorraine
LITA, Department of Computer Science
Ile du Saulcy, Metz, F-57000, France
sorin.stratulat@univ-lorraine.fr

Noetherian induction is an effective formal proof method to finitely capture the cyclic reasoning encountered during the traditional schemata-based induction proofs, i.e., when the proof of a formula ϕ requires as *induction hypotheses* (IHs) 'not-yet proved' instances of ϕ, as well as the mutual induction proofs, i.e., when the proofs of ϕ and another formula from the same proof session mutually require as IHs instances of the other. Any Noetherian induction principle is based on a well-founded (induction) ordering that guarantees the sound usage of the IHs and the termination of the cyclic reasoning if some ordering constraints are satisfied.

Since the seminal paper by Musser [4], two important groups of first-order Noetherian induction-based proof methods are distinguished: i) the *explicit* induction, that covers the traditional schemata-based methods, and ii) the *implicit* induction, based on reductive procedures. From a qualitative point of view, they can be distinguished according to the kind of elements the well-founded ordering underlying the Noetherian induction principle is defined on: the first group helps to prove that some property, formalized as a universally quantified first-order formula, holds for a set of (vectors of) *terms*, while the second can check the validity of a set of (first-order) *formulas*. In addition, they have features that complement each other.

The explicit induction methods perform locally the cyclic reasoning, at the formula level, by the means of *induction schemas* that attach a set of IHs to some formula, called *induction conclusion*. An example of common schemata-based induction is the *structural induction* [3] for which the induction schemas are generated from recursively defined data structures. The locality feature of explicit induction reasoning helps its integration into sequent-based inference systems in terms of inference rules encoding the explicit induction schemas. On the other hand, it may happen that the induction schemas define useless IHs or that the proof of the induction conclusions lack crucial IHs. Moreover, any induction conclusion and its attached IHs are instances of a same formula, hence it cannot help to define mutual induction schemas.

In turn, implicit induction reasoning is lazy and by need, allowing instances of *any* conjecture from a proof to play the role of IHs as long as they are smaller or equivalent (w.r.t. the well-founded ordering) to, and sometimes strictly smaller than, the induction conclusion. Hence the possibility to naturally deal with mutual induction. On the other hand, the induction reasoning is defined by one global induction schema that helps proving all conjectures from a proof. The induction reasoning cannot be captured by only one inference rule, hence the method cannot be directly integrated into sequent-based theorem provers. The induction ordering is unique, defined on for-

mulas, and it needs additional *reductive* constraints to be satisfied by the processed and the newly generated conjectures when applying an inference rule.

This talk is about a recent formula-based cyclic method, firstly presented in [7][1]. It will be shown that it generalizes and keeps the best features of the term- and formula-based induction methods. The application of any IH involved in a cyclic reasoning can be validated by a *cycle*, representing a circular list of formulas from the proof. Any cycle can validate one or several IHs using a proof strategy that allows for simultaneous induction. The cyclic reasoning is local, at the cycle level, reductive-free, and naturally performs mutual and lazy induction.

From a theoretical point of view, the method synthesizes the overall usage of Noetherian induction reasoning in first-order logic. A particular attention will be drawn on the relations between term- and formula-based induction principles, by establishing the conditions required to 'customize' term- to formula-based induction proofs, and viceversa. It will be shown that any reductive formula-based proof can be directly transformed into a reductive-free cyclic proof. Also, any term-based cycle can be customized to a formula-based reductive-free cycle checking only one IH.

From a practical point of view, it allows for less restrictive specifications, more efficient implementations and proof certification processes. We conclude with some experimental results about the cyclic proof method implemented into the implicit induction theorem prover SPIKE [2, 6, 1]; comparisons between the implicit and cyclic induction proofs of conjectures used for the validation proof of a conformance algorithm for the ABR protocol [5] will be given.

References

1. G. Barthe and S. Stratulat. Validation of the JavaCard platform with implicit induction techniques. In R. Nieuwenhuis, editor, *RTA*, volume 2706 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2003.
2. A. Bouhoula. Automated theorem proving by test set induction. *Journal of Symbolic Computation*, 23(1):47–77, 1997.
3. R. M. Burstall. Proving properties of programs by structural induction. *The Computer Journal*, 12:41–48, 1969.
4. D. R. Musser. On proving inductive properties of abstract data types. In *POPL*, pages 154–162, 1980.
5. M. Rusinowitch, S. Stratulat, and F. Klay. Mechanical verification of an ideal incremental ABR conformance algorithm. *J. Autom. Reasoning*, 30(2):53–177, 2003.
6. S. Stratulat. A general framework to build contextual cover set induction provers. *J. Symb. Comput.*, 32(4):403–445, 2001.
7. S. Stratulat. A unified view of induction reasoning for first-order logic. In A. Voronkov, editor, *Turing-100 (The Alan Turing Centenary Conference)*, volume 10 of *EPiC Series*, pages 326–352. EasyChair, 2012.

[1] The paper received one of the best paper awards at 'The Turing Centenary Conference' in Manchester, June 2012.

First-order linear logic as a general framework for logic-based computational linguistics

Richard Moot

October 1, 2013

The Lambek calculus (Lambek 1958) is one of the simplest possible logics which can be used to give an account of natural language syntax — as well as providing a very elegant syntax-semantics interface by means of the Curry-Howard isomorphishm, which guarantees that each proof corresponds to a reading of the sentence. With the arrival of linear logic and its proof theoretic innovations, proof nets have become one of the standard ways of performing proof search (or *parsing*) for Lambek calculus grammar, overcoming the limitations of sequent-based proof search methods developed in the eighties.

However, the Lambek calculus also has some well-known limitations: Lambek grammars generate only context-free languages, widely considered insufficient for natural language, and, given the non-commutativity of the logic, the syntax-semantics interface is too rigid in some cases (for example, in not producing enough derivations/scope possibilities for generalized quantifiers, (Moortgat 1997, Moot & Retoré 2012)). Several extensions of the Lambek calculus have been proposed, with the goal of solving these two problems. These extensions can roughly be grouped into to general categories: the *structural* extensions license structural rules like commutativity based on formula decorations, much like the exponentials of linear logic license contraction and weakening; the *tuple-based* extensions generalize the Lambek calculus, which is the logic of strings, by proposing logics of tuples of strings. We will mostly be concerned with examples of the second category, which include the Displacement calculus (Morrill, Valentín & Fadda 2011), Abstract Categorial Grammars (de Groote 2001), Linear Grammars (Pollard & Smith 2012) and Hybrid Type-Logical Grammars (Levine & Kubota 2012) (though I will show how to emulate and improve upon some key aspects of the first category as well).

The main result of this talk will be to extend the result of (Moot 2013) and show that each of these logics corresponds to a fragment of MILL1 (first-order multiplicative intuitionistic linear logic) by providing a translation of grammars in these different systems into MILL1 grammars. This makes MILL1 a general framework for studying and comparing these systems. Doing so will bring to light some of the limitations of existing systems. I will also show that several additional linguistic phenomena receive a simple treatment in MILL1. As a simple corollary, this means we can use linear logic's proof nets for each of these

systems as well — by translation into first-order linear logic and using the proof nets of (Girard 1991) — providing a new and uniform proof search strategy and greatly simplifying existing proof systems for the different calculi.

References

de Groote, P. (2001), Towards abstract categorial grammars, *in* 'Proceedings ACL 2001', Toulouse.

Girard, J.-Y. (1991), Quantifiers in linear logic II, *in* G. Corsi & G. Sambin, eds, 'Nuovi problemi della logica e della filosofia della scienza', Vol. II, CLUEB, Bologna, Italy. Proceedings of the conference with the same name, Viareggio, Italy, January 1990.

Lambek, J. (1958), 'The mathematics of sentence structure', *American Mathematical Monthly* **65**, 154–170.

Levine, R. & Kubota, Y. (2012), Gapping as like-category coordination, *in* D. Béchet & A. Dikovsky, eds, 'Proceedings of LACL 2012', pp. 135–150.

Moortgat, M. (1997), Categorial type logics, *in* J. van Benthem & A. ter Meulen, eds, 'Handbook of Logic and Language', Elsevier/MIT Press, chapter 2, pp. 93–177.

Moot, R. (2013), 'Extended lambek calculi and first-order linear logic', to appear in Springer Lecture Notes in Artificial Intelligence.

Moot, R. & Retoré, C. (2012), *The Logic of Categorial Grammars*, Lecture Notes in Artificial Intelligence, Springer.

Morrill, G., Valentín, O. & Fadda, M. (2011), 'The displacement calculus', *Journal of Logic, Language and Information* **20**(1), 1–48.

Pollard, C. & Smith, E. A. (2012), A unified analysis of the same, phrasal comparatives, and superlatives, *in* 'Proceedings SALT 22', pp. 307–325.

Towards syntactic cut-elimination for temporal logics

Thomas Studer*

The proof theory of temporal logics, and of modal fixed point logics in general, is notoriously difficult. It is not even clear how to design a finitary deductive system for linear time temporal logic LTL with nice proof-theoretic properties.

Brünnler and Lange [1] proposed an elegant formalism for LTL using focus games from Lange and Stirling [2] as an inspiration. The main technical feature of their system are annotated sequents, which are employed to derive greatest fixed points. However, some very basic proof-theoretic problems turn out to be surprisingly hard in this setting. For instance, despite the admissibility of several structural rules, including cut, being proved semantically in [1], it remains to prove the same facts by proper proof-theoretic methods. Even the admissibility of weakening, which is quite trivial for most types of sequent calculi, is far from being simple for annotated sequents due to the presence of sequent contexts in the annotations. More precisely, a permutation of a weakening rule upward in an annotated sequent derivation may break the match between the current sequent context and the one recorded in an annotation, the match necessary to complete the derivation.

In this talk, we present our recent results and ideas about syntactic cut-elimination procedures for annotated sequent calculi for temporal logics.

References

[1] Kai Brünnler and Martin Lange. Cut-free sequent systems for temporal logic. *Journal of Logic and Algebraic Programming*, 76(2):216 – 225, 2008.

[2] Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *In Proc. 16th IEEE Symp. on Logic in Computer Science, LICS01*, pages 357–365. IEEE, 2001.

*Institut für Informatik und angewandte Mathematik, Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland, tstuder@iam.unibe.ch

Cyclic Abduction of Inductive Safety & Termination Preconditions

James Brotherston
Dept. of Computer Science,
University College London

Whether a given pointer program ever encounters a memory fault, or ever terminates, are natural (and undecidable) problems in program analysis. Usually, we are interested in establishing such safety and termination properties under a given *precondition* in logic that expresses some known initial conditions about the program.

In this talk, we consider the even more difficult problem of *inferring* a reasonable precondition, in separation logic with inductive definitions, for such heap-aware programs. Indeed, we will show that this problem can be seen as a matter of heuristically guided search in a formal proof system.

We demonstrate a new method, called *cyclic abduction*, for automatically inferring the inductive definitions of safety and termination preconditions for heap-manipulating programs. Cyclic abduction essentially works by searching for a *cyclic proof* of safety or termination as desired, abducing definitional clauses of the precondition as necessary to advance the proof search process. In particular, the formation of a cycle in the proof typically forces the instantiation of recursion in the corresponding precondition.

Our cyclic abduction method has been implemented in the Cyclist theorem prover, and tested on a suite of small programs. In particular, we are often able to automatically infer the preconditions that, in other tools, previously had to be supplied by hand.

This is joint work with Nikos Gorogiannis (also at UCL).

Construction of Bipolar Focussing Proof Structures*

Roberto Maieli
Dipartimento di Matematica e Fisica, Università "Roma Tre"
Largo San Leonardo Murialdo, 1 – 00146 Roma, Italia
roberto.maieli@uniroma3.it

October 17, 2013

Keywords: linear logic, sequent calculus, focussing proofs, proof nets, rewriting graphs, proof construction.

1 Introduction

This work takes a further step towards the development of a research programme, launched by Andreoli in 2001 (see [1], [2] and [3]), which aims at a theoretical foundation of a computational programming paradigm based on the construction of proofs of *linear logic* (LL, [9]). Naively, this paradigm relies on the following isomorphism: *proof = state* and *construction step* (or *inference*) *= state transition*.
Traditionally, this paradigm is formulated as an incremental (bottom-up) construction of possibly *incomplete* (i.e., *open* or without logical axioms) proofs of the *bipolar focussing sequent calculus* . This calculus satisfies the property that the complete (closed or with logical axioms) bipolar focussing proofs are fully representative of all the closed proofs of linear logic (this correspondence is, in general, not satisfied by the polarized fragments of linear logic). Bipolarity and focussing properties ensure more compact proofs since they get rid of some irrelevant intermediate steps during the proof search (or proof construction).
Now, while the view of proof construction is well adapted to theorem proving, it is inadequate when we want to model some proof-theoretic intuitions behind, e.g., concurrent logic programming which requires very flexible and modular approaches. Due to their artificial sequential nature, sequent proofs are difficult to cut into modular (reusable) concurrent components. A much more appealing solution consists of using the technology offered by *proof nets* of linear logic or, more precisely, some forms of de-sequentialized (geometrical, indeed) proof structures in which the composition operation is simply given by (possibly, constrained) juxtaposition, obeying to some correctness criteria.
Actually, the proof net construction, as well the proof net cut reduction, can be performed in parallel (concurrently), but despite the cut reduction, there may not exist executable (i.e., sequentializable) construction steps: in other words, construction steps must satisfy a, possibly efficient, correctness criterion.
Here, a proof net net is a particular "open" proof structure, called *transitory net*, that is incrementally built bottom-up by juxtaposing, via construction steps, simple proof structures or modules, called *bipoles*. Roughly, bipoles correspond to Prolog-like *methods* of Logic Programming Languages: the *head* is represented by a multiple trigger (i.e., a multiset of positive atoms) and the *body* is represented by a layer of negative connectives with negative atoms.
We say that a construction step is correct (that is, a *transaction*) when it preserves, after the juxtaposition, the property of being a transitory net: that is the case when the abstract transitory structures *retracts* (by means of a finite sequence of rewriting steps) to elementary collapsed graphs, called *normal forms*. Each retraction step consists of a simple (local) graph deformation or graph rewriting. The rewriting system here proposed is shown to be convergent (i.e., terminating and confluent), moreover, it preserves, step by step, the property of being a transitory structure. Transitory nets (i.e., retractible structures) correspond to derivations of the focussing bipolar sequent calculus.
The first retraction algorithm for checking correctness of the proof structures of the pure (units-free) multiplicative fragment of linear logic (MLL), was given by Danos in his thesis ([6]); the complexity of this algorithm was later shown to be linear, in the size of the given proof structure, by Guerrini in [11]. Subsequently, the retractility criterion was extended, respectively, by the author, in [16], to the pure multiplicative and additive (MALL) proof nets with *boolean weights* and then by Fouqueré and Mogbil, in [8], to polarized multiplicative and exponential proof structures. Here, we further extend retractility to more liberal proof structures that are: focussing, bipolar, open, with generalized (n-ary) links and weights-free.

*Paper available at: http://logica.uniroma3.it/∼maieli/CoTPS.pdf.

Technically, one of the main contributions of this work is to provide a very simple syntax for open proof structures that allows to extend the paradigm of proof construction to the MALL fragment of LL. We set, indeed, a precise correspondence (called, *sequentialization* between focussing bipolar (open) sequent proofs and correct transitory structures.
Although there already exist several satisfactory syntaxes for MALL proof structures (see, e.g., those ones given, respectively, by Hughes–van Glabbeek in [13], Laurent–Maieli in [15], Di Giamberardino in [7] and Heijltjes in [12]), our choice is motivated by the fact that the proposed simple retractile correctness criterion allows to approach "efficiently" the problem of the incremental construction of transitory proof structures. Actually, in the construction process, a construction step is correct (i.e. a transition) when it preserves the property of being a transitory net. Now, checking correctness (the retractility of the given structure) is a task which may involve visiting (retracting) a large portion of the expanded net. So, since this construction is performed collaboratively and concurrently by a cluster of bipoles, we need at least to restrict the traveling area (the retraction area) in such a way to possibly reduce conflicts among bipoles. Limited to the MLL case, some good bounds for these tasks can be found in [2, 3]. We now show that checking correctness (contractility) of a MALL transitory net, after a construction attempt, is a task that can be performed by restricting to some "minimal" (partially retracted) transitory nets. The reason of that is that some subgraphs of the given transitory net will not play an active role in the construction process, since they are already correct and encapsulated (i.e., interface-free): so, their retraction can be performed regardless of the construction process.
Finally, comparing with the related literature, we only mention other analogous attempts to proof-theoretically model "concurrent logic programming", notably:
– some works of Pfenning and co-authors, from 2002 and later (see, e.g., [4]), which rely neither on focussing (or polarities) nor on proof nets but on "softer" notions of sequent calculus proofs;
– some works of Miller and co-authors which generalize focused sequent proofs to admit multiple "foci": see, e.g., [17] and [5]; the latter also provides a bijection to the unit-free proof nets of the MLL fragment, but it only discusses the possibility of a similar correspondence for larger fragments.

References

[1] J.-M. Andreoli. Focussing and Proof Construction. *APAL*, 2001.

[2] J.-M. Andreoli. Focussing proof-nets construction as a middleware paradigm. LNCS, 2002.

[3] J.-M. Andreoli and L. Mazarè, L.. Concurrent Construction of Proof-Nets. *LNCS*, 2003.

[4] I. Cervesato, F. Pfenning, D. Walker, and K. Watkins. A concurrent logical framework II: Examples and applications. *Technical Report CMU-CS-02-102*, May 2003.

[5] K. Chaudhuri, D. Miller and A. Saurin. Canonical Sequent Proofs via Multi-Focusing. In *IFIP TCS*, 2008.

[6] V. Danos. La Logique Linéaire appliquée à l'étude de divers processus de normalisation. *PhD Thesis*, 1990.

[7] P. Di Giamberardino. Jump from parallel to sequential proofs: on polarities and sequentiality in Linear Logic. *PhD Thesis*, Universitá Roma Tre – IML-CNRS, 2008.

[8] C. Fouqueré and V. Mogbil. Rewritings for polarized multiplicative-exponential proof structures. *ETCS*, 2008.

[9] J.-Y. Girard. Linear Logic. *TCS*, 1987.

[10] J.-Y. Girard. Proof nets:the parallel syntax for proof theory. *Logic&Algebra*, 1996.

[11] S. Guerrini. A linear algorithm for MLL proof net correctness and sequentialization. *TCS*, 2011.

[12] W. Heijltjes. Proof Nets for Additive Linear Logic with Units. In *LICS*, 2011.

[13] D. Hughes and R. van Glabbeek. Proof Nets for Unit-free Multiplicative-Additive Linear Logic. In *LICS*, 2003.

[14] O. Laurent. *Polarized Proof-Nets: Proof-Nets for LC.* In J.-Y. Girard, editor, TLCA 1999, *LNCS*, 1999.

[15] O. Laurent and R. Maieli. Cut Elimination for Monomial MALL Proof Nets. In *LICS*, 2008.

[16] R. Maieli. *Retractile Proof Nets of the Purely Multiplicative-Additive Frag. of Linear Logic*. *LPAR*, 2007.

[17] D. Miller. Forum: a multiple-conclusion specification logic. *TCS*, 1996.

Extending the scope of labelled sequent calculi: the case of classical counterfactuals

Sara Negri
University of Helsinki

Labelled sequent calculi provide a versatile formalism for the proof-theoretical investigation of large families of non-classical logics, through the internalization of Kripke semantics. In recent work (Negri 2013) we have shown that the method covers frame conditions beyond geometric implication and the whole of Sahlqvist fragment of modal logics. The semantics of important intensional connectives such as counterfactual conditionals, however, is based on a more general, neighbourhood-style, semantics. A conditional implication $A > B$ is said to be true at a world x if either A is never possible, or if there is a neighbourhood of x of worlds similar to x where the antecedent is satisfied and the (classical) implication always holds. The strong assumption of existence of a minimal satisfying neighbourhood that witnesses the *ceteris paribus* similarity condition, criticized both on philosophical and mathematical grounds since the work of Lewis, permits to eliminate the quantifier alternation in the semantic explanation of counterfactuals and to obtain a truth condition structurally similar to that of the standard modal operator. Both labelled tableaux and sequent systems have been formulated on the basis of this assumption (Olivetti et al., 2007, Priest 2008).

It will be shown how complete sequent calculi for classical counterfactuals can be obtained without this simplifying assumption. In particular, the systems obtained enjoy invertibility of the rules, height-preserving admissibility of contraction, and syntactic cut elimination (Negri and Sbardolini 2013).

References

[1] Lewis, D. (1973) *Counterfactuals*. Blackwell.

[2] Negri, S. (2013) Proof analysis beyond geometric theories: from rule systems to systems of rules. *Journal of Logic and Computation*, in press.

[3] Negri, S. and J. von Plato (2011)*Proof Analysis*. Cambridge University Press.

[4] Negri, S. and G. Sbardolini (2013) Systems of proof for classical counterfactual, ms.

[5] Olivetti, N., G.L. Pozzato, and C.B. Schwind (2007) A sequent calculus and a theorem prover for standard conditional logics. *ACM Transaction on Computational Logic*, vol. 8, no. 4.

[6] Priest, G. (2008) *An Introduction to Non-Classical Logic*, Cambridge University Press.

The Theory and Application of Formal Proofs
LIX Colloquium, November 2013

Automating LPO Reasoning about Termination

(Abstract)

Roy Dyckhoff
rd@st-andrews.ac.uk
St Andrews University, Scotland

This gives details of an essential but tedious part of some work reported elsewhere [3]. First, the background. It is well-known that, in intuitionistic logic, sequent calculus derivations (with or without *Cut*) are recipes for constructing natural deductions, and that, by the Curry-Howard correspondence, one can represent both the former and the latter using terms of a typed lambda calculus. Natural deduction terms may, by various standard reductions, be normalised; there are however many sequent calculi S, reduction systems R for S and reduction strategies for R, including but not limited to those given by Gentzen. We presented in [3] a complete single-succedent sequent calculus (essentially the Ketonen-Kleene system **G3ip** from [8], including all the usual zero-order connectives, including disjunction and absurdity) and a cut-reduction system, with the virtues that (a) it is strongly normalising (b) it is confluent and (c) it allows a surjective homomorphism from cut-elimination to normalisation: in other words, a homomorphism from **G3ip** derivations to **NJ** natural deductions with the property that each cut reduction step translates into a sequence of zero or more reductions in the natural deduction setting. See [7, 9, 10] for earlier work in this area.

The cut-reduction system has 32 rules: proving strong normalisation is tedious. The LPO method [6, 1] can be used, but confidence in its correct use may be low unless one is systematic. Here is one of the reduction rules, using a straightforward term notation for sequent derivations (C is for *Cut*; W is for the rule $L\vee$):

$$\begin{aligned} C(W(w,w_1.L_1,w_2.L_2),x.W(x,x'.L',x''.L'')) \quad \rightsquigarrow \quad & W(w,w_1.C(L_1,x.W(x,x'.C(W(w,w_1.L_1,w_2.L_2),x.L'),\\ & \qquad x''.C(W(w,w_1.L_1,w_2.L_2),x.L''))),\\ & \quad w_2.C(L_2,x.W(x,x'.C(W(w,w_1.L_1,w_2.L_2),x.L'),\\ & \qquad x''.C(W(w,w_1.L_1,w_2.L_2),x.L'')))) \end{aligned}$$

For the LPO method to be used, one must first (along lines to be found in [2]) remove the binders, obtaining

$$\begin{aligned} C(W(w,L_1,L_2),W(x,L',L'')) \quad \rightsquigarrow \quad & W(w,C(L_1,W(x,x'.C(W(w,L_1,L_2),L'),\\ & \qquad C(W(w,L_1,L_2),L''))),\\ & \quad C(L_2,W(x,C(W(w,L_1,L_2),L'),\\ & \qquad C(W(w,L_1,L_2),L'')))) \end{aligned}$$

and identify a suitable ordering $>$ on the constructors (here they are C and W) of what is now a first-order signature. The tedious part is then to show, using the recursive definition of the lexicographic path order $>_{lpo}$, that (for **each** of the 32 rules) the LHS $>_{lpo}$ RHS. The rules (for LPO ordering $>$ of terms $s,t,\ldots$) are as follows (where $\triangleright$ indicates the relation between a term

and each of its immediate subterms, $\triangleleft$ is the converse, and $>^{lex}$ is the lexicographic extension of $>$ to tuples, with associated rule $>_{lex}$):

$$\frac{\exists u \triangleleft s.\ u \geq t}{s > t}\ >_i \qquad\qquad \frac{s \gg t \quad \forall u \triangleleft t.\ s > u}{s > t}\ >_{ii}$$

$$\frac{f > g}{f(s_1, \ldots, s_m) \gg g(t_1, \ldots, t_n)}\ \gg_i \quad \frac{(s_1, \ldots, s_n) >^{lex} (t_1, \ldots, t_n)}{f(s_1, \ldots, s_n) \gg f(t_1, \ldots, t_n)}\ \gg_{ii}.$$

Rather than do this by hand, we wrote a small Prolog program to do it automatically. It can easily be adapted for other problems of the same kind; it will even generate LaTeX representations of the proofs, such as (with some bits removed for layout reasons):

$$\dfrac{\dfrac{C > p}{C(L, p(L', L'')) \gg p(C(L, L'), C(L, L''))}\ \gg_i \quad \dfrac{\dfrac{\dfrac{L' \geq L'}{p(L', L'') > L'}\ >_i}{(L, p(L', L'')) >^{lex} (L, L')}\ >_{lex}}{C(L, p(L', L'')) \gg C(L, L')}\ \gg_{ii} \quad \ldots \quad \ldots}{C(L, p(L', L'')) > C(L, L')}\ >_{ii} \quad \ldots}{C(L, p(L', L'')) > p(C(L, L'), C(L, L''))}\ >_{ii}$$

in which p is the constructor for pairs.. The bits indicated here by ... are similar, concluding that $C(L, p(L', L'')) > L$, that $C(L, p(L', L'')) > L'$ and that $C(L, p(L', L'')) > C(L, L'')$ respectively.)

Rather than writing a Prolog program, it is possible that we could have used a termination checker, such as TTT [5], or a strongly typed system such as Abella [4]. The Prolog route seemed to be easiest.

References

[1] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*, Cambridge University Press, 1998.

[2] Roel Bloo and Herman Geuvers. *Explicit Substitution On the Edge of Strong Normalization*, Theor. Comput. Sci., **211**, pp 375–395, 1999.

[3] Roy Dyckhoff. *Cut-elimination, Substitution and Normalisation*, for inclusion in *Dag Prawitz on Proofs and Meaning*, ed. H.Wansing, 2014 ?.

[4] Andrew Gacek. *The Abella interactive theorem prover (system description)*, Proceedings of IJCAR 2008, LNAI **5195**, Springer, pp 154–161, 2008.

[5] Nao Hirokawa and Aart Middeldorp. *Tyrolean Termination Tool*, Proc. of the 16th International Conference on Rewriting Techniques and Applications, LNCS 3467, pp. 175-184, 2005.

[6] Samuel Kamin and Jean-Jacques Lévy. *Two generalisations of the recursive path ordering*, University of Illinois at Urbana-Champaign. Unpublished ms, 1980.

[7] Garrel Pottinger. *Normalisation as a homomorphic image of cut-elimination*, Annals of Mathematical Logic **12**, pp 323–357, 1977.

[8] Anne Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*, 2nd ed, Cambridge, 2000.

[9] Christian Urban. *Revisiting Zucker's work on the correspondence between cut-elimination and normalisation*, in L.C. Pereira, E.H. Haeusler and V. de Paiva (eds.), *Advances in Natural Deduction. A Celebration of Dag Prawitz's Work*, Springer, Dordrecht, 21 pp, 2014.

[10] Jeff Zucker. *The correspondence between cut-elimination and normalisation*, Annals of Mathematical Logic **7**, pp 1–112, 1974.

Some ideas on bounded arithmetics for systems of monotone proofs

Anupam Das

October 25, 2013

One of the outstanding problems in proof complexity is to find nontrivial lower bounds on the size of proofs in Hilbert-Frege systems [2], or equivalently sequent calculi with cut. To this end researchers have proposed imposing various restrictions on proofs to simplify the problem and bridge the gap between these and weaker systems for which exponential lower bounds are known, such as the cut-free sequent calculus. For example, a well-studied direction has been to restrict the depth of cut-formulae in proofs, yielding bounded-depth and resolution systems, whose complexities are now well understood.[1]

Another direction is to consider *monotone proofs*, ones whose cut-formulae are free of negation. This is equivalent to formula-rewriting in the following system,

$$\mathsf{w}{\downarrow}\,\frac{\bot}{A} \quad , \quad \mathsf{w}{\uparrow}\,\frac{A}{\top} \quad , \quad \mathsf{c}{\downarrow}\,\frac{A \vee A}{A} \quad , \quad \mathsf{c}{\uparrow}\,\frac{A}{A \wedge A} \quad , \quad \mathsf{s}\,\frac{A \wedge [B \vee C]}{(A \wedge B) \vee C}$$

modulo associativity, commutativity and some basic unit equations. Atserias et al. have shown that the system of monotone proofs quasipolynomially simulate Hilbert-Frege systems [1], and further that superpolynomial lower bounds for monotone proofs imply superpolynomial lower bounds for Hilbert-Frege systems. To continue in this direction it is therefore pertinent to pursue further restrictions on monotone proofs, or to find normal forms of these proofs that might admit nontrivial lower bounds.

A natural notion of normal form one might consider is the class of monotone proofs where all $\uparrow$-rules occur before all $\downarrow$-rules, called *streamlined proofs* [7]; such proofs are equivalent to monotone sequent proofs where there are no cuts between descendants of structural steps [4]. Due to the development of local normalisation procedures for related deep inference systems, we gain a fine-grained analysis of the complexity of 'streamlining', and this has recently been used to obtain many positive complexity results for the system of streamlined proofs, for example a construction of quasipolynomial-size proofs of the propositional pigeonhole principle and a polynomial simulation of resolution systems [6] [5].

Theories of *bounded arithmetic* have proved useful in proof complexity, with arithmetic proofs serving as templates for uniform classes of small proofs in some associated propositional proof system, via appropriate translations. For example the theory $I\Delta_0$, Robinson's arithmetic augmented with induction on Δ_0-formulae, corresponds in this way to polynomial-size bounded-depth Hilbert-Frege proofs, by the Paris-Wilkie translation [8]. As an example of application, Pudlák has proved a version of Ramsey's theorem in this theory,[2] yielding quasipolynomial-size

[1] In particular, exponential lower bounds on proofs of the pigeonhole principle are known for these systems [9].

[2] In fact this proof operates in a slightly larger theory that assumes the totality of quasipolynomials, hence the slightly above polynomial bound.

bounded-depth proofs of certain propositional encodings of this theorem [3].

In this talk we present some ideas towards designing theories of bounded arithmetic corresponding to monotone and streamlined proofs. We incorporate a form of inductive definitions to simulate classes of propositional formulae of unbounded depth, and restrict mathematical induction to formulae where non-logical symbols occur in positive context to simulate monotonicity. One of the nice features of this approach is that the set of streamlined proofs can be recovered, under the given propositional translation, by simply insisting that all inductions are divide-and-conquer inductions. Proving this relies crucially on the aforementioned normalisation procedures from deep inference and recent results on their complexity.

As an application we use this theory to prove the correctness of merge-sort (as a sorting algorithm) and obtain as corollaries many basic counting arguments such as the generalised pigeonhole principle and the parity principle, yielding quasipolynomial-size streamlined proofs of their propositional encodings. We hope that such a theory might provide a framework to deliver further positive complexity results, in particular simulations, and provide intuitions towards finding lower bounds for streamlined proofs and their related deep inference systems.

References

[1] A.Atserias, N.Galesi, and P.Pudlák. Monotone simulations of non-monotone proofs. *Journal of Computer and System Sciences*, 65(4):626–638, 2002.

[2] M.L. Bonet, S.R. Buss, and T.Pitassi. Are there hard examples for frege systems? In P.Clote and J.Remmel, editors, *Feasible Mathematics II*, volume13 of *Progress in Computer Science and Applied Logic*, pages 30–56. Birkhäuser Boston, 1995.

[3] E.Börger, H.K. Büning, M.M. Richter, and W.Schönfeld, editors. *Computer Science Logic, 4th Workshop, CSL '90, Heidelberg, Germany, October 1-5, 1990, Proceedings*, volume 533 of *Lecture Notes in Computer Science*. Springer, 1991.

[4] K.Brünnler. Deep inference and its normal form of derivations. In A.Beckmann, U.Berger, B.Löwe, and J.V. Tucker, editors, *Computability in Europe 2006*, volume 3988 of *Lecture Notes in Computer Science*, pages 65–74. Springer-Verlag, July 2006. http://www.iam.unibe.ch/~kai/Papers/n.pdf.

[5] A.Das. Complexity of deep inference via atomic flows. In S.B. Cooper, A.Dawar, and B.Löwe, editors, *Computability in Europe*, volume 7318 of *Lecture Notes in Computer Science*, pages 139–150. Springer-Verlag, 2012. http://www.anupamdas.com/items/RelComp/RelComp.pdf.

[6] A.Das. The pigeonhole principle and related counting arguments in weak monotone systems. http://www.anupamdas.com/items/WeakMonProofsPHP/WeakMonProofsPHP.pdf, 2013.

[7] A.Guglielmi and T.Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9):1–36, 2008. http://www.lmcs-online.org/ojs/viewarticle.php?id=341.

[8] J.Paris and A.Wilkie. δ_0 sets and induction. *Open Days in Model Theory and Set Theory, W. Guzicki, W. Marek, A. Pelc, and C. Rauszer, eds*, pages 237–248, 1981.

[9] T.Pitassi, P.Beame, and R.Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3:97–140, 1993. 10.1007/BF01200117.

Proof Theory for Lattice-Ordered Groups*

George Metcalfe

Mathematical Institute
University of Bern, Switzerland

george.metcalfe@math.unibe.ch

Proof theory provides useful tools for tackling problems in algebra. In particular, Gentzen systems admitting cut-elimination have been used to establish decidability, amalgamation, and generation results for varieties of residuated lattices corresponding to substructural logics (see, e.g., [1, 2, 3]). However, for classes of algebras bearing some family resemblance to *groups* – e.g., lattice-ordered groups, MV-algebras, BL-algebras, and cancellative residuated lattices – the proof-theoretic approach has met so far only with limited success (see [4, 3]).

The aim of this talk is to introduce and exploit proof-theoretic methods for the class of *lattice-ordered groups* (or *ℓ-groups* for short): algebraic structures $(L, \wedge, \vee, \cdot, ^{-1}, 1)$ such that $(L, \wedge, \vee)$ is a lattice, $(L, ^{-1}, 1)$ is a group, and the group multiplication $\cdot$ preserves the order in both arguments; i.e., $a \leq b$ implies $ac \leq bc$ and $ca \leq cb$ for all $a, b, c \in L$. Commutative examples include the sets of reals, rationals, or integers with the usual total order and addition. Non-commutative examples are obtained by equipping the set $Aut(\Omega)$ of all order-preserving bijections on a linearly ordered set Ω with coordinatewise lattice operations, functional composition, and function inverse. In fact every ℓ-group embeds into such an ℓ-group [6], and this result has been used to show that the variety of ℓ-groups is generated by the automorphism ℓ-group of the real numbers [5] and that the equational theory of this variety is decidable [7].

This talk presents new syntactic proofs of these generation and decidability results, refining the latter to obtain a new co-NP completeness result. Gentzen systems for ℓ-groups with and without the cut rule are defined in a one-sided hypersequent framework where hypersequents correspond to disjunctions of group terms. A crucial role in the completeness proofs for these systems is played by the elimination of applications of a certain "resolution-like" rule.

References

[1] N. Galatos, P. Jipsen, T. Kowalski, and H. Ono, *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Elsevier, 2007.

[2] A. Ciabattoni, N. Galatos, and K. Terui, "From axioms to analytic rules in nonclassical logics," in *Proceedings of LICS 2008*, pp. 229–240, 2008.

[3] G. Metcalfe, N. Olivetti, and D. Gabbay, *Proof Theory for Fuzzy Logics*, vol. 36 of *Applied Logic*. Springer, 2009.

[4] G. Metcalfe, N. Olivetti, and D. Gabbay, "Sequent and hypersequent calculi for abelian and Łukasiewicz logics," *ACM Transactions on Computational Logic*, vol. 6, no. 3, pp. 578–613, 2005.

[5] W. C. Holland, "The largest proper variety of lattice ordered groups," *Proc. Amer. Math. Soc.*, vol. 57, pp. 25–28, 1976.

[6] W. C. Holland, "The lattice-ordered group of automorphisms of an ordered set," *Michigan Math. J.*, vol. 10, pp. 399–408, 1963.

[7] W. C. Holland and S. H. McCleary, "Solvability of the word problem in free lattice-ordered groups," *Houston Journal of Mathematics*, vol. 5, no. 1, pp. 99–105, 1979.

*Based on joint work with Nikolaos Galatos.

First-order proofs without syntax

Dominic J. D. Hughes
Chief Scientist, concept.io

I shall present a reformulation of first-order logic in which proofs are combinatorial, rather than syntactic. A combinatorial proof satisfies graph-theoretic conditions which can be verified in polynomial time, and there is a polynomial-time translation from Gentzen sequent calculus proofs to combinatorial proofs, identifying many proofs. This work extends the propositional combinatorial proofs presented in 'Proofs without syntax' [Annals of Math., 2006].

Link: `http://boole.stanford.edu/~dominic/papers/pws/`

Expansion Trees with Cut

Stefan Hetzl and Daniel Weller

October 25, 2013

Herbrand's theorem [Her30, Bus95], one of the most fundamental insights of logic, characterizes the validity of a formula in classical first-order logic by the existence of a propositional tautology composed of instances of that formula.

From the syntactic point of view this theorem induces a way of describing proofs: by recording which instances have been picked for which quantifiers we obtain a description of a proof up to its propositional part, a part we often want to abstract from. An example for a formalism that carries out this abstraction are Herbrand proofs [Bus95]. This generalizes nicely to most classical systems with quantifiers, for example to simple type theory as in the expansion tree proofs of [Mil87]. Such formalisms are compact and useful proof certificates in many situations; they are for example produced naturally by methods of automated deduction such as instantiation-based reasoning [Kor09].

These formalisms consider only instances of the formula that has been proved and hence are *analytic* proof formalisms (corresponding to cut-free proofs in the sequent calculus). Considering an expansion tree to be a compact representation of a proof, it is thus natural to ask about the possibility of extending this kind of representation to *non-analytic* proofs (corresponding to proofs with cut in the sequent calculus).

Two instance-based proof formalisms incorporating a notion of cut have recently been proposed: proof forests [Hei10] and Herbrand nets [McK13]. While proof forests are motivated by the game semantics for classical arithmetic of [Coq95], Herbrand nets are based on methods for proof nets [Gir87]. These two formalisms share a number of properties: both of them work in a graphical notation for proofs, both work on prenex formulas only, for both weak but no strong normalization results are known.

In this talk we present a new approach which works directly in the formalism of expansion tree proofs and hence naturally extends the existing literature in this tradition. As in [Hei10, McK13] we define a cut-elimination procedure and prove it weakly normalizing but in contrast to [Hei10, McK13] we also treat non-prenex formulas, therefore avoiding the distortion of the intuitive meaning of a formula by prenexification.

We describe expansion trees with cuts for non-prenex end-sequents and cuts, including their correctness criterion and how to translate from and to sequent calculus. We describe natural cut-reduction steps and show that they are weakly normalizing. A technical key for proving weak normalization is to use methods

of Hilbert's epsilon-calculus which is a formalism for representing non-analytic first-order proofs modulo propositional logic. Finally, we consider the question whether our cut-reduction rules are strongly normalizing: we don't know (yet), but a counterexample from the setting of proof forests is no counterexample in our setting.

References

[Bus95] Samuel R. Buss. On Herbrand's Theorem. In *Logic and Computational Complexity*, volume 960 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 1995.

[Coq95] Thierry Coquand. A semantics of evidence for classical arithmetic. *Journal of Symbolic Logic*, 60(1):325–337, 1995.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[Hei10] Willem Heijltjes. Classical proof forestry. *Annals of Pure and Applied Logic*, 161(11):1346–1366, 2010.

[Her30] Jacques Herbrand. *Recherches sur la théorie de la démonstration*. PhD thesis, Université de Paris, 1930.

[Kor09] Konstantin Korovin. Instantiation-Based Automated Reasoning: From Theory to Practice. In Renate A. Schmidt, editor, *22nd International Conference on Automated Deduction (CADE)*, volume 5663 of *Lecture Notes in Computer Science*, pages 163–166. Springer, 2009.

[McK13] Richard McKinley. Proof nets for Herbrand's Theorem. *ACM Transactions on Computational Logic*, 14(1), 2013.

[Mil87] Dale Miller. A Compact Representation of Proofs. *Studia Logica*, 46(4):347–370, 1987.

Interpolation in finitely-valued first-order logics

Matthias Baaz
Vienna University of Technology

We show, that under minimal assumptions interpolation in propositional finitely-valued logics is decidable and that first-order interpolation coincides with propositional interpolation. This provides an uniform proof of first-order interpolation for finite Kripke frames with constant domains, which are known to admit propositional interpolation. We discuss the non-constant domain case.

Guessing induction formulas for proofs of universal statements

Sebastian Eberhard Stefan Hetzl

Institut für Diskrete Mathematik und Geometrie
TU Wien, Wiedner Hauptstrasse 8-10, A-1040 Wien, Austria
{sebastian.eberhard,stefan.hetzl}@tuwien.ac.at

Induction is the most prominent rule to prove non-trivial universal statements in theories about numbers. Nevertheless, the task of finding induction formulas useful in a proof of such a statement is difficult. For simple sentences like $(\forall x)\big((x+x)+x=x+(x+x)\big)$ automatic theorem provers typically are only able to cut-free prove any instance of the universal property in a simple universal base theory without induction. This gives rise to a Herbrand disjunction $H(n)$ for each natural number n. Nevertheless, using e.g. the induction formula $(x+x)+y=y+(x+x)$ the universal property is proved by an easy induction.

In our talk, we present a strategy to guess Π_1 induction formulas F from a finite set of Herbrand disjunctions $H(n)$ for instances of a universal statement. A guess yielding F is correct if a derivation of a fixed form containing induction over F is a proof of the universal statement. We cannot give a guarantee that a correct induction formula is produced by our strategy since in the base theory extended by Π_1 induction the universal statement might still not be derivable. In addition, because of efficiency reasons, some sensible guesses have to be ignored.

Nevertheless, the strategy delivers correct induction formulas in natural test cases, and should be efficiently implementable.

In our talk, we explain several steps of the strategy in detail. The interpretation of Herbrand disjunctions as languages of terms and the analysis of their decomposition grammars is crucial. Here, we rely on earlier results in the second author's [1].

We hope that further refinements of the presented strategy help to overcome the weakness of automatic theorem provers in using induction.

References

[1] HETZL, STEFAN AND LEITSCH, ALEXANDER AND WELLER, DANIEL, *Towards Algorithmic Cut-Introduction*, ***Logic for Programming, Artificial Intelligence and Reasoning (LPAR-18)***, Lecture Notes in Computer Science, vol. 7180, pages 228–242, 2012.

Power and Limits of Structural Rules

Agata Ciabattoni
Vienna University of Technology, A–1040 Vienna, Austria
agata@logic.at

1 Abstract

Non-classical logics are often defined by adding suitable properties to basic systems. We present a method to extract structural rules in various Gentzen-style formalisms out of such properties. The method, that works for large classes of Hilbert axioms and Frame conditions, allows for the introduction of analytic calculi for a wide range of non-classical logics. It applies to various formalisms, including hypersequent calculus, labelled deductive systems and display calculus, and sheds some light on the expressive power of their structural rules.

References

[1] A. Ciabattoni, R. Ramanayake. Structural extensions of display calculi: a general recipe. Proceedings of WOLLIC 2013.

[2] A. Ciabattoni, P. Maffezioli and L. Spendier. Hypersequent and Labelled Calculi for Intermediate Logics, Proceedings of TABLEAUX 2013.

[3] A. Ciabattoni, N. Galatos and K. Terui. Algebraic proof theory for substructural logics: cut-elimination and completions, Annals of Pure and Applied Logic, 163(3): 266-290, 2012.

[4] A. Ciabattoni, N. Galatos and K. Terui. From axioms to analytic rules in nonclassical logics. Proceedings of LICS 2008.

Linear Session Types for Solos

Nicolas Guenot

IT University of Copenhagen
ngue@itu.dk

While the logical approach to sequential computation is founded on the relation between λ-terms and proofs in intuitionistic natural deduction, such *Curry-Howard correspondence* has long been elusive in the case of concurrent computation. In this setting, a natural candidate as computational model is the π-calculus, which offers a high expressivity wrapped in a simple syntax. The nature of the typing needed to describe which subset of π-terms is well-behaved was less obvious, until the notion of *session types* [Hon93] was introduced, partially inspired by linear logic. Based on this choice of ingredients, the correspondence could defined between session-typed π-terms and proofs in sequent calculi for linear logic. This was described by Caires and Pfenning [CP10] in the case of intuitionistic linear logic, and Wadler [Wad12] in the case of classical linear logic.

This correspondence allows to ensure that well-typed π-terms are well-behaved and respect the protocol described by their session type, but it suffers from technical problems, due to the flexibility of the syntax in π-calculi and to a certain mismatch between terms and proofs. We argue here that although session types provide the right abstraction to be matched with linear formulas, the choice of the π-calculus as computational model disallows a perfect matching of proofs and terms. In order to improve this matching, we introduce a variant of the *solos* fragment of the fusion calculus [LV03], where the use of names allows for a linear discipline. The decisive feature of solos is that they disallow explicit sequentialisation, by avoiding the prefix operator of the π-calculus. This corresponds to the possibility of permuting rule instances in the sequent calculus, where sequentiality is always the result of a *causality* observed when a rule depends on another.

The potential linearity of names is coupled to a mechanism for sequentialisation: a solo with subject x can provide the information that a name y is the *continuation* of the session it embodies. Therefore, this linearity is not a limitation, but rather a useful *bookkeeping* discipline, as it allows to enforce that a given interaction will happen before another. The syntax we use departs on several points from the one used in the standard solos calculus.

$$\begin{array}{rcl} P,Q & ::= & \mathbf{0} \mid P \mid Q \mid x \mid xy \triangleright z \mid [x \leftrightarrow y] \mid \overline{x}\mathtt{l} \triangleright y \mid !xy.P \\ & & \mid (x)P \mid \overline{x} \mid \overline{x}y \triangleright z \mid x \triangleright y[P,Q] \mid \overline{x}\mathtt{r} \triangleright y \mid !\overline{x}y \end{array}$$

This syntax allows to handle monadic input or output, as well as binary guarded choice and selection, and persistent servers with connection requests. A syntactic congruence on terms is then defined, that extends the one usually considered. The reduction rules defining the dynamic behaviour of the calculus are then similar to the reductions used by Caires and Pfenning, adapted to the setting of solos. Unlike

most process calculi, our calculus allows in general reduction under any context, reflecting the fact that cut elimination can be performed in any order. For example, some of the reduction rules are:

$$\begin{array}{rcll} (y)([x \leftrightarrow y] \mid P) & \longrightarrow & P\{x/y\} & (x \neq y) \\ (uv)((\overline{x}y \triangleright z \mid P) \mid (xu \triangleright v \mid Q)) & \longrightarrow & (P \mid Q)\{y/u\}\{z/v\} & (u, v \neq y \wedge u, v \neq z) \\ !xy.P \mid (!\overline{x}z \mid Q) & \longrightarrow & !xy.P \mid (P\{z/y\} \mid Q) & \end{array}$$

The type system that we define for this calculus is based on the sequent calculus for intuitionistic linear logic It is a variant of the system of Caires and Pfenning, although the treatment of scope is modified to allow more flexibility. Some basic typing rules are:

$$\frac{\mathbb{S}, x \rangle \Gamma \vdash P :: w : A}{\mathbb{S} \rangle \Gamma \vdash (x)P :: w : A} \qquad \frac{\mathbb{S} \rangle\, !\Psi, \Gamma \vdash P :: y : A \qquad \mathbb{R} \rangle\, !\Psi, \Delta, z : B \vdash Q :: w : C}{\mathbb{S}, \mathbb{R}, y, z \rangle\, !\Psi, \Gamma, \Delta, x : A \multimap B \vdash \overline{x}y \triangleright z \mid (P \mid Q) :: w : C}$$

The correspondence thus obtained between terms and proofs is more faithful to the structure of the sequent calculus than the previous proposals. Moreover, this leads us to consider some solos as *linear forwards* [GLW03] in an extended syntax, allowing for an interpretation of the $\otimes$ connective:

$$\frac{\mathbb{S} \rangle\, \Gamma, y : A, z : B \vdash P :: w : C}{\mathbb{S}, y, z \rangle\, \Gamma, x : A \otimes B \vdash [x \multimap yz] \mid P :: w : C}$$

that is much more intuitive than the asymmetric interpretation given by Caires and Pfenning, or Wadler in the classical setting. This sheds some light on the concurrent interpretation of cut elimination in linear logic, and invites to recast other parts of the syntax into forwarders, which may provide a convenient syntax for proofs in both classical and intuitionistic variants of linear logic.

References

[CP10] Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In P. Gastin and F. Laroussinie, editors, *CONCUR'10*, volume 6269 of *LNCS*, pages 222–236, 2010.

[GLW03] Philippa Gardner, Cosimo Laneve, and Lucian Wischik. Linear forwarders. In *CONCUR'03*, volume 2761 of *LNCS*, pages 415–430, 2003.

[Hon93] Kohei Honda. Types for dyadic interaction. In E. Best, editor, *CONCUR'93*, volume 715 of *LNCS*, pages 509–523, 1993.

[LV03] Cosimo Laneve and Björn Victor. Solos in concert. *Mathematical Structures in Computer Science*, 13(5):657–683, 2003.

[Wad12] Philip Wadler. Propositions as sessions. In P. Thiemann and R. B. Findler, editors, *ICFP'12*, pages 273–286. ACM, 2012.

Meta-Reasoning in the Concurrent Logical Framework CLF

Iliano Cervesato Jorge Luis Sacchini
Carnegie Mellon University
iliano@cmu.edu sacchini@qatar.cmu.edu

The Concurrent Logical Framework CLF [1, 5] is an extension of the Edinburgh Logical Framework (LF) [3] designed for specifying concurrent and distributed systems. CLF extends LF with synchronous and asynchronous linear types, and the lax modality from lax logic to encapsulate concurrent effects.[1] The lax modality can be used to naturally encode concurrent traces, i.e. sequence of computations where independent steps can be permuted. Kinds and types in CLF are defined by the following grammar:

$$K ::= \mathsf{type} \mid \Pi!x{:}T.K \qquad \text{(Kinds)}$$
$$T ::= \Pi!x{:}T.T \mid T \multimap T \mid A \mid \{\Delta\} \qquad \text{(Types)}$$
$$\Delta ::= \cdot \mid \Delta, !x{:}T \mid \Delta, x{:}T \qquad \text{(Contexts)}$$

where A denotes an atomic type family and $\{\cdot\}$ is the lax modality. Contexts are sequences of declarations of persistent and linear variables.

Concurrent traces are the introduction form for the lax modality. They are defined by the following grammar:

$$\varepsilon ::= \diamond \mid \{\Delta\}{\leftarrow}P \mid \varepsilon_1; \varepsilon_2$$

where $\diamond$ is the empty trace, $\{\Delta\}{\leftarrow}P$ is an atomic computation step, and $\varepsilon_1; \varepsilon_2$ defines trace composition. The typing rules for traces have the form $\Delta \vdash \varepsilon : \Delta'$, which can be read as a context transformation (or rewriting) from Δ to Δ'. Furthermore, traces are endowed with a notion of equality that allows permutation of *independent steps*. Two steps are independent if they produce and consume different sets of variables. This notion of trace equality can be used to encode concurrency and distribution.

CLF has been successfully used for specifying a wide variety of systems such as semantics of concurrent programming languages, the π-calculus, and voting protocols, among others. However, CLF lacks the expressive power necessary for specifying and proving meta-theoretical properties about such systems (e.g. type preservation, or correctness of program transformations). The main reason is that traces are not first-class values, and therefore they cannot be analyzed or manipulated.

In recent work [2], we proposed an extension of LF, called Meta-CLF, designed to reason about a CLF specification. Meta-CLF extends LF with quantification over contexts and *trace types* to represent CLF traces. For example, in Meta-CLF we can defines types of the form

$$\Pi\Delta_0.\, \Pi\Delta_1.\, \{\Delta_0\}\, \Sigma\, \{\Delta_1\} \to \{\Delta_0\}\, \Sigma'\, \{\Delta_1\} \to \mathsf{type}$$

where $\{\Delta_0\}\, \Sigma\, \{\Delta_1\}$ is the type of CLF traces from context Δ_0 to Δ_1 that use steps defined in the (CLF) signature Σ. Types of this form can be used, for example, to encode a relation between two program executions.

In [2] we showed that Meta-CLF can be used to encode safety properties (i.e. type preservation and progress) for a substructural operational semantics (SSOS) [4] of a simple programming language. However, many issues remain to be solved. We have not yet addressed the meta-theory of Meta-CLF itself. In particular, we have not studied coverage and termination checking, which are essential properties for any logical framework. Coverage for trace types is a difficult problem due to the nature of trace equality that allows reordering of independent steps.

In this proposed talk, we will look at the design choices we made in Meta-CLF and how it can be used to encode safety proofs for SSOS specifications of programming languages written in CLF. We will also discuss the remaining challenges and possible solutions.

[1] CLF also features affine types, which we omit here for space reasons.

References

[1] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. A Concurrent Logical Framework II: Examples and Applications. Technical Report CMU-CS-02-102, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 2003.

[2] Iliano Cervesato and Jorge Luis Sacchini. Towards meta-reasoning in the concurrent logical framework CLF. In Johannes Borgström and Bas Luttik, editors, *EXPRESS/SOS*, volume 120 of *EPTCS*, 2013.

[3] Robert Harper, Furio Honsell, and Gordon D. Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, 1993.

[4] Frank Pfenning. Substructural operational semantics and linear destination-passing style. In W.-N. Chin, editor, *Proceedings of the 2nd Asian Symposium on Programming Languages and Systems (APLAS'04)*, page 196, Taipei, Taiwan, November 2004. Springer-Verlag LNCS 3302. Abstract of invited talk.

[5] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A Concurrent Logical Framework I: Judgments and Properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, May 2003.

Toward a Theory of Contexts of Assumptions in Logical Frameworks

Amy Felty[1], Alberto Momigliano[2], and Brigitte Pientka[3]

[1] School of Electrical Engineering and Computer Science, University of Ottawa, Canada, afelty@eecs.uottawa.ca
[2] Dipartimento di Informatica, Università degli Studi di Milano, Italy, momigliano@di.unimi.it
[3] School of Computer Science, McGill University, Montreal, Canada, bpientka@cs.mcgill.ca

In the beginning Gentzen created natural deduction, but then he switched to the sequent calculus in order to sort out the meta-theory [3]. Something similar happened to logical frameworks supporting higher-order abstract syntax (HOAS): first Edinburgh LF adopted Martin-Löf's parametric-hypothetical judgments to encode object logics in such a way that contexts were left *implicit* [5]. Later on, Twelf [9] had to provide some characterization of contexts (regular worlds) to verify the meta-theory of those very object logics. The same applies to λProlog [6] vs. Abella [2] and Hybrid [1] and, in a more principled way, to Beluga [7].

It is fair to say that, prior to Girard, proof-theory had been quite oblivious to what contexts look like. Even sub-structural logics view a context of assumptions as a flat collection of formulas $A_1, A_2, \ldots, A_n$ listing its elements separated by commas [4]. However, this is inadequate, especially if we aim to mechanize this matter, as it ignores that assumptions come in *blocks*. Consider as an object logic the type checking rules for the polymorphic lambda-calculus:

$$\frac{\overline{x\ \mathsf{term}}\ tm_x \qquad \overline{x : A}\ of_v}{\begin{array}{c}\vdots\\ M : B\end{array}}{\scriptstyle\ of_l^{tm_x, of_v}}\ \frac{}{(\mathsf{lam}\, x.\, M) : (\mathsf{arr} A\, B)} \qquad \frac{\overline{\alpha\ \mathsf{tp}}\ tp_v}{\begin{array}{c}\vdots\\ M : A\end{array}}{\scriptstyle\ of_{tl}^{tp_v}}\ \frac{}{(\mathsf{tlam}\, \alpha.\, M) : (\mathsf{all}\, \alpha.\, A)}$$

$$\frac{M_1 : (\mathsf{arr} A\, B) \quad M_2 : A}{(\mathsf{app}\ M_1\ M_2) : B}\ of_a \qquad \frac{M : (\mathsf{all}\, \alpha.\, A) \quad B\ \mathsf{tp}}{(\mathsf{tapp}\ M\ A) : [B/\alpha]A}\ of_{ta}$$

We propose to view contexts as *structured sequences* of declarations D where a declaration is a block of individual atomic assumptions separated by ';'.

$$\begin{array}{rl} \text{Atom} & A \\ \text{Block of declarations} & D ::= A \mid D; A \\ \text{Context} & \Gamma ::= \cdot \mid \Gamma, D \\ \text{Schema} & S ::= D_s \mid D_s + S \end{array}$$

A schema classifies contexts and consists of declarations D_s that may be more general than those occurring in a concrete context having schema S. This yields for the above example:

$$\begin{array}{l} \Gamma ::= \cdot \mid \Gamma, (x\ \mathsf{term}; x{:}A) \mid \Gamma, \alpha\ \mathsf{tp} \\ S ::= \alpha\ \mathsf{tp} + (x\ \mathsf{term}; x{:}A) \end{array}$$

where, e.g., the context $\alpha_1\ \mathsf{tp},\ (x_1\ \mathsf{term}; x_1{:}\mathsf{nat}),\ (x_2\ \mathsf{term}; x_2{:}\alpha_1)$ has schema S.

Since contexts are structured sequences, they admit structural *properties* on the level of sequences (for example by adding a new declaration) as well as inside a block of declarations (for example by adding an element to an existing declaration). We distinguish also between structural properties of a *concrete* context and structural properties of *all* contexts of a given schema. We give a unified treatment of all such weakening/strengthening/exchange re-arrangements, by introducing total operations rm and perm that remove an element of a declaration, and permute elements within a declaration. For example, declaration weakening can be seen as:

$$\frac{\Gamma, \mathsf{rm}_A(D), \Gamma' \vdash J}{\Gamma, D, \Gamma' \vdash J}\ d\text{-}wk$$

Suppose now that we want to prove in a logical framework some meta-theorem involving different contexts, say "if $\Gamma_1 \vdash J_1$ then $\Gamma_2 \vdash J_2$", for Γ_i of schema S_i. HOAS-based logical frameworks have so far pursued two apparently different options:

(G) We reinterpret the statement in a *generalized context* containing all the relevant assumptions—we call this the *generalized context* approach, as taken in Twelf, Delphin [8] and Beluga—and prove "if $\Gamma_1 \cup \Gamma_2 \vdash J_1$ then $\Gamma_1 \cup \Gamma_2 \vdash J_2$", where "$\cup$" denotes the *join* of the two contexts.

(R) We state how two (or more) contexts are *related*—we call this the *context relations* approach. The statement becomes therefore "if $\Gamma_1 \sim \Gamma_2$ and $\Gamma_1 \vdash J_1$ then $\Gamma_2 \vdash J_2$", with an explicit and typically inductive definition of this relation. This approach is taken in Abella and Hybrid.

If we had a common grounding of both approaches, this would pave the way toward moving proofs from one system to another, in particular breaking the type/proof theory barrier [10]. It turns out, roughly, that a context relation can be seen as the graph of one or more appropriate rm operation on a generalized context. Further, if we take the above join metaphor seriously, we can organize declarations and contexts in a *semi-lattice*, where $x \preceq y$ holds iff x can be reached from y by some rm operation. A generalized context will indeed be the join of two contexts and context relations can be identified by navigating the lattice starting from the join of the to-be-related contexts. Finally, we aim to use the lattice structure to give a declarative account of theorem *promotion/demotion* (known in the Twelf lingo as "subsumption"), where a statement proven in a certain context can be used in a "related" one. We may formulate subsumption rules akin to upward and downward casting over the lattice order, i.e. promotion would look like:

$$\frac{\Gamma' \vdash J \qquad \Gamma \preceq \Gamma'}{\Gamma \vdash J}\ \mathit{prom}$$

This work also has a practical outcome in our ongoing work designing *ORBI* (Open challenge problem Repository for systems supporting reasoning with BInders), a repository for sharing benchmark problems and their solutions for HOAS-based systems, in the spirit of libraries such as TPTP [11].

References

[1] Amy P. Felty and Alberto Momigliano. Hybrid: A definitional two-level approach to reasoning with higher-order abstract syntax. *Journal of Automated Reasoning*, 48(1):43–105, 2012.

[2] Andrew Gacek. The Abella interactive theorem prover (system description). In *4th International Joint Conference on Automated Reasoning*, volume 5195 of *Lecture Notes in Computer Science*, pages 154–161. Springer, 2008.

[3] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969.

[4] J.-Y. Girard, Y. Lafont, and P. Tayor. *Proofs and Types*. Cambridge University Press, 1990.

[5] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[6] Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, 2012.

[7] Brigitte Pientka and Joshua Dunfield. Beluga:a framework for programming and reasoning with deductive systems (system description). In *5th International Joint Conference on Automated Reasoning*, volume 6173 of *Lecture Notes in Computer Science*, pages 15–21. Springer, 2010.

[8] Adam B. Poswolsky and Carsten Schürmann. Practical programming with higher-order encodings and dependent types. In *17th European Symposium on Programming*, volume 4960 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2008.

[9] Carsten Schürmann. The Twelf proof assistant. In *22nd International Conference on Theorem Proving in Higher Order Logics*, volume 567 of *Lecture Notes in Computer Science*, pages 79–83. Springer, 2009.

[10] Zachary Snow, David Baelde, and Gopalan Nadathur. A meta-programming approach to realizing dependently typed logic programming. In *12th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pages 187–198. ACM, 2010.

[11] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

A Semantics for Proof Evidence

Zakaria Chihani, Dale Miller, and Fabien Renaud
INRIA and LIX, Ecole Polytechnique, Palaiseau, France

October 1, 2013

Theorem provers are generally complex systems that search for proofs using some combination of automatic and interactive tools. Given their nature, it is important for them to be formally correct. Given their complexity, they are extremely hard to certify as such. One can arrange, however, for provers to be "certifying" in the sense that they can be built to output evidence of the proofs they achieved and then employ an independent proof checker to check them. Some proof systems, such as Coq, use this certifying approach by employing a trusted kernel to check proofs built outside the kernel [2].

In the ProofCert project, we are exploring to what extent we might be able to accommodate a wide range of proof evidence from various provers and then to have independent checkers certify such proof evidence. We have identified three different stages to achieving such a proof checking scheme [4].

1. The implementers of theorem provers must describe their proof evidence in some textual form. Presumably, such documents are roughly the result of "pretty printing" the evidence that they have collected. For example, the implementer of a resolution prover might output a numbered list of clauses as well as a list of triples (indicating which two clauses resolve to form a third clause). Such documents will be called *proof certificates*. One expects that there will be a great many kinds of proof certificate formats that are ultimately created and used.

2. A general framework for defining the semantics of proof evidence must be designed. In such a framework, the format and structure of proof certificates would be given a clear and precise semantics. Such a semantic framework will need to be sufficiently low-level so as to capture the essence of a proof and general enough to accommodate a wide range of proof systems. In the resolution prover example, the relationship of "resolvent" must be defined in terms of more basic inference rules of proof.

3. Trusted *proof checkers* must be implemented to execute the semantic descriptions of proof certificates. We need to be able to trust that a successful execution of the proof checker on a given certificate implies that the certificate does, in fact, elaborate into a recognized formal proof.

We shall show in this talk how focused sequent calculus proof systems, which are now available for linear, intuitionistic, and classical logics [1, 5, 6], can be used to address the second of these stages: i.e., they provide a flexible framework for defining the "semantics of proof evidence".

A simple analogy in the area of programming languages is worth pointing out for this three stage organization. (1) There are many kinds of programming languages and researchers are routinely designing new ones. (2) In order to define such programming languages precisely (for, say, mathematical treatment or for implementations), the semantics of such programming languages need to be given: a popular form of such semantic specifications is Structural Operational Semantics (SOS) [8]. (3) Finally, compliant interpreters and compilers for a given programming language can be built based on such semantic descriptions.

This analogy can be pushed an additional step. Since an SOS specification is given as a set of simple inference rules, a general purpose interpreters for SOS specifications can be given using logic programming languages [3, 7]. Similarly, since our evidence of proof is based on the generation of sequent calculus proofs, logic programming (particularly the more expressive and abstract form available in λProlog [7]) can be used to provide a reference interpreter for checking proof certificates.

References

[1] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992.

[2] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer, 2004.

[3] P. Borras, D. Clément, Th. Despeyroux, J. Incerpi, G. Kahn, B. Lang, and V. Pascual. Centaur: the system. In *Third Annual Symposium on Software Development Environments (SDE3)*, pages 14–24, Boston, 1988.

[4] Zakaria Chihani, Dale Miller, and Fabien Renaud. Foundational proof certificates in first-order logic. In Maria Paola Bonacina, editor, *CADE 24: Conference on Automated Deduction 2013*, LNAI 7898, pages 162–177, 2013.

[5] Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46), 2009.

[6] Chuck Liang and Dale Miller. A focused approach to combining logics. *Annals of Pure and Applied Logic*, 162(9):679–697, 2011.

[7] Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, June 2012.

[8] Gordon Plotkin. A structural approach to operational semantics. DAIMI FN-19, Aarhus University, Aarhus, Denmark, September 1981.

Pure Type Systems revisited

Herman Geuvers
Radboud University Nijmegen

Pure Type Systems (PTS) aim at forming a generic basis for various type theories. It captures a variety of systems (simple types, polymorphic types, dependent types, higher order types) but does not include e.g. inductive types, type inclusion and identity types.

Since PTSs have been introduced, various properties of them have been proved and variations on them have been studied. At the same time, various questions have remained unanswered. In the talk, we will overview some known "classic" results, some recent developments and some open problems. Topics we will treat are:

- type checking algorithms and syntax directed systems
- context free presentation
- explicit substitution
- the problem with adding eta
- the power of the conversion rule and making conversion explicit
- weak normalization and strong normalization
- inconsistent Pure Type Systems and the looping combinator

References

[1] F. van Doorn, H. Geuvers F. Wiedijk, Explicit Convertibility Proofs in Pure Type Systems, LFMTP 2013

[2] H. Geuvers, R. Krebbers, J. McKinna and F. Wiedijk, Pure Type Systems without Explicit Contexts, LFMTP 2010, EPTCS 34, pages 53-67.

[3] H. Barendregt and H. Geuvers, Proof Assistants using Dependent Type Systems, Chapter 18 of the Handbook of Automated Reasoning (Vol 2), Elsevier 2001, pp. 1149 – 1238.

[4] H. Geuvers and B. Werner, On the Church-Rosser property for Expressive Type Systems and its Consequences for their Metatheoretic Study, in LICS 1994, pp 320–329.

Type isomorphisms in presence of strong sums: axiomatizability, practical decidability and subtyping

Danko Ilik

October 9, 2013

Abstract

Disjoint unions are a basic artifact of programming and reasoning. In the context of typed lambda calculus, they are called (strong) sum types, and their Curry-Howard counterpart on the side of Logic is disjunction. Indeed, logical disjunction (more generally, existential quantification) is the feature that distinguishes intuitionistic from classical logic: while in classical logic, disjunction is definable from negative connectives, its true meaning is not preserved, since classical logic does not have the property that from a closed proof of (A or B) one can either prove A or prove B.

It is then a strange fact that in spite of omnipresence and importance of sums, there are still open question with respect to their full beta-eta-equality, or equivalently, with respect to characterizing type isomorphisms in their presence. For example, while we do know that the beta-eta-equality relation is not finitely axiomatizable, we do not know whether it is axiomatizable, and, if so, how. We do not know whether there is an efficient (sub-exponential) algorithm to decide type isomorphisms, and, as a matter of fact, I am not aware of any implementation of a complete algorithm deciding them.

These problems have practical implications, as well. For example, modern proof assistants based on constructive type theories, like Agda and Coq, allow the user to develop proofs modulo eta-equality for pi-types, but stop short of supporting eta-equality for inductive types (disjoint sums, in particular).

In this talk, we will revise the problems around isomorphisms in presence of sums, as well as the well known connection to the work around Tarski's High School Algebra Problem in Logic. We will show that there are still things to be learned from this connection on the typed lambda calculus side. We will also show that one can obtain a non-trivial well-founded (decidable) subtyping relation on types (in presence on sums) by using constructive proofs of Kruskal's Theorem.

Proofs in monoidal closed bifibrations

Noam Zeilberger

The concept of refinement in type theory is a way of reconciling the "intrinsic" and the "extrinsic" meanings of types (also known as "types à la Church" and "types à la Curry"). In a recent paper with Paul-André Melliès [1], we looked at this idea more carefully and settled on the simple conclusion that the type-theoretic notion of "type refinement system" may be identified with the category-theoretic notion of "functor". We recall here the basic correspondence (illustrated graphically in Figure 1):

Let a functor $p : \mathbb{E} \to \mathbb{I}$ be fixed.

Definition 0.1. *We say that an object $S \in \mathbb{E}$* **refines** *an object $A \in \mathbb{I}$, written $S \sqsubset A$, if $p(S) = A$.*

Definition 0.2. *Suppose given an $\mathbb{I}$-map $f : A \to B$ and two $\mathbb{E}$-objects $S \sqsubset A$ and $T \sqsubset B$. Such a triple of information is called a* **typing judgment***, which we notate by writing f below an arrow from S to T:*

$$S \underset{f}{\longrightarrow} T$$

In the special case where $A = B$ and $f = -_A$, we use the abbreviated notation

$$S \longrightarrow T \stackrel{def}{=} S \underset{-_A}{\longrightarrow} T$$

which we call a **subtyping judgment***.*

Definition 0.3. *A* **typing derivation** *for a (sub)typing judgment $S \underset{f}{\longrightarrow} T$ is an $\mathbb{E}$-map $\alpha : S \to T$ such that $p(\alpha) = f$. We notate this concisely by placing α over the judgment:*

$$S \overset{\alpha}{\underset{f}{\longrightarrow}} T$$

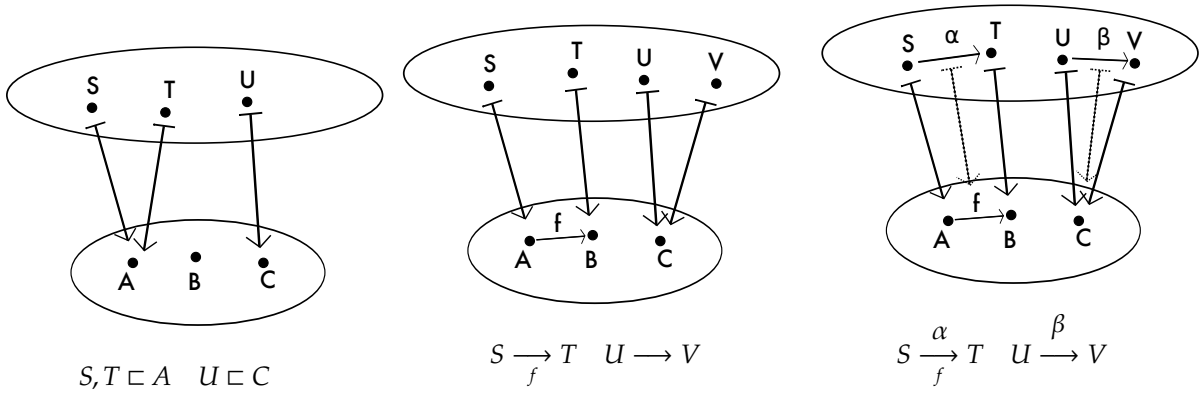
A (sub)typing judgment is said to be **derivable** *if there exists a typing derivation for that judgment. We notate this with a turnstile to the left of the judgment:*

$$\vdash S \underset{f}{\longrightarrow} T$$

In the talk, I would like to discuss some ideas in and around the paper and these definitions, in particular:

1. How this generalization of the traditional analogy (type system ~ category) gives a rigorous mathematical status to many workaday concepts from proof theory, side-stepping the usual very *formal* approach which relies heavily on inductive definitions.

2. The rich logic that emerges by combining the basic connectives of monoidal closed categories (i.e., product and implication) with the basic connectives of a bifibration (inverse image and direct image), and how this can be used to represent logical systems "internally" to monoidal closed bifibrations.



(a) Refinement (b) Typing and subtyping judgments (c) Typing derivations

Figure 1: An illustration of various type-theoretic concepts associated to a functor $p : \mathbb{E} \to \mathbb{I}$

References

[1] Paul-André Melliès and Noam Zeilberger. "Type refinement and monoidal closed bifibrations". Unpublished manuscript of 1 October, 2013, available at `http://arxiv.org/abs/1310.0263`.

A classical theory of values and computations

José Espírito Santo[1], Ralph Matthes[2], Koji Nakazawa[3], Luís Pinto[1]
[1] Departamento de Matemática, Universidade do Minho, Portugal
[2] I.R.I.T. (C.N.R.S. and University of Toulouse III), France
[3] Graduate School of Informatics, Kyoto University, Japan

October 1, 2013

This research continues the line of thought of our previous paper [ENMP14] which already aimed at a better understanding of reduction-preserving translations of call-by-name (cbn) and call-by-value (cbv) $\overline{\lambda}\mu\tilde{\mu}$ [CH00] into simply-typed lambda-calculi.

We introduce a refinement of our monadic $\lambda\mu$-calculus [ENMP14] to a new calculus $\mathsf{VC}\mu_{\mathsf{M}}$ of values and computations, which is based on a sub-language of the former.

Crucially, and inspired by the presentation of the monadic meta-language in [HD94], we restrict function space / implication (which we write $\supset$) to the form $A \supset MA'$, where M is the formal monad, see [Mog91]. We exclude types of the form $M(MA)$ altogether: types are given by $A ::= B \,|\, C$, with $B ::= X \,|\, A \supset C$ (with type variables X) and $C ::= MB$. Following *op. cit.*, we call types B (resp. C) "value types" (resp. "computation types"). Also the expressions of the calculus are divided into *values* V and *computations* P, and this is obtained by separating two sets of term variables, *value variables* v and *computation variables* y, with the intention of having a *well-moded* typing system, that is, one that assigns to the variables types with the right "mode" (value or computation).

At a glance, the syntax is given in Figure 1 (using forms of explicit substitution in the commands), and the typing rules in Figure 2. The next step are more refined reduction rules for the expressions, but this would lead technically too far here. The talk will motivate the design decisions for $\mathsf{VC}\mu_{\mathsf{M}}$ and give the system in full details.

We recall the overview of results from our previous paper in Figure 3, which in fact contains two pictures, one for cbn and another for cbv. In essence, our results on $\mathsf{VC}\mu_{\mathsf{M}}$ improve [ENMP14] in the following way:

1. we study a new system - hybrid $\overline{\lambda}\mu\tilde{\mu}$ - that amalgamates cbn and cbv $\overline{\lambda}\mu\tilde{\mu}$ without losing confluence, and can serve as the source of a new monadic translation.

2. we give a new monadic translation, with target on $\mathsf{VC}\mu_{\mathsf{M}}$ (instead of $\overline{\lambda}\mu\tilde{\mu}$), which, for this reason, works for both cbn and cbv.

3. instantiation with the continuations monad works for both cbn and cbv without dedicated optimizations.

4. the forgetful map from hybrid $\overline{\lambda}\mu\tilde{\mu}$ blurs the distinction cbn/cbv *with* loss of confluence; the forgetful map from $\mathsf{VC}\mu_{\mathsf{M}}$ blurs the distiction value/computation *without* loss of confluence.

Figure 1: Expressions of $\mathsf{VC}\mu_{\mathsf{M}}$

$$\begin{array}{rrcl} \text{(values)} & V, W & ::= & v \,|\, \lambda x.P \\ \text{(computations)} & P, Q & ::= & y \,|\, \eta V \,|\, V\,u \,|\, \mu a.c \\ \text{(terms)} & t, u & ::= & V \,|\, P \\ \text{(commands)} & c & ::= & aP \,|\, \mathsf{bind}(P, v.c) \,|\, \{P/y\}c \end{array}$$

Figure 2: Typing rules of $\mathsf{VC}\mu_{\mathsf{M}}$

$$\frac{}{\Gamma, v : B \vdash v : B|\Delta}\ Axv \qquad \frac{}{\Gamma, y : C \vdash y : C|\Delta}\ Axc$$

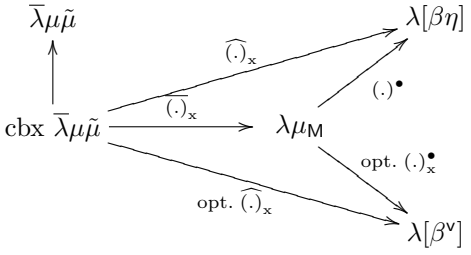
$$\frac{\Gamma, x : A \vdash P : C|\Delta}{\Gamma \vdash \lambda x.P : A \supset C|\Delta}\ Intro \qquad \frac{\Gamma \vdash V : A \supset C|\Delta \quad \Gamma \vdash u : A|\Delta}{\Gamma \vdash Vu : C|\Delta}\ Elim$$

$$\frac{\Gamma \vdash P : C|a : C, \Delta}{aP : (\Gamma \vdash a : C, \Delta)}\ Pass \qquad \frac{c : (\Gamma \vdash a : C, \Delta)}{\Gamma \vdash \mu a.c : C|\Delta}\ Act$$

$$\frac{\Gamma \vdash V : B|\Delta}{\Gamma \vdash \eta V : MB|\Delta} \qquad \frac{\Gamma \vdash P : MB|\Delta \quad c : (\Gamma, v : B \vdash \Delta)}{\mathsf{bind}(P, v.c) : (\Gamma \vdash \Delta)}$$

$$\frac{\Gamma \vdash P : C|\Delta \quad c : (\Gamma, y : C \vdash \Delta)}{\{P/y\}c : (\Gamma \vdash \Delta)}$$

Figure 3: The cbx picture from [ENMP14]



cbx	cbn or cbv
$\lambda\mu_{\mathsf{M}}$	monadic $\lambda\mu$-calculus
$\overline{(.)}_{\mathrm{x}}$	cbx monadic translation
$(.)^{\bullet}$	instantiation to continuations monad
$\widehat{(.)}_{\mathrm{x}}$	cbx CPS translation
opt. $(.)^{\bullet}_{\mathrm{x}}$	optimized cbx inst. to conts. monad
opt. $\widehat{(.)}_{\mathrm{x}}$	optimized cbx CPS translation

This will be illustrated with a new picture that captures both the cbn and cbv paradigm and is thus a single picture.

References

[CH00] P.-L. Curien and H. Herbelin. The duality of computation. In *Proc. of ICFP 2000*, pages 233–243. IEEE, 2000.

[ENMP14] José Espírito Santo, Koji Nakazawa, Ralph Matthes, and Luís Pinto. Monadic translation of classical sequent calculus. *to appear in MSCS – available online at the publisher since January 2013*, 2014.

[HD94] John Hatcliff and Olivier Danvy. A generic account of continuation-passing styles. In *Proc. of POPL 1994*, pages 458–471. ACM, 1994.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991.

Distributed Mathematics

Chad Brown

As theorem proving systems have advanced, it has become possible to realistically formalize large mathematical theories. In addition, different parts of the theories can be formalized by different people anywhere in the world. The primary concerns are that the proofs are correct and that the formalization efforts are not redundant. An example of such an effort is the project to classify the finite simple groups in SSReflect. Another example is the Flyspeck project, an attempt to formalize Thomas Hales' proof of the Kepler Conjecture in HOL-light.

Before beginning a formalization project, the participants must fix the notions of propositions and proofs. In other words, there must be a common foundation. There are many theorem proving systems with many different foundations. The Flyspeck project uses the HOL-light system, based on a form of classical higher-order logic. Coq (and hence SSReflect) is based on a rich intuitionistic higher-order type theory. Another system in which a great deal of mathematics has been formalized is Mizar, which is based on first order Tarski-Grothendieck set theory. Tarski-Grothendieck set theory is an extension of Zermelo-Fraenkel set theory with universes. The addition of universes makes the set theory strong enough to interpret not only the higher-order logic of HOL-light but also the higher-order type theory of Coq (using a proof irrelevant model). In principle, one could translate formalizations from HOL-light and Coq into formalizations in Mizar. (In practice, this would fail in some cases since one-step conversion proofs in HOL-light and Coq would need to translate to many-step equational proofs in Mizar.)

In my talk, I will describe a simply typed version of Tarski-Grothendieck set theory as a foundation for formal mathematics. This foundation is strong enough to interpret Mizar (in principle) and by extension the other systems above (in principle).

Once we fix the foundation as being simply typed Tarski-Grothendieck, we need a representation of proofs. The easy, well-known way of representing proofs is using the Curry-Howard-de Bruijn correspondence. That is, proofs are natural deduction proofs. Each such proof may be represented as a proof term: a lambda term with constants for the axioms of the theory. This is arguably the oldest representation of formal proofs on a computer, as they were used in de Bruijn's AUTOMATH project. Serialization and communication of proof terms is no more difficult than serialization and communication of propositions.

Since the underlying logic of our foundation is simple type theory, we also have the possibility of using generic automated higher-order theorem provers to find proofs. In the past few years, I have developed such a higher-order theorem prover, Satallax. Satallax has placed first or second in the higher-order division of the CASC system competition the past four years, and has won two of those years. I will spend some time talking about how Satallax can be used to search for proofs and generate proof terms.

In addition to proofs, a formal mathematical development will also contain definitions and lemmas. Inevitably, different people will make the same definitions and prove the same lemmas, but give them different names. This can be detected through the use of collision-resistant cryptographic hashing functions. Furthermore, cryptographic digital signatures can be used by a contributor to claim to have a proof of a conjecture before the contributor is prepared to publish the proof. The hope is that these ideas could be used to prevent duplication of effort.

I am currently working on an implementation of these ideas, and will describe the current state of the implementation.

References

[1] Aczel, Peter. On Relating Type Theories and Set Theories. In: Types for Proofs and Programs, International Workshop TYPES '98, Kloster Irsee, Germany, March 27-31, 1998, Selected Papers, ed. T. Altenkirch, W. Naraschewski, B. Reus. Lecture Notes in Computer Science 1657, pp. 1-18, 1998.

[2] Barras, Bruno. Sets in Coq, Coq in Sets. Journal of Formalized Reasoning 3(1), 2010.

[3] Brown, Chad E. Reducing Higher-Order Theorem Proving to a Sequence of SAT Problems. Journal of Automated Reasoning 51(1), pp. 57-77, March 2013.

[4] Brown, Chad E. Reconsidering Pairs and Functions as Sets. Technical Report, Saarland University, August 2013. (Submitted)

[5] de Bruijn, N. G. A survey of the project AUTOMATH. In: To H.B. Curry: Essays in combinatory logic, lambda calculus and formalism, ed. J.P. Seldin and J.R. Hindley, Academic Press, pp. 579-606, 1980.

[6] Gonthier, Georges and Asperti, Andrea and Avigad, Jeremy and Bertot, Yves and Cohen, Cyril and Garillot, Franois and Le Roux, Stphane and Mahboubi, Assia and O'Connor, Russell and Ould Biha, Sidi and Pasca, Ioana and Rideau, Laurence and Solovyev, Alexey and Tassi, Enrico and Thry, Laurent. A Machine-Checked Proof of the Odd Order Theorem. ITP 2013, 4th Conference on Interactive Theorem Proving, ed. Sandrine Blazy, Christine Paulin and David Pichardie. Springer 7998, pp. 163-179, 2013.

[7] Grothendieck, A. and Verdier, J.-L. Thorie des topos et cohomologie tale des schmas - (SGA 4) - vol. 1. Lecture Notes in Mathematics 269, pp. 185-217, 1972.

[8] Hales, Thomas. Formal proof. Notices of the AMS, 55(11):13701380, 2008.

[9] Lee, Gyesik and Werner, Benjamin. Proof-irrelevant model of CC with predicative induction and judgmental equality. Logical Methods in Computer Science 7 (4:5), 2011.

[10] Prawitz, Dag. Natural deduction: a proof-theoretical study. Dover, 2006.

[11] Stallings, William. Cryptography and Network Security: Principles and Practice (4th Edition). Prentice Hall, 2005.

[12] Trybulec, Andrzej. Tarski Grothendieck Set Theory. Journal of Formalized Mathematics. Axiomatics. Inst. of Computer Science, University of Bialystok, 2002 (Released 1989)

[13] Urban, Josef. Translating Mizar for First Order Theorem Provers. Mathematical Knowledge Management, Second International Conference, MKM 2003, Bertinoro, Italy, February 16-18, 2003, Proceedings, ed. Andrea Asperti, Bruno Buchberger and James H. Davenport. Lecture Notes in Computer Science 2594, pp. 203-215, 2003.

[14] Werner, Benjamin. Sets in Types, Types in Sets. In:Theoretical Aspects of Computer Software, Third International Symposium, TACS '97, Sendai, Japan, September 23-26, 1997, Proceedings, ed. M. Abadi, T. Ito. Lecture Notes in Computer Science 1281, pp. 530-546, 1997.