Report from Dagstuhl Seminar 16421

Universality of Proofs

Edited by

Gilles Dowek[1], Catherine Dubois[2], Brigitte Pientka[3], and Florian Rabe[4]

1 **INRIA & ENS Cachan, FR,** gilles.dowek@ens-cachan.fr
2 **ENSIIE – Evry, FR,** catherine.dubois@ensiie.fr
3 **McGill University – Montreal, CA,** bpientka@cs.mcgill.ca
4 **Jacobs University Bremen, DE,** f.rabe@jacobs-university.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 16421 *Universality of Proofs* which took place October 16–21, 2016.

The seminar was motivated by the fact that it is nowadays difficult to exchange proofs from one proof assistant to another one. Thus a formal proof cannot be considered as a *universal* proof, reusable in different contexts. The seminar aims at providing a comprehensive overview of the existing techniques for interoperability and going further into the development of a common objective and framework for proof developments that support the communication, reuse and interoperability of proofs.

The seminar included participants coming from different fields of computer science such as logic, proof engineering, program verification, formal mathematics. It included overview talks, technical talks and breakout sessions. This report collects the abstracts of talks and summarizes the outcomes of the breakout sessions.

Seminar October 16–21, 2016 – http://www.dagstuhl.de/16421
1998 ACM Subject Classification Semantics / Formal Methods, Verification / Logic
Keywords and phrases Formal proofs, Interoperability, Logical frameworks, Logics, Proof formats, Provers, Reusability
Digital Object Identifier 10.4230/DagRep.6.10.75

1 Executive Summary

Gilles Dowek
Catherine Dubois
Brigitte Pientka
Florian Rabe

Proof systems are software systems that allow us to build formal proofs, either interactively or automatically, and to check the correctness of such proofs. Building such a formal proof is always a difficult task – for instance the Feit-Thompson odd order theorem, the CompCert verified C compiler, the seL4 verified operating system micro-kernel, and the proof of the Kepler conjecture required several years with a medium to large team of developers to be completed. Moreover, the fact that each of these proofs is formalized in a specific logic and the language of a specific proof tool is a severe limitation to its dissemination within the community of mathematicians and computer scientists. Compared to many other branches of computer science, for instance software engineering, we are still very far from having off-the-shelf and ready-to-use components, “proving in the large” techniques, and interoperability of theory and systems. However, several teams around the world are working on this issue and partial solutions have been proposed including point-to-point translations, proof standards, and logical frameworks. Yet, a lot still remains to be done as there is currently no overarching general foundation and methodology.

This seminar has been organized to bring together researchers from different communities, such as automated proving, interactive proving and SAT/SMT solving as well as from logic, proof engineering, program verification and formal mathematics. An essential goal has been to form a community around these issues in order to learn about and reconcile these different approaches. This will allow us to develop a common objective and framework for proof developments that support the communication, reuse, and interoperability of proofs.

The program of the seminar included introductions to different methods and techniques, the definition of precise objectives, and the description of recent achievements and current trends. It consisted of 30 contributed talks from experts on the above topics and six breakout sessions on major problems: theory graph – based reasoning, benchmarks, conflicting logics and system designs, proof certificates, design of a universal library of elementary mathematics, and a standard for system integration and proof interchange. The contributed talks took place in the morning, and two parallel breakout sessions each took place on Monday, Tuesday and Thursday afternoon, followed by plenary discussions organized by each session’s moderator.

The organizers would like to thank the Dagstuhl team and all the participants for making this first seminar a success and, hopefully, an event to be repeated.

2 Table of Contents

3 Overview of Talks

3.1 Translating between Agda and Dedukti

Andreas Martin Abel (Chalmers UT – Göteborg, SE)

Boespflug and Burel have designed CoqInE, a translation of Coq's Calculus of Inductive Constructions into Dedukti, the logical framework with rewriting. We conjecture a similar translation could be possible from Agda to Dedukti. The reverse translation is possible since Agda recently got extended with rewrite tools. In this talk, we propose an implementation of translations between Agda and Dedukti.

3.2 Transferring Lemmas and Proofs in Isabelle/HOL: a Survey

Jesús María Aransay Azofra (University of La Rioja – Logroño, ES)

Data types admit different representations. The reasons to prioritise one or another range from efficiency to simplicity. Interactive theorem provers allow users to work with these representations and offer tools that ease their communication. These tools are required to preserve the formalism among the representations. In this talk we present some use cases in the proof assistant Isabelle/HOL, defining the scope of each of these tools and their possible scenarios.

3.3 Uniform Proofs via Shallow Semantic Embeddings?

Christoph Benzmüller (FU Berlin, DE)

Main reference C. Benzmüller, B. Woltzenlogel Paleo, "The Inconsistency in Gödel's Ontological Argument: A Success Story for AI in Metaphysics", IJCAI 2016, pp. 936–942, 2016.
URL http://www.ijcai.org/Abstract/16/137

Many classical and non-classical logics can be elegantly mechanised and automated by exploiting shallow semantical embeddings in classical higher-order logic. In recent research this approach has been successfully applied in various disciplines, including metaphysics, mathematics and artifical intelligence. Moreover, it has recently been utilised as the core framework in my (awarded) lecture course on Computational Metaphysics at Freie Universität Berlin. In this talk I demonstrate the approach and discuss its potential for achieving uniform proofs across various logics.

References

1 Christoph Benzmüller and Dana Scott, *Axiomatizing Category Theory in Free Logic*. arXiv, http://arxiv.org/abs/1609.01493, 2016.

2 Christoph Benzmüller and Dana Scott, *Automating Free Logic in Isabelle/HOL*. In Mathematical Software – ICMS 2016, 5th International Congress, Proceedings (G.-M. Greuel, T. Koch, P. Paule, A. Sommese, eds.), Springer, LNCS, volume 9725, pp. 43-50, 2016.
3 Christoph Benzmüller, Max Wisniewski and Alexander Steen, Computational Metaphysics – Bewerbung zum zentralen Lehrpreis der Freien Universität Berlin, FU Berlin, 2015.
4 Christoph Benzmüller and Bruno Woltzenlogel Paleo, *The Inconsistency in Gödel's Ontological Argument: A Success Story for AI in Metaphysics*. In IJCAI 2016 (Subbarao Kambhampati, ed.), AAAI Press, volume 1-3, pp. 936-942, 2016.
5 Christoph Benzmüller and Bruno Woltzenlogel Paleo, *Automating Gödel's Ontological Proof of God's Existence with Higher-order Automated Theorem Provers*. In ECAI 2014 (Torsten Schaub, Gerhard Friedrich, Barry O'Sullivan, eds.), IOS Press, Frontiers in Artificial Intelligence and Applications, volume 263, pp. 93 – 98, 2014.

3.4 Are Translations between Proof Assistants Possible or Even Desirable at All?

Jasmin Christian Blanchette (MPI für Informatik – Saarbrücken, DE)

Joint work of Jasmin Christian Blanchette, Sascha Böhme, Mathias Fleury, Steffen Juilf Smolka, Albert Steckermeier
Main reference J. C. Blanchette, S. Böhme, M. Fleury, S. J. Smolka, A. Steckermeier,"Semi-intelligible Isar proofs from machine-generated proofs", J. Autom. Reasoning, Vol. 56(2), pp. 155–200, Springer, 2016.
URL https://dx.doi.org/10.1007/s10817-015-9335-3

Combining proofs developed using different proof assistants would seem to be highly desirable. After all, a lot of effort goes into formalizing a result in one assistant, and it makes sense to reuse it as much as possible. However, there are lots of obstacles before we have tools that can be widely deployed. When Isabelle/HOL users port analysis results from HOL Light, in practice they cannot rely on automatic bridges such as OpenTheory, even though both systems are based on simple type theory. I will review different approaches to prove exchange and emphasize their shortcomings from the viewpoint of end users. I will also briefly describe my work on translation of proofs between automatic theorem provers and Isabelle/HOL.

3.5 Using External Provers in Proof Assistants

Frédéric Blanqui (ENS – Cachan, FR)

URL http://cl-informatik.uibk.ac.at/software/cpf/

Using external provers in proof assistants is an example of small scale inter-operability. It relieves proof assistant users and proof assistant developers. The main obstacle is to be able to certify the results of the external prover. It is now well established for proving propositional or first-order subgoals, except perhaps in proof assistants using dependent types. But it is also possible to use external provers for termination and confluence problems. Indeed, since 2009, there is a common format called CPF for termination certificates and certified tools to check their correctness.

3.6 The Continuity of Monadic Stream Functions

Venanzio Capretta (University of Nottingham, GB)

Streams are infinite sequences of values. They inhabit a frontier region of constructive mathematics and computer science: they cannot be represented fully inside human minds and computer memories, but they are omnipresent as input, output and interactive behaviour.

Brouwer formulated the notion of "choice sequence", a progression of values that is not generated by an effective rule, but rather by a creative subject. Alternatively, they may model repeated measurement of physical phenomena or interactive input from a non-predictable user.

If the streams themselves are not computable, functions on them, realized as programs, must be effective. From this requirement, Brouwer concluded that a Continuity Principle must hold: all functions on streams of natural numbers are continuous. This means that the value of a function on a specific stream only depends on a finite initial segment.

The principle seems to be justifiable in a computational view and is certainly true from the meta-theoretical standpoint. We may be tempted to add it in a formulation of the foundations of constructive mathematics. However, recently Martín Escardó discovered that if we add the Continuity Principle to Constructive Type Theory, we obtain a contradiction. I will discuss the paradox and the possible avenues of repair. One way is to weaken the principle, using an existential quantifier that does not provide a witness.

I suggest a different solution. In the original formulation, we consider functions on the internal type of streams, encoded as functions from natural numbers to natural numbers. But this type does not capture the idea of an unpredictable sequence not subject to rule and possibly coming from an outside source. I propose that "monadic streams" are a better model: these are sequences in which a monadic action must be executed to obtain the next element and the continuation. A monadic action is any of a wide class of enriched structures and modes of value presentation. Monadic programming has been very successful in modelling interaction and side effects in functional programming.

I propose a version of the Continuity Principle for monadic streams. This has great potential not only as a foundational theory but in practical applications. Monadic streams have already been successfully used in functional reactive programming and game implementation. The principle applies to functions on monadic streams that are polymorphic in the monad and natural on it. They are a reasonable description of mappings on sequences that do not depend on how the sequence is generated. I will prove that these functions are always continuous.

I think these issues are extremely important for the future of computer-assisted mathematics. Monadic streams are a very promising data structure, needed to model reactive continuous processes. This work shows that they are also relevant in the design of the logical principles underlying formalized mathematics.

3.7 Reengineering Proofs in Dedukti: an Example

Gilles Dowek (INRIA & ENS Cachan, FR)

The system Dedukti is a logical framework. We illustrate how it can be used to reengineer proofs with the example of the translation to Simple type theory of proofs expressed in the Calculus of constructions.

3.8 FoCaLiZe and Dedukti to the Rescue for Proof Interoperability

Catherine Dubois (ENSIIE – Evry, FR)

Joint work of Raphaël Cauderlier, Catherine Dubois

We propose a methodology to combine proofs coming from different provers relying on Dedukti as a common formalism in which proofs can be translated and combined. To relate the independently developed mathematical libraries used in proof assistants, we rely on the structuring features offered by FoCaLiZe. We illustrate this methodology on the Sieve of Eratosthenes, which we prove correct using HOL and Coq in combination.

3.9 We Need a Better Style of Proof

William M. Farmer (McMaster University – Hamilton, CA)

Proofs serve several diverse purposes in mathematics. They are used to communicate mathematical ideas, certify that mathematical results are correct, discover new mathematical facts, learn mathematics, establish the interconnections between mathematical ideas, show the universality of mathematical results, and create mathematical beauty. Traditional proofs and (computer-supported) formal proofs do not fulfill these purposes equally well. In fact, traditional proofs serve some purposes much better than formal proofs, and vice versa. For example, traditional proofs are usually better for communication, while formal proofs are usually better for certification. We compare both traditional and formal proofs with respect to these seven purposes and show that both styles of proof have serious shortcomings. We offer a new style of proof in which (1) informal and formal proof components are combined (in accordance with Michael Kohlhase's notion of flexiformality), (2) results are proved at the optimal level of abstraction (in accordance with the little theories method), and (3) cross-checks are employed systematically. We argue that this style of proof fulfills the purposes of mathematical proofs much better than both traditional and formal proofs.

3.10 Comparing Systems for Reasoning with Higher-Order Abstract Syntax Representations

Amy Felty (University of Ottawa, CA)

Over the past three decades, a variety of meta-reasoning systems which support reasoning about higher-order abstract specifications have been designed and developed. We summarize our work on surveying and comparing four meta-reasoning systems, Twelf, Beluga, Abella and Hybrid, using several benchmarks from the open repository ORBI that describes challenge problems for reasoning with higher-order abstract syntax representations. In particular, we investigate how these systems mechanize and support reasoning using a context of assumptions. This highlights commonalities and differences in these systems and is a first step towards translating between them.

3.11 Aligning Concepts across Proof Assistant Libraries

Thibault Gauthier (Universität Innsbruck, AT)

As the knowledge available in the computer understandable proof corpora grows, recognizing repeating patterns becomes a necessary requirement in order to organize, synthesize, share, and transmit ideas. In this work, we automatically discover patterns in the libraries of interactive the- orem provers and thus provide the basis for such applications for proof assistants. This involves detecting close properties, inducing the presence of matching concepts, as well as dynamically evaluating the quality of matches from the similarity of the environment of each concept. We further propose a classification process, which involves a disambiguation mechanism to decide which concepts actually represent the same mathematical ideas. We evaluate the approach on the libraries of six proof assistants based on different logical foundations: HOL4, HOL Light, and Isabelle/HOL for higher-order logic, Coq and Matita for intuitionistic type theory, and the Mizar Mathematical Library for set theory. Comparing the structures available in these libraries our algorithm automatically discovers hundreds of isomor- phic concepts and thousands of highly similar ones.

3.12 Extending Higher-order Logic with Predicate Subtyping

Frédéric Gilbert (ENS – Cachan, FR)

Predicate subtyping is an extension of higher-order logic where the grammar of types is enriched with a construction for restricted comprehension allowing to define, for any type A and any predicate P on A, a new type {A | P}. The inhabitants of such a type are the inhabitants t of A for which P(t) is provable. As a consequence, type-checking becomes undecidable. We present a possible formalization of predicate subtyping, which can be the base of a formalization of the proof assistant PVS. We also present a similar system using explicit proofs and coercions. This system is used as a lightweight language for predicate subtyping. It is also a first step towards the expression of predicate subtyping in a universal system.

3.13 Inference Systems for Satisfiability Problems

Stéphane Graham-Lengrand (Ecole Polytechnique – Palaiseau, FR)

Joint work of Maria Paola Bonacina, Stéphane Graham-Lengrand, Natarajan Shankar

One of the most popular approaches for solving propositional SAT problems is CDCL (Conflict-Driven Clause Learning), a variant of DPLL where model construction steps alternate with conflict analysis steps. In terms of proof theory, this is an alternation between bottom-up and top-down applications of rules from an inference system.

MCSat is a methodology for generalising CDCL to other theories than propositional logic. It thereby addresses (quantifier-free) SAT-Modulo-Theories problems, but in a way that seems rather different from the widely used architecture where DPLL interacts with the combination, by the Nelson-Oppen method, of theory-specific decision procedures.

We identify the notion of an MCSat-friendly inference system, and define a generic MCSat calculus that is sound and complete for satisfiability in the union of n arbitrary theories (including for instance propositional logic), as long as each of them comes with an MCSat-friendly inference system.

We show how the Nelson-Oppen method is a particular case of our MCSat-combination method, reconciling the widely-implemented technique with the new MCSat ideas.

References

1 M. P. Bonacina, S. Graham-Lengrand, and N. Shankar. A model-constructing framework for theory combination. Research report, Università degli Studi di Verona – SRI International – CNRS – INRIA, 2016. Available at http://hal.archives-ouvertes.fr/hal-01425305

2 Leonardo de Moura and Dejan Jovanović. A model-constructing satisfiability calculus. In Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni, editors, *Proc. of the 14th Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 7737 of *LNCS*, pages 1–12. Springer, 2013.

3 Dejan Jovanović, Clark Barrett, and Leonardo de Moura. The design and implementation of the model-constructing satisfiability calculus. In Barbara Jobstman and Sandip Ray, editors, *Proc. of the 13th Conf. on Formal Methods in Computer Aided Design (FMCAD)*. ACM and IEEE, 2013.

3.14 Not Incompatible Logics

Olivier Hermant (Ecole des Mines de Paris, FR)

The formalisms used to express proofs are most of the time incompatible, either in a strong form, inconsistent with each other, or in a weaker form, leading to different properties of the logic.

In this talk, I introduce the story of overcoming this difficulty on the example of classical and intuitionistic logic.

3.15 Lazy Proofs for DPLL(T)-Based SMT Solvers

Guy Katz (Stanford University, US)

Joint work of Guy Katz, Clark Barrett, Cesare Tinelli, Andrew Reynolds, Liana Hadarean
Main reference G. Katz, C. Barrett, C. Tinelli, A. Reynolds, L. Hadarean, "Lazy Proofs for DPLL(T)-Based SMT Solvers", in Proc. of the 16th Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD), pp. 93–100, 2016.
URL https://stanford.edu/~guyk/pub/FMCAD2016.pdf

With the integration of SMT solvers into analysis frameworks aimed at ensuring a system's end-to-end correctness, having a high level of confidence in these solvers' results has become crucial. For unsatisfiable queries, a reasonable approach is to have the solver return an independently checkable proof of unsatisfiability. We propose a lazy, extensible and robust method for enhancing DPLL(T)-style SMT solvers with proof-generation capabilities. Our method maintains separate Boolean-level and theory-level proofs, and weaves them together into one coherent artifact. Each theory-specific solver is called upon lazily, a posteriori, to prove precisely those solution steps it is responsible for and that are needed for the final proof. We present an implementation of our technique in the CVC4 SMT solver, capable of producing unsatisfiability proofs for quantifier-free queries involving uninterpreted functions, arrays, bitvectors and combinations thereof. We discuss an evaluation of our tool using industrial benchmarks and benchmarks from the SMTLIB library, which shows promising results.

3.16 The Triumvirate of Automation, Expressivity, and Safety

Chantal Keller (University of Paris Sud – Orsay, FR)

In this survey, I will analyze various approaches to interoperability between proof systems: what effort does this interoperability requires? Can two systems be really agnostic of each other to communicate? How deep can we go into automation, expressivity and safety?

In particular, I will present:

- the interoperability a posteriori between already established interactive and automatic theorem provers, such as SMTCoq or Ergo for the Coq proof assistant or sledgehammer for the Isabelle/HOL proof assistant;
- the interoperability a priori inside proof systems that are designed to be automatic, expressive and safe in a more tightened way, such as lean, F* or Why3.

3.17 Reproducibility, Trust, and Proof Checkings

Dale Miller (INRIA Saclay – Île-de-France, FR)

Formal proofs are produced and checked by machines. Machines are physical devices, of course, and their software and their execution are subject to errors. As in other scientific domains, reproducibility is key to establishing trust, whether it is a claim in physics or a claim that a given file contains a valid proof. A high degree of trust in a formal proof comes from executing a trusted proof checker on a claimed proof, thereby, reproducing the claim. In order to trust a proof checker, it should be possible to implement new proof checkers or to exam the source of existing provers and to be convinced that they are sound implementations of logic. Providing a formal semantics for proof languages is an important step in allowing for this kind of independent and trustworthy proof checking to be achieved.

3.18 Benchmarks for Mechanized Meta-theory: a very Personal and Partial View

Alberto Momigliano (University of Milan, IT)

Joint work of Amy Felty, Alberto Momigliano, Brigitte Pientka

Benchmarks in theorem proving have been very useful, made the state of the art progress or at least take stock, as the bright example of *TPTP* testifies, whose influence on the development, testing and evaluation of automated theorem provers cannot be underestimated. The situation is less satisfactory for proof assistants, where each system comes with its own set of examples/libraries, some of them gigantic. This is not surprising, since we are potentially addressing the whole realm of mathematics.

In this talk I try to evaluate the impact, if any, that benchmarks have had on the sub-field of the meta-theory of deductive systems, such as the ones studied in Programming Language Theory, and its feedback, again if any, on the development of logical frameworks.

References

1 Amy P. Felty, Alberto Momigliano, Brigitte Pientka: *The Next 700 Challenge Problems for Reasoning with Higher-Order Abstract Syntax Representations – Part 2 – A Survey.* J. Autom. Reasoning 55(4): 307-372 (2015)
2 Amy P. Felty, Alberto Momigliano, Brigitte Pientka: *An Open Challenge Problem Repository for Systems Supporting Binders.* LFMTP 2015: 18-32

3.19 Mechanizing Meta-Theory in Beluga

Brigitte Pientka (McGill University – Montreal, CA)

Joint work of Andrew Cave, Brigitte Pientka

Mechanizing formal systems, given via axioms and inference rules, together with proofs about them plays an important role in establishing trust in formal developments. In this talk, I will survey the proof environment Beluga. To specify formal systems and represent derivations within them, Beluga provides a sophisticated infrastructure based on the logical framework LF; to reason about formal systems, Beluga provides a dependently typed functional language for implementing inductive proofs about derivation trees as recursive functions following the Curry-Howard isomorphism. Key to this approach is the ability to model derivation trees that depend on a context of assumptions using a generalization of the logical framework LF, i.e. contextual LF which supports first-class contexts and simultaneous substitutions.

Our experience has demonstrated that Beluga enables direct and compact mechanizations of the meta-theory of formal systems, in particular programming languages and logics. To demonstrate Beluga's strength in this talk, we develop a weak normalization proof using logical relations.

References

1 A. Cave and B. Pientka. *Programming with binders and indexed data-types*. In *POPL'12*, pages 413–424. ACM, 2012.
2 A. Cave and B. Pientka. *A case study on logical relations using contextual types*. In *LFMTP'15*, pages 18–33. Electr. Proc. in Theoretical Computer Science (EPTCS), 2015.
3 B. Pientka. *A type-theoretic foundation for programming with higher-order abstract syntax and first-class substitutions*. In *POPL'08*, pages 371–382. ACM, 2008.
4 B. Pientka and A. Cave. *Inductive Beluga: Programming proofs (system description)*. In *CADE-25*, LNCS 9195, pages 272–281. Springer, 2015.

3.20 On Universality of Proof Systems

Elaine Pimentel (Federal University of Rio Grande do Norte, BR)

Joint work of Björn Lellmann, Carlos Olarte, Elaine Pimentel

We propose a notion of modular linear nested sequent calculi (LNS) for different modalities which brings down the complexity of proof search to that of the corresponding sequent calculi. Examples include normal and non-normal classical modal logics as well as multiplicative additive linear logic (MALL) plus simply dependent multimodalities. Since LNS systems can be adequately encoded into (plain) linear logic, LL can be seen, in fact, as an "universal framework" for the specification of logical systems. While the modularity of the systems lead to a generic way of building theorem provers for different logics (all of them based on the same grounds), universality of LL allows for the use of the same logical framework for reasoning about all such logical systems.

3.21 MMT: A UniFormal Approach to Knowledge Representation

Florian Rabe (Jacobs University Bremen, DE)

UniFormal is the idea of representing all aspects of knowledge uniformly, including narratione, deduction, computation, and databases. Moreover, it means to abstract from the multitude of individual systems, which not only often focus on just one aspect but are doing so in mutually incompatible ways, thus creating a universal framework of formal knowledge.

MMT is a concrete representation language to that end. It systematically abstracts from assumptions typically inherent in the syntax and semantics of concrete systems, and focuses on language-independence, modularity, and system interoperability. While constantly evolving in order to converge towards UniFormal, its design and implementation have become very mature. It is now a readily usable high-level platform for the design, analysis, and implementation of formal systems.

This talk gives an overview of the current state of MMT, its existing successes and its future challenges.

3.22 Higher Order Constraint Logic Programming for Interactive Theorem Proving

Claudio Sacerdoti Coen (University of Bologna, IT)

Some interactive theorem provers, like Coq, Matita and Agda are implemented around the Curry-Howard isomorphism. Proof checking is type checking, and it can be compactly represented in an Higher Order Logic Programming (HOLP) language / logical framework. Interactive proof construction, however, requires the manipulation of terms containing metavariables, and a significant amount of logic independent code to accomodate metavariables (and narrowing) in "type checking" (aka elaboration, refinement). We propose to delegate such work to the metalanguage by extending HOLP with Constraint Programming features induced by a delay mechanism for "too flexible" goals.

3.23 LLFP: a Framework for Interconnecting Logical Frameworks

Ivan Scagnetto (University of Udine, IT)

Joint work of F. Honsell, M. Lenisa, L. Liquori, P. Maksimovic, V. Michielini, Ivan Scagnetto
Main reference F. Honsell, L. Liquori, P. Maksimovic, I. Scagnetto, "LLFP: A Logical Framework for Modeling External Evidence, Side Conditions, and Proof Irrelevance using Monads", 2016.
URL https://users.dimi.uniud.it/~ivan.scagnetto/LLFP__LMCS.pdf

LLFP (Lax LF with Predicates) is an extension of Edinburgh Logical Framework (LF) with locking type constructors and with a family of monads indexed by predicates over typed terms. Locks are a sort of modality constructors, releasing their argument under the condition that a predicate, possibly external to the system, is satisfied on an appropriate typed judgement. This mechanism paves the way for a proof assistant allowing the user to make calls to external oracles (e.g., other proof assistants) during the proof development activity. Such calls are usually made to factor out the complexity of encoding specific features of logical systems which would otherwise be awkwardly encoded in LF, e.g. side-conditions in the application of rules in Modal Logics, and sub-structural rules, as in noncommutative Linear Logic. Using LLFP, these conditions need only to be specified, while their verification can be delegated to an external proof engine, according to the Poincaré Principle. Moreover, monads also express the effect of postponing verifications. This fact allows the user to focus on the main proof, leaving the possibly external verification of details at the end. A first prototype of a type checker for LLFP has been recently written in OCaml by V. Michielini (ENS Lyon): the software currently supports the Coq System as an external proof assistant.

3.24 External termination proofs for Isabelle with IsaFoR and CeTA

René Thiemann (Universität Innsbruck, AT)

Joint work of Alexander Krauss, Christian Sternagel, René Thiemann, Carsten Fuhs, Jürgen Giesl
Main reference A. Krauss, C. Sternagel, R. Thiemann, C. Fuhs, J. Giesl, "Termination of Isabelle Functions via Termination of Rewriting", in Proc. of the 2nd Int'l Conf. on Interactive Theorem Proving (ITP'11), LNCS Vol. 6898, pp. 152–167, Springer, 2011.
URL http://dx.doi.org/10.1007/978-3-642-22863-6_13

CeTA is a certifier for automatically generated termination proofs, which supports a wide variety of termination techniques. Its soundness is proven is IsaFoR, the Isabelle formalization of rewriting.

We will present an overview of the capabilities of CeTA, and also discuss to which extent CeTA can be used to discharge termination proof obligations that arise from function definitions in Isabelle itself.

3.25 Parsing Mathematics by Learning from Aligned Corpora and Theorem Proving

Josef Urban (Czech Technical University – Prague, CZ)

Joint work of Cezary Kaliszyk, Josef Urban, Jiri Vyskocil
Main reference C. Kaliszyk, J. Urban, J. Vyskocil, "Learning to Parse on Aligned Corpora (Rough Diamond)", in Proc. of the Int'l Conf. on Interactive Theorem Proving (ITP 2015), LNCS, Vol. 9236, pp. 227–233, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-22102-1_15

One of the biggest hurdles that mathematicians encounter when working with formal proof assistants is the necessity to get acquainted with the formal terminology and the parsing mechanisms used in formal proof. While overloading and syntactic ambiguity are ubiquitous in regular mathematics,theorem proving requires full formality. This makes computer verification of mathematical proofs a laborious and so far rare enterprise. In this work we start to address this problem by developing probabilistic AI methods that autonomously train disambiguation on large aligned corpora of informal and formal mathematical formulas.

The resulting parse trees are then filtered by strong semantic AI methods such as large-theory automated theorem proving. We describe the general motivation and our first experiments, and show an online system for parsing ambiguous formulas over the Flyspeck library.

References

1 Blanchette, J. C.; Kaliszyk, C.; Paulson, L. C.; and Urban, J. *Hammering towards QED*. J. Formalized Reasoning. 9(1):101–148. 2016.
2 Kaliszyk, C., and Urban, J. *Learning-assisted automated reasoning with Flyspeck*. J. Autom. Reasoning 53(2):173–213. 2014.
3 Kaliszyk, C.; Urban, J.; and Vyskocil, J. *Learning to parse on aligned corpora (rough diamond)*. In Urban, C., and Zhang, X., eds., Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015, Proceedings, volume 9236 of *Lecture Notes in Computer Science*, 227–233. Springer.
4 Tankink, C.; Kaliszyk, C.; Urban, J.; and Geuvers, H. *Formal mathematics on display: A wiki for Flyspeck*. In Carette, J.; Aspinall, D.; Lange, C.; Sojka, P.; and Windsteiger, W., eds., *MKM/Calculemus/DML*, volume 7961 of *LNCS*, 152–167. Springer. 2013.
5 Zinn, C. *Understanding informal mathematical discourse*. Ph.D. Dissertation, University of Erlangen-Nuremberg. 2004.

3.26 Plugging External Provers into the Rodin Platform

Laurent Voisin (SYSTEREL Aix-en-Provence, FR)

Main reference D. Déharbe, P. Fontaine, Y. Guyot, L. Voisin, "Integrating SMT solvers in Rodin", Science of Computer Programming, Vol. 94, pp. 130–143, Elsevier, 2014.
URL http://dx.doi.org/10.1016/j.scico.2014.04.012

The Rodin platform allows to model reactive systems and prove them correct using the Event-B formal notation. The mathematical logic used in classical first-order predicate calculus with equality, set theory and integer arithmetic. The proof are expressed in the sequent calculus, where the inference rules are computed by external reasoners. Some reasoners are implemented by connecting external provers, which provides terminating inference rules. These external provers allow to reduce drastically the need to perform manual proofs, by providing the automation to discharge all trivial facts.

3.27 Computation in Proofs

Freek Wiedijk (Radboud University Nijmegen, NL)

I discuss the Poincare Principle: the notion that calculations do not need to be proved. As part of this I show a small experiment to add a Poincare Principle to HOL Light.

3.28 Ancient History of the Quest for Universality of Proofs

Bruno Woltzenlogel Paleo (Australian National University – Canberra, AU)

This was the second last talk in a seminar where much had already been said about the present and future of universality of proofs. To complement that, I decided to talk briefly about the distant past, sharing interesting facts about Leibniz, which I learned during a historical research triggered by the 300th anniversary of his death. The talk was based on an analysis of selected quotations from Leibniz, which give insight into what Leibniz would have thought if he could see today's state of the art.

Leibniz was a pioneer in the topics of the seminar. Three and a half centuries ago he already dreamt of a universal logical language (characteristica universalis) and a reasoning calculus. But his contribution was not only a dream. He also took concrete initial steps to fulfil his dream, by defining his own language for an algebra of concepts and even describing how to encode its logical sentences into arithmetical expressions that automated calculating machines of his time could handle. While Leibniz desired a universal logical language because he had none, today we seek universality because of we have too many logics and proof languages competing for acceptance. This is a clear, sign of the astonishing success achieved by our community so far. Although somewhat ironic, the plurality of alternatives is a good problem to have.

The potential of a universal logic for solving concrete controversies among people was a major motivation for Leibniz, who also explicitly aimed at all fields of inquiry capable of certainty. When he compares mathematics and metaphysics, for instance, Leibniz shows that he considered mathematics neither controversial enough nor in particular need of extremely precise formal reasoning. In contrast, today's applications of automated reasoning are still heavily biased towards mathematics. Despite a few exceptions, the mainstream attitude is currently not yet as universal with respect to application domains as it could be.

Leibniz was also overly optimistic about how easy it would be to learn a universal logical language. He wanted it to be so simple that anyone could learn it in a week or two. But the most sophisticated expressive universal languages that we have today may still require semester-long advanced courses for gifted students who already have a strong background in logic. Nevertheless, user interfaces for theorem provers have been progressing rapidly and maybe it will not take long for our technology to become universally accessible to all after only a short period of training.

3.29 First-Order Conflict-Driven Clause Learning from a Proof-Theoretical Perspective

Bruno Woltzenlogel Paleo (Australian National University – Canberra, AU)

In this talk I present the new (first-order) conflict resolution calculus: an extension of the resolution calculus inspired by techniques used in modern SAT-solvers. The resolution inference rule is restricted to (first-order) unit propagation and the calculus is extended with a mechanism for assuming decision literals and with a new inference rule for clause learning, which is a first-order generalization of the propositional conflict-driven clause learning (CDCL) procedure. The calculus is sound (because it can be simulated by natural deduction) and refutationally complete (because it can simulate resolution).

4 Working groups

4.1 Breakout Session on Theory Graph Based Reasoning

William M. Farmer (McMaster University – Hamilton, CA)

A *theory graph* is a network of *axiomatic theories* linked by *meaning-preserving mappings*. The theories serve as abstract mathematical models and the mappings serve as information conduits that enable definitions and theorems to be passed from an abstract setting to many other usually more concrete settings. The theories may have different underlying *logics* and *foundations*.

In the first part of the session, we discussed how theory graphs can be used to represent mathematical knowledge and facilitate reasoning in a proof assistant. In the second part, we discussed the following questions:

1. What are examples of contemporary proof assistants and formal software specification systems that implement theory graph techniques?
2. What kind of objects and information (such as decision procedures and parsing/printing rules) can be attached to the theories in a theory graph?
3. What kind of reasoning is needed to build and exploit theory graphs?
4. How can theory graph technology be added to contemporary proof assistants in which all mathematical knowledge resides in a single theory?
5. Would the development of a logic-independent theorem prover for a system supporting theory graphs like MMT be a worthwhile project?

We were not able to achieve a consensus on what should be the answers to these questions. However, we did agree on the following action item: Select a set of theory graph techniques and compare how these techniques are implemented (if at all) in the leading proof assistants and formal software specification systems.

4.2 Breakout Session on Conflicting Logics and System Designs

Olivier Hermant (Ecole des Mines de Paris, FR) and Chantal Keller (University of Paris Sud – Orsay, FR)

We can observe many conflicts between logics, formal systems and even libraries inside the same tools. This session discussed in particular the following questions: how to take advantage in one system of the work in another system, the essence of conflicts in logic, and the ability to switch logics during a formalization process. A wide range of issues was tackled, forming a continuum between the research topics identified above. Some conflicts, like set representation, can be resolved by defining morphisms, and coupling these with an abstraction step may ease the reuse of libraries across systems. This has led to several inter-platform developments, and stressed the need for a language that allows us to navigate between various levels of abstraction. Stronger conflicts, that lead to inconsistencies, might still be solved by a reverse analysis of proofs, so as to import only compatible, yet sufficient, slices of the frameworks, emphasizing the advantage to reason within little theories.

4.3 Breakout Session on a Universal Library

Michael Kohlhase (Universität Erlangen-Nürnberg, DE) and Catherine Dubois (ENSIIE – Evry, FR)

When formalizing mathematics, we usually need to rely on some knowledge which may or may not be formalized in proof assistants. Furthermore these theories may reside in many places and many forms. So the inventory of such formalized mathematical theories is not easy. The question raised in this breakout session concerns the requirements and design of a universal mathematical library. The discussion was organized following the Five W's method (When, What, Where, Who, Why?).

The first point discussed is the content of the universal library: participants agreed on limits, at least high school mathematics and wikipedia as the upper limit. Such a library should contain, for a notion or concept, definitions (multiple definitions if any), some examples and instances, its relations and dependencies, its main properties, a set of theorems characteristic to the properties and links to formal proofs. The next question is related to the organization of such data. Two directions were proposed: a glossary/dictionary or a theory graph. A first step would focus on the definitions of the concepts including their relations; a second step would be to collaborate on the data and develop some services. In this first step we can see many issues: how do we relate concepts? how do we take into account for parallel concepts? how can commutations be represented? what about the detection of problems? A possible solution is to rely on a graph of concepts where each node has a unique definition. Synonyms for similar concepts are attached to the node, giving different views. Examples could also be considered as views. A mechanism of composition is required. A quality control consisting in checking if concepts are similar is required.

Developing such a library is huge work. For example, wikipedia, PlanetMath, MathWorld count 100 000 concepts whereas a traditional mathematics dictionary has 35 000 words. Help should come from retired mathematicians, students, etc.

After having established these requirements, participants discussed the design of such a system. The conclusion was to build a prototype first, exploring existing systems, e.g. MathHub (`https://mathhub.info)/`).

4.4 Breakout session on A standard for system integration and proof interchange

Ramana Kumar (Data61 / NICTA – Sydney, AU) and Florian Rabe (Jacobs University Bremen, DE)

Several requirements were put forward. Jasmin Blanchette suggested a standard similar to TSTP but with more structure, possibly a standardized subset of Isar. Freek Wiedijk suggested that the original source file of the proof should be recoverable and that the high-level structure of the proof should be apparent. Gilles Dowek pointed out that there are three categories of approach to this language: λ-terms, low-level proof steps (like LCF inference rules), or the statements of intermediate results (plus hints and other structure).

Other issues that were discussed were low level versus high level proofs, forwards versus backwards proofs and complete versus partial proofs, as well as the use of metadata to indicate the specific logic or its general features (e.g., being constructive).

The session then split into groups to gain more insight from considering concrete examples.

Finally the session discussed the following sketch by Florian Rabe for the core grammar of a possible proof standard:

S	$::=$	$G\ \mathit{is}\ C \vdash^{L}_{T^*} \{E\}\ \mathit{by}\ \{P\}$	theorem statement
P	$::=$	E	proof term
	$\mid$	$G(C,\ \{E\}^*,\ \{P\}^*)$	operator/tactic applied to arguments
	$\mid$	$\mathit{let}\ C\ \mathit{in}\ \{P\}$	local definition
	$\mid$	$\mathit{hence}\ C\ \mathit{by}\ P; \{P\}$	forward step
	$\mid$	$\mathit{goal}\ C\ \mathit{by}\ P; \{P\}$	backward step
	$\mid$	$\mathit{use}\ E^*$	partial proof
E	$::=$	$G \mid X \mid \ldots$	expressions, terms, types, formulas, etc.
C	$::=$	$(X\ [: E]\ [= E])^*$	contexts
L	$::=$		logic identifier
T	$::=$		theory identifier
G	$::=$		global id from logic, theories, theorems
X	$::=$		local id introduced in proof

Here curly brackets indicate the scope of the local identifiers introduced in the corresponding context, and $C \vdash^{L}_{T^*} E$ expresses the theorem "in logic L after importing the theories T^* we have for all C that E". A more refined version should include metadata to attach, e.g., original sources. It is straightforward to adapt the concrete syntax of existing standards such as TSTP or OMDoc to subsume (and possibly converge to) this abstract syntax.

To move forward, the session concluded that the community should collect standard prototypes and proof examples to better understand if this grammar suffices. Ramana Kumar and Florian Rabe volunteered to host this process using the repository https://github.com/UniFormal/Proofs which is open to and solicits community distributions.

4.5 Breakout Session on Proof Certificates

Dale Miller (INRIA Saclay – Île-de-France, FR)

Diversity

There is range of settings in which proofs and proof certificates are used. There are the familiar axis: classical vs intuitionistic and logic vs arithmetic[1]. If we view formal proofs as a means to communicate between software systems, then such communication takes place across both time and space. If we examine short-distance and long-distance communication in these two dimensions, we have the following grid.

Time	Space	Example of proof
Short	Short	A section in an interactive proof assistant can be dumped to disk in order to resume another day in the same proof assistant.
Long	Short	Completed proofs can be stored in a library associated with a particular proof assistant.
Short	Long	Cooperating but different provers may use specific certificates for their particular and immediate needs.
Long	Long	Proofs given high-level, declarative definitions may allow anyone to recheck them at anytime in the future.

Another aspect of diversity occurs along the specific vs general spectrum. The current most significant use of proof certificates can be found in the areas where the role of logic is significantly constrained. For example, the following areas make use of well established proof formats: SAT solving (e.g., RUP, DRUP, DRAT), SMT (e.g. VeriT), and resolution refutation (e.g., IVY). Proof formats are also established for more encompassing logics: these include LF ($\lambda\Pi$), LFSC, and TPTP. In the field of arithmetic, there are the OpenTheory project (for the HOL family of provers) and Dedukti (for the λ-cube). The Foundational Proof Certificate project is also attempting to find high-level definitions for a wide variety of proof certificates.

Insist on communication of proofs

There was a universal agreement in this breakout session that proof systems should provide options for outputting (exporting) proofs that they find. The following time-line was proposed in order to push the community towards the development of a standard format for proofs.

1. Provers need to be able to output some kind of useful information about their proofs. While the spirit of this proposal is meant to be informal, the intention is for developers of provers to make an effort to output documents that can be useful to others who want to consume proofs (whether to replay them in other systems, to extract information from them, etc). The goal is to insist on a commitment to an act of communication.

[1] We assumed that both induction and co-induction are treated within “arithmetic”.

2. The output from provers should be certified by proof checkers that are independent of the prover. Of course, there may be many different proof checkers which check proofs in a range of formats. The existence of independent proof checkers should start a move to the standardization of proof formats.
3. A single framework for certificates should be developed, tested, and analyzed. This effort builds on the previous two steps and is likely to contain both theoretical and engineering effort.
4. Develop a standard within some official standardization organization, such as ISO.

It is worth noting the role that competitions have played in helping to promote standards within the field. They provide a means for establishing an authority that is able to insist on standards.

What next?

While establishing a single standard for proof certificates seems to be at least several years away, it is worth noting that several hard problems remain even after we are able to make proofs and proof checking into a commodity. Since this breakout session was limited to the certification of proof, the following topics seem to be independent and not directly addressed.

1. Proofs generally are used for both certification and didactics. The problem of being able to read, browse, and interact with proofs and proof certificates was not addressed.
2. The existence of different theories for the same concepts (e.g., groups, real numbers, etc) was also not addressed. Generally theories are taken as axioms about various non-logical symbols and often there is no canonical selection of such non-logical symbols and their axiomization.

4.6 Breakout Session on Benchmarks

Alberto Momigliano (University of Milan, IT) and Amy Felty (University of Ottawa, CA)

This breakout session addressed the problem of designing benchmarks and challenge problems for interative theorem proving systems. The discussion centered around four central questions, and resulted in partial answers and future directions of study. 1) The first question addressed why there is a need for benchmarks. One reason is to understand the differences between systems and highlight their strengths and limitaitons. Another reason is to use them as a starting point for the translation of theorem statements between systems. Furthermore, they can help to stimulate new development in the systems under study. 2) The group also addressed the question of whether we want universal benchmarks or different benchmarks in different areas. There was a general consensus for the latter. 3) The question of how formal benchmark descriptions should be is another important question. On one end of the spectrum, they could be informal natural language descriptions, and on the other end they could be formal parseable specifications. The informal text is always an important component, and there were varying degrees of support for more formal specifications. On the one hand, they provide a precise description, and on the other hand, formulating the theorem statement precisely could be part of the challenge. 4) The last question addressed was the social process of developing benchmarks. We could appoint one or two people in the community to develop and collect benchmarks, and/or we could work on specific benchmarks in a few specific areas during the rest of the week. Two areas chosen for further discussion during the seminar were benchmarks involving binders (well-suited to systems supporting higher-order abstract syntax and related approaches) and benchmarks that involve induction/coinduction, fixpoints, corecursion, etc. (to test and better understand these capabilities in existing widely used proof assistants such as Coq, Isabelle, and Agda).

Participants

- Andreas Martin Abel
Chalmers UT – Göteborg, SE
- Jesús María Aransay Azofra
University of La Rioja – Logroño, ES
- Christoph Benzmüller
FU Berlin, DE
- Jasmin Christian Blanchette
MPI für Informatik – Saarbrücken, DE
- Frédéric Blanqui
ENS – Cachan, FR
- Peter Brottveit Bock
IT University of Copenhagen, DK
- Venanzio Capretta
University of Nottingham, GB
- Benjamin Delaware
Purdue University – West Lafayette, US
- Gilles Dowek
INRIA & ENS Cachan, FR
- Catherine Dubois
ENSIIE – Evry, FR
- William M. Farmer
McMaster University – Hamilton, CA
- Amy Felty
University of Ottawa, CA
- Thibault Gauthier
Universität Innsbruck, AT
- Frédéric Gilbert
ENS – Cachan, FR
- Georges Gonthier
INRIA Saclay – Île-de-France, FR
- Stéphane Graham-Lengrand
Ecole Polytechnique – Palaiseau, FR
- Hugo Herbelin
University Paris-Diderot, FR
- Olivier Hermant
Ecole des Mines de Paris, FR
- Guy Katz
Stanford University, US
- Chantal Keller
University of Paris Sud – Orsay, FR
- Michael Kohlhase
Universität Erlangen-Nürnberg, DE
- Ramana Kumar
Data61 / NICTA – Sydney, AU
- Dale Miller
INRIA Saclay – Île-de-France, FR
- Alberto Momigliano
University of Milan, IT
- César A. Muñoz
NASA Langley – Hampton, US
- Adam Naumowicz
University of Bialystok, PL
- Brigitte Pientka
McGill University – Montreal, CA
- Elaine Pimentel
Federal University of Rio Grande do Norte, BR
- Florian Rabe
Jacobs University Bremen, DE
- Claudio Sacerdoti Coen
University of Bologna, IT
- Ivan Scagnetto
University of Udine, IT
- Gert Smolka
Universität des Saarlandes, DE
- René Thiemann
Universität Innsbruck, AT
- Josef Urban
Czech Technical University – Prague, CZ
- Laurent Voisin
SYSTEREL
Aix-en-Provence, FR
- Freek Wiedijk
Radboud University
Nijmegen, NL
- Bruno Woltzenlogel Paleo
Australian National University – Canberra, AU