Report from Dagstuhl Seminar 17371

Deduction Beyond First-Order Logic

Edited by

Jasmin Christian Blanchette[1], Carsten Fuhs[2], Viorica Sofronie-Stokkermans[3], and Cesare Tinelli[4]

1 **Vrije Universiteit Amsterdam, NL,** `j.c.blanchette@vu.nl`
2 **Birkbeck, University of London, GB,** `carsten@dcs.bbk.ac.uk`
3 **Universität Koblenz-Landau, DE,** `sofronie@uni-koblenz.de`
4 **The University of Iowa – Iowa City, US,** `cesare-tinelli@uiowa.edu`

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 17371 "Deduction Beyond First-Order Logic." Much research in the past two decades was dedicated to automating first-order logic with equality. However, applications often need reasoning beyond this logic. This includes genuinely higher-order reasoning, reasoning in theories that are not finitely axiomatisable in first-order logic (such as those including transitive closure operators or standard arithmetic on integers or reals), or reasoning by mathematical induction. Other practical problems need a mixture of first-order proof search and some more advanced reasoning (for instance, about higher-order formulas), or simply higher-level reasoning steps. The aim of the seminar was to bring together first-order automated reasoning experts and researchers working on deduction methods and tools that go beyond first-order logic. The seminar was dedicated to the exchange of ideas to facilitate the transition from first-order to more expressive settings.

Seminar September 10–15, 2017 – http://www.dagstuhl.de/17371
1998 ACM Subject Classification D.2.4 Software/Program Verification, F.2.2 Nonnumerical Algorithms and Problems, F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic, F.4.2 Grammars and Other Rewriting Systems, G.1.6 Optimization, I.2.3 Deduction and Theorem Proving
Keywords and phrases Automated Deduction, Program Verification, Certification
Digital Object Identifier 10.4230/DagRep.7.9.26
Edited in cooperation with Philipp Rümmer

1 Executive Summary

Jasmin Christian Blanchette
Carsten Fuhs
Viorica Sofronie-Stokkermans
Cesare Tinelli

Much research on automated deduction has traditionally focused on automated reasoning in first-order logic. First-order logic with equality is generally considered a sweet spot on the logic design continuum. Yet, from the point of view of several applications it can be too restrictive as a modeling and reasoning tool. In recent years, there has been a realization that while first-order reasoning is very useful to discharge the bulk of proof obligations, it must be tightly integrated with richer features to be useful in many applications. Practical problems often need a mixture of first-order proof search and some more advanced reasoning, for instance, about non-first-order-axiomatisable theories, higher-order formulas, or simply higher-level reasoning steps.

First-order logic cannot be used to finitely axiomatize many interesting theories, such as those including transitive closure operators, inductive predicates, datatypes, and standard arithmetic on integers or reals. Even provers that provide native support for some of these theories typically fail to prove trivial-looking problems because they lack general support for mathematical induction. Some applications need a richer set of constructs than those provided by first-order logic such as, for instance, the separating conjunction ($*$) and magic wand ($-\!\!*$) connectives of Separation Logic or the disjunctive well-foundedness predicates used in HSF, a popular approach to software model checking based on first-order Horn logic.

There are potential synergies between automatic first-order proving and verification methods developed in the context of richer logics. However, they have not received enough attention by the various deduction sub-communities so far. In general, there is a cultural gap between the various deduction communities that hinders cross-fertilization of ideas and progress.

This Dagstuhl Seminar brought together experts in automated reasoning in first-order logic and researchers working on deduction methods and tools that go beyond first-order logic. The latter included specialists on proof methods for induction, proof planning, and other higher-order or higher-level procedures; and consumers of deduction technology whose specification languages contain non-first-order features. The main goal of the seminar was to exchange ideas and explore ways to facilitate the transition from first-order to more expressive settings.

Research questions that were discussed and answered at the seminar included the following:

- What higher-order features do applications need, and what features can be incorporated smoothly in existing first-order proof calculi and provers?
- How can we best extend first-order reasoning techniques beyond first-order logic?
- Can proof-assistant-style automation and first-order reasoning techniques be combined in a synergetic fashion?
- What are good strategies for automatic induction and coinduction or invariant synthesis?
- Is a higher layer of reasoning, in the spirit of proof planning, necessary to solve more difficult higher-order problems?

2 Table of Contents

3 Overview of Talks

3.1 What QFBAPA can do for Description Logics

Franz Baader (TU Dresden, DE)

Joint work of Franz Baader, Andreas Ecke
Main reference Franz Baader, Andreas Ecke: "Extending the Description Logic ALC with More Expressive Cardinality Constraints on Concepts", in Proc. of the 3rd Global Conf. on Artificial Intelligence (GCAI 2017), EPiC Series in Computing, Vol. 50, pp. 6–19, EasyChair, 2017.
URL http://dx.doi.org/10.29007/f3hh

Considered from an abstract point of view, Description Logics (DLs) allow their users to state inclusion constraints between concepts (i.e., sets) and to state cardinality constraints for concepts and role successors. The constraints that can be formulated in DLs are usually of a very restricted form. We show that, by using the quantifier-free fragment of Boolean Algebra with Presburger Arithmetic (QFBAPA) to formulate constraints on sets and their cardinalities, we can considerably extend the expressive power without increasing the complexity of reasoning.

3.2 Automating Free Logic in HOL, with an Experimental Application in Category Theory

Christoph Benzmüller (FU Berlin, DE)

Joint work of Christoph Benzmüller, Dana S. Scott
Main reference Christoph Benzmüller, Dana S. Scott: "Axiomatizing Category Theory in Free Logic", CoRR, Vol. abs/1609.01493, 2016.
URL http://arxiv.org/abs/1609.01493

A shallow semantical embedding of free logic in classical higher-order logic is presented, which enables the off-the-shelf application of higher-order interactive and automated theorem provers for the formalisation and verification of free logic theories. Subsequently, this approach is applied to a selected domain of mathematics: starting from a generalization of the standard axioms for a monoid a stepwise development of various, mutually equivalent foundational axiom systems for category theory is presented. As a side-effect of this work some (minor) issue in a prominent category theory textbook has been revealed.

The purpose of this work is not to claim any novel results in category theory, but to demonstrate an elegant way to "implement" and utilize interactive and automated reasoning in free logic, and to present illustrative experiments.

References

1 Christoph Benzmüller and Dana Scott. Automating free logic in Isabelle/HOL. In G.-M. Greuel, T. Koch, P. Paule, and A. Sommese, editors, *Mathematical Software – ICMS 2016, 5th International Congress, Proceedings*, volume 9725 of *LNCS*, pages 43–50, Berlin, Germany, 2016. Springer. URL: http://christoph-benzmueller.de/papers/C57.pdf, doi:10.1007/978-3-319-42432-3_6.

2 Christoph Benzmüller and Dana S. Scott. Axiomatizing category theory in free logic. *arXiv, http://arxiv.org/abs/1609.01493*, 2016.

3 Christoph Benzmüller. Universal Reasoning, Rational Argumentation and Human-Machine Interaction. *arXiv, http://arxiv.org/abs/1703.09620*, 2017.

3.3 Towards Strong Higher-Order Automation for Fast Interactive Verification

Jasmin Christian Blanchette (Vrije Universiteit Amsterdam, NL)

We believe that first-order automatic provers are the best tools available to perform most of the tedious logical work inside proof assistants. From this point of view, it seems desirable to enrich superposition and SMT (satisfiability modulo theories) with higher-order reasoning in a careful manner, to preserve their good properties. Representative benchmarks from the interactive theorem proving community can guide the design of proof rules and strategies. With higher-order superposition and higher-order SMT in place, highly automatic provers could be built on modern superposition provers and SMT solvers, following a stratified architecture reminiscent of that of modern SMT solvers. We hope that these provers will bring a new level of automation to the users of proof assistants. These challenges and work plan are at the core of the Matryoshka project, funded for five years by the European Research Council. We encourage researchers motivated by the same goals to get in touch with us, subscribe to our mailing list, and join forces.

3.4 Building a Proof Checker with Partial Functions

Hans de Nivelle (University of Wroclaw, PL)

Main reference Hans de Nivelle: "Theorem proving for classical logic with partial functions by reduction to Kleene logic", J. Log. Comput., Vol. 27(2), pp. 509–548, 2017.
URL http://dx.doi.org/10.1093/logcom/exu071

In 2010–2013 I developed a 3-valued logic for partial functions. During 2013–2014, I tried to integrate this logic into an interactive proof checker. This attempt was unsuccessful. The system worked, but the main goal, to obtain a truly user friendly proof checker, was not obtained. In this talk, I summarize a new attempt, which has not been implemented yet, hoping to get feedback. I discuss the following components:

- The basics of the underlaying 3-valued logic and how to generalize this logic to higher-order.
- How I think that one should build theories (using the little theory approach of Farmer, Guttman and Thayer). Theories can be substantive or adjective in nature.
- Type Reductions. Explicit type conditions are useful, but turned out unpleasant in practical use, especially in higher-order. I explain how conventional, more user-friendly type declarations can be translated into explicit type declarations by means of reduction.

3.5 Scalable Fine-Grained Proofs for Formula Processing

Pascal Fontaine (LORIA & Inria – Nancy, FR)

License Creative Commons BY 3.0 Unported license
© Pascal Fontaine
Joint work of Haniel Barbosa, Jasmin Christian Blanchette, Simon Cruanes, Daniel El Ouraoui, Pascal Fontaine
Main reference Haniel Barbosa, Jasmin Christian Blanchette, Pascal Fontaine: "Scalable Fine-Grained Proofs for Formula Processing", in Proc. of the Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10395, pp. 398–412, Springer, 2017.
URL http://dx.doi.org/10.1007/978-3-319-63046-5_25

We presented a framework for processing formulas in automatic theorem provers, with generation of detailed proofs. The main components are a generic contextual recursion algorithm and an extensible set of inference rules. Clausification, skolemization, theory-specific simplifications, and expansion of 'let' expressions, and beta-reduction are instances of this framework. With suitable data structures, proof generation adds only a linear-time overhead, and proofs can be checked in linear time. We implemented the approach in the SMT solver veriT. This allowed us to dramatically simplify the code base while increasing the number of problems for which detailed proofs can be produced, which is important for independent checking and reconstruction in proof assistants. This talk presented material accepted at CADE 2017 and at PxTP 2017.

3.6 Harnessing First Order Termination Provers Using Higher Order Dependency Pairs

Carsten Fuhs (Birkbeck, University of London, GB)

License Creative Commons BY 3.0 Unported license
© Carsten Fuhs
Joint work of Carsten Fuhs, Cynthia Kop
Main reference Carsten Fuhs, Cynthia Kop: "Harnessing First Order Termination Provers Using Higher Order Dependency Pairs", in Proc. of the Frontiers of Combining Systems, 8th International Symposium, FroCoS 2011, Saarbrücken, Germany, October 5-7, 2011. Proceedings, Lecture Notes in Computer Science, Vol. 6989, pp. 147–162, Springer, 2011.
URL https://doi.org/10.1007/978-3-642-24364-6_11

Many functional programs and higher order term rewrite systems contain, besides higher order rules, also a significant first order part. We discuss how an automatic termination prover can split a rewrite system into a first order and a higher order part. The results are applicable to all common styles of higher order rewriting with simple types, although some dependency pair approach is needed to use them.

This talk is based on joint work with Cynthia Kop. A corresponding paper has appeared in the proceedings of FroCoS 2011.

3.7 Automated Complexity Analysis for Java Programs

Jürgen Giesl (RWTH Aachen, DE) and Florian Frohn

Joint work of Florian Frohn, Jürgen Giesl
Main reference Florian Frohn, Jürgen Giesl: "Complexity Analysis for Java with AProVE", in Proc. of the Integrated Formal Methods - 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10510, pp. 85–101, Springer, 2017.
URL https://doi.org/10.1007/978-3-319-66845-1_6

Automated termination analysis is an important area in program verification which goes beyond classical first-order reasoning. While AProVE is one of the most powerful tools for termination analysis of Java since many years, we now extend our technique in order to analyze the complexity of Java programs as well.

Our approach first executes the program symbolically on an abstract domain which uses heap predicates in addition to the usual first-order constructs. Based on this symbolic execution, we develop a novel transformation of (possibly heap-manipulating) Java programs to integer transition systems (ITSs). This allows us to apply existing complexity analyzers for standard first-order ITSs in order to infer runtime bounds for Java programs. We demonstrate the power of our implementation on an established benchmark set.

3.8 Why user experiments matter for automated reasoning

Reiner Hähnle (TU Darmstadt, DE)

Joint work of Martin Hentschel, Reiner Hähnle, Richard Bubel
Main reference Martin Hentschel, Reiner Hähnle, Richard Bubel: "An empirical evaluation of two user interfaces of an interactive program verifier", in Proc. of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016, pp. 403–413, ACM, 2016.
URL http://dx.doi.org/10.1145/2970276.2970303

I argue why empirical research, such as experimental studies, are a valuable form of contribution in automated reasoning and should have a place in our conferences and journals.

3.9 Automating Proofs by (co)-Induction and Theory Exploration

Moa Johansson (Chalmers University of Technology – Göteborg, SE)

Joint work of Moa Johansson, Nicholas Smallbone, Koen Claessen, Dan Rosen, Irene Lobo Valbuena
Main reference Moa Johansson: "Automated Theory Exploration for Interactive Theorem Proving: - An Introduction to the Hipster System", in Proc. of the Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10499, pp. 1–11, Springer, 2017.
URL http://dx.doi.org/10.1007/978-3-319-66107-0_1

One of the more challenging aspects in automating all but the simplest inductive proofs is how to discover auxiliary lemmas. In our recent work, we have taken a "bottom-up" approach to lemma discovery using theory exploration. Theory exploration is a technique for automatically discovering interesting lemmas using testing. A richer background theory can then be constructed, allowing harder theorems to be proved automatically. I will show a demo of our theory exploration system Hipster for Isabelle/HOL, and explain a bit about how it works.

Earlier work on lemma discovery by proof-planning critics took the opposite "top-down" approach: here proof failures were analysed in an attempt to patch the failed proof. This worked very well for many cases were the missing lemma was a simple generalisation of the stuck proof state (called lemma calculation), but less well when the required lemma for instance was a generalisation of the original conjecture.

I believe lemma discovery by theory exploration could fit very nicely in with systems like Sledgehammer. It can work as a complement when useful facts are missing from the available libraries, for example in new theory developments. Unlike proof-critics, it is not dependent on particular proof-planning heuristics and systems (like rippling), and could therefore more easily be used in conjunction with first- or higher-order automated provers.

3.10 What else can automation do for proof assistants

Cezary Kaliszyk (Universität Innsbruck, AT)

Joint work of Łukasz Czajka, Thibault Gauthier, Cezary Kaliszyk

I this talk I will present the progress in automation for proof assistants. I will introduce the hammer for Coq, which can now re-prove 40% of the theorems in the Coq standard library fully automatically. I will discuss combining hammer-style premise selection with learning to use tactics and discuss automation optimizations for reasoning about types in a logical framework.

References

1 Łukasz Czajka and Cezary Kaliszyk. Hammer for Coq: Automation for Dependent Type Theory. Submitted, 2017

2 Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. TacticToe: Learning to reason with HOL4 Tactics. LPAR 2017, volume 46 of EPiC, pp. 125–143. 2017

3.11 Efficient Interpolant generation algorithms based on quantifier elimination: EUF, Octagons, ...

Deepak Kapur (University of New Mexico – Albuquerque, US)

In a paper in 2006, Kapur, Majumdar and Zarba observed a connection between quantifier elimination and interpolant generation which was probably well-known but not explicitly reported in the literature on automated reasoning and formal methods. Since then I have been investigating how to develop heuristics for quantifier elimination to generate interpolants. Particularly, there is no need to have access to a proof to generate interpolants, a methodology widely used in the formal methods community.

I will start with an interpolant generation algorithm in the quantifier-free theory of equality over uninterpreted symbols. Even though there are many algorithms reported in the literature, there is little investigation about their complexity. Interpolants generated are simple and can be efficiently represented using new symbols defined in terms of common symbols. This is followed by an interpolant generation algorithm for octagonal formulas, which is of complexity $O(n^3)$, where n is the number of variables; an interpolant generated is a conjunction of octagonal formulas. Combination methods for interpolant generation over subtheories can be developed as well. Another interesting outcome is an efficient algorithm for generating congruence closure of conditional equations.

3.12 Higher-order Term Rewriting

Cynthia Kop (Radboud University Nijmegen, NL)

One of the key problems in higher-order term rewriting is that there is no true consensus of what, exactly, “higher-order term rewriting” means. There are disagreements on the necessity of including types and/or binders, and various–sometimes incompatible–definitions.

In this talk, I have discussed a number of different styles of higher-order term rewriting, their strengths and weaknesses, and the rough differences between them. I have also discussed some of the technology for proving termination, in particular the notion of computability.

3.13 An Abstraction-Refinement Framework for Reasoning with Large Theories

Konstantin Korovin (University of Manchester, UK)

Joint work of Konstantin Korovin, Julio Cesar Lopez Hernandez
Main reference Julio Cesar Lopez Hernandez, Konstantin Korovin: “Towards an Abstraction-Refinement Framework for Reasoning with Large Theories”, in IWIL@LPAR 2017, Vol. 1, pp. 119-123, Kalpa Publications in Computing, EasyChair, 2017.
URL https://doi.org/10.29007/4zh8

We presented an approach to reasoning with large theories which is based on the abstraction-refinement framework [1]. The proposed approach consists of over-approximations, under-approximations and their combination. We discussed different abstractions and refinement strategies for reasoning with large first-order theories.

References

1 Julio Cesar Lopez Hernandez and Konstantin Korovin. Towards an Abstraction-Refinement Framework for Reasoning with Large Theories. IWIL@LPAR 2017, vol. 1, Kalpa Publications in Computing, EasyChair, 2017.

3.14 Constrained Resolution via (Almost) First-order Theorem Provers

Tomer Libal (Inria Saclay – Île-de-France, FR)

When considering how to use techniques from first-order theorem proving in higher-order provers, the ideal would be to use the first-order theorem provers themselves. In order to deal with the complexities which arise when dealing with higher-order terms, these provers are sometimes being applied in a constrained manner within higher-order ones. We consider a possible approach of isolating the (almost) first-order content of higher-order formulae by pre-processing and then using existing first-order provers in order to obtain a (partial) proof. This proof will be pending the successful discharge of constraints generated in the pre-processing step. An advantage of this approach is its ability to use the full spectrum of capabilities of first-order theorem provers, such as indexing, redundancy elimination, etc.

3.15 Root-balanced Trees: Verified Algorithms Analysis

Tobias Nipkow (TU München, DE)

Main reference Tobias Nipkow: "Verified Root-Balanced Trees", in Proc. of the Programming Languages and Systems - 15th Asian Symposium, APLAS 2017, Suzhou, China, November 27-29, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10695, pp. 255–272, Springer, 2017.
URL http://dx.doi.org/10.1007/978-3-319-71237-6_13

This talk presents recent work on verifying complexity of functional programs in Isabelle/HOL [1, 2, 3]. The focus of the presentation will be on the amortized complexity of a brand of search trees (invented by Andersson) where rebalancing happens only when the tree becomes badly unbalanced at the root. This is accompanied by a general discussion on modelling techniques for timing analysis and on automatic proofs of functional correctness.

References

1 Tobias Nipkow. Amortized Complexity Verified. Interactive Theorem Proving 2015, LNCS 9236, Springer, 2015.
2 Tobias Nipkow. Automatic Functional Correctness Proofs for Functional Search Trees. Interactive Theorem Proving 2016, LNCS 9807, Springer, 2016.
3 Tobias Nipkow. Verified Root-Balanced Trees. Asian Symposium on Programming Languages and Systems 2017, LNCS, Springer, 2017.

3.16 Difference between Program Verification via Hoare Logic and Rewriting Induction

Naoki Nishida (Nagoya University, JP)

Joint work of Naoki Nishida, Shinnosuke Mizutani
Main reference Shinnosuke Mizutani, Naoki Nishida, "Transforming Proof Tableaux of Hoare Logic into Inference Sequences of Rewriting Induction", Workshop on Rewriting Techniques for Program Transformations and Evaluation, Oxford, UK, September 8, 2017.
URL https://www.cs.ox.ac.uk/conferences/fscd2017/preproceedings_unprotected/WPTE_Mizutani.pdf

In this talk, I first introduce rewriting induction on constrained term rewriting and then introduce a transformation of a proof tableau of Hoare logic into an inference sequence of constrained rewriting induction. Finally, I discuss difference between program verification via these two approaches.

3.17 Featherweight alias control using types

Andrei Paskevich (University of Paris Sud – Orsay, FR)

Joint work of Andrei Paskevich, Léon Gondelman, Jean-Christophe Filliâtre
Main reference Jean-Christophe Filliâtre, Léon Gondelman, Andrei Paskevich, "A Pragmatic Type System for Deductive Verification", Technical report, Inria hal-01256434, 2016.
URL https://hal.inria.fr/hal-01256434

In the context of deductive verification, it is customary today to handle programs with pointers using either separation logic, dynamic frames, or explicit memory models. Yet we can observe that in numerous programs, a large amount of code fits within the scope of Hoare logic, provided we can statically control aliasing. When this is the case, the code correctness can be reduced to simpler verification conditions which do not require any explicit memory model. This makes verification conditions more amenable both to automated theorem proving and to manual inspection and debugging.

In this talk, we show a method of such static aliasing control for a programming language featuring nested data structures with mutable components. Our solution is based on a type system with singleton regions and effects.

3.18 Automating Separation Logic Reasoning using SMT Solvers

Ruzica Piskac (Yale University – New Haven, US)

Main reference Ruzica Piskac, Thomas Wies, Damien Zufferey: "Automating Separation Logic Using SMT", in Proc. of the Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings, Lecture Notes in Computer Science, Vol. 8044, pp. 773–789, Springer, 2013.
URL http://dx.doi.org/10.1007/978-3-642-39799-8_54

Separation logic (SL) follows a discipline of local reasoning that mimics human intuition about how to prove the correctness of heap-manipulating programs. Central to this discipline is the frame rule, a Hoare logic proof rule that decomposes the global heap into a footprint, the region on which a program fragment operates, and a frame, the region that remains untouched by the program fragment. Automation of the frame rule involves the actual inference of the frame from SL assertions expressing the global heap and the footprint.

In this talk, I present reductions of decidable separation logic fragments to decidable first-order theories that fit well into the SMT framework. We show how these reductions can be used to automate satisfiability, entailment, frame inference, and abduction problems for separation logic using SMT solvers. Our approach provides a simple method of integrating separation logic into existing verification tools that provide SMT backends, and an elegant way of combining separation logic fragments with other decidable first-order theories.

3.19 Friends with benefits: Coinduction and corecursion in Isabelle/HOL

Andrei Popescu (Middlesex University – London, GB)

Joint work of Andrei Popescu, Jasmin Blanchette, Dmitriy Traytel, Aymeric Bouzy, Andreas Lochbihler, and others
Main reference Jasmin Christian Blanchette, Aymeric Bouzy, Andreas Lochbihler, Andrei Popescu, Dmitriy Traytel: "Friends with Benefits - Implementing Corecursion in Foundational Proof Assistants", in Proc. of the Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10201, pp. 111–140, Springer, 2017.
URL http://dx.doi.org/10.1007/978-3-662-54434-1_5

Isabelle/HOL has been recently endowed with an infrastructure for coinductive datatypes (codatatypes), corecursive functions and coinductive proofs. A codatatype's corecursion and coinduction schemes evolve in tandem by learning of new "friendly" operators from the user.

3.20 Fast and Slow Synthesis Procedures in SMT

Andrew Joseph Reynolds (University of Iowa – Iowa City, US)

Joint work of Andrew Joseph Reynolds, Cesare Tinelli, Clark Barrett, Viktor Kuncak, Morgan Deters, Tim King
Main reference Andrew Reynolds, Morgan Deters, Viktor Kuncak, Cesare Tinelli, Clark W. Barrett: "Counterexample-Guided Quantifier Instantiation for Synthesis in SMT", in Proc. of the Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 9207, pp. 198–216, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-21668-3_12

Recent techniques for automated synthesis in SMT solvers follow two paradigms. The first is based on first-order quantifier instantiation, and can be used to tackle a restricted but fairly common class of properties, known as single invocation properties. The second relies on a deep embedding of the synthesis problem into the theory of inductive datatypes, which can then be solved using enumerative syntax-guided techniques. This talk focuses on the advantages and disadvantages of these two paradigms, and how they can potentially be combined.

3.21 Synthesising Regular Sets and Relations with a SAT Solver

Philipp Rümmer (Uppsala University, SE)

Joint work of Ondrej Lengál, Anthony Widjaja Lin, Rupak Majumdar, Philipp Rümmer
Main reference Anthony W. Lin, Philipp Rümmer: "Liveness of Randomised Parameterised Systems under Arbitrary Schedulers", in Proc. of the Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 9780, pp. 112–133, Springer, 2016.
URL https://doi.org/10.1007/978-3-319-41540-6_7

We consider the problem of verifying liveness for systems with a finite, but unbounded, number of processes, commonly known as parameterised systems. Typical examples of such systems include distributed protocols (e.g., for the dining philosopher problem). Unlike the case of verifying safety, proving liveness is still considered extremely challenging, especially in the presence of randomness in the system. We introduce an automatic method of proving liveness for randomised parameterised systems under arbitrary schedulers. Viewing liveness as a two-player reachability game (between Scheduler and Process), our method is a CEGAR approach that synthesises a progress relation for Process that can be symbolically represented as a finite-state automaton. The method constructs a progress relation by means of a suitable Boolean encoding and incremental SAT solving. Our experiments show that our algorithm is able to prove liveness automatically for well-known randomised distributed protocols, including Lehmann-Rabin Randomised Dining Philosopher Protocol and randomised self-stabilising protocols (such as the Israeli-Jalfon Protocol). To the best of our knowledge, this is the first fully-automatic method that can prove liveness for randomised protocols.

3.22 Automated Forgetting, Uniform Interpolation and Second-Order Quantifier Elimination

Renate Schmidt (University of Manchester, GB)

Joint work of Renate Schmidt, Andrzej Szalas, Patrick Koopmann, Yizheng Zhao

Forgetting transforms a knowledge base into a compact representation by eliminating undesired symbols, which allows users to focus on specific parts of ontologies in order to create decompositions and restricted views for in depth analysis or sharing with other users. Forgetting is also useful for information hiding, explanation generation, semantic difference computation and ontology debugging. Other names for forgetting are: second-order quantifier elimination uniform interpolation, variable elimination, predicate elimination, and projection. Because forgetting is an inherently difficult problem – it is much harder than standard reasoning (satisfiability and validity testing) – and very few logics are known to be complete for forgetting (or have the uniform interpolation property), there has been insufficient research on the topic and few forgetting tools are available.

In my presentation gave a brief overview of the methods and success stories of three forgetting tools: SCAN which performs second-order quantifier elimination [1, 2], LETHE which solves the uniform interpolation problem for many expressive description problems extending ALC [3, 4], and FAME which computes semantic forgetting solutions for description logics of different expressivity [5, 6].

References

1 Gabbay, D. M. and Ohlbach, H. J. (1992), Quantifier Elimination in Second-Order Predicate Logic. *South African Computer Journal* 7, 35–43.

2 Gabbay, D. M., Schmidt, R. A. and Szałas, A. (2008), Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications. Studies in Logic: Mathematical Logic and Foundations 12, College Publications.

3 Koopmann, P. and Schmidt, R. A. (2013), Forgetting Concept and Role Symbols in $\mathcal{ALCH}$-Ontologies. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2013)*. Lecture Notes in Computer Science 8312, Springer, 552–567.

4 Koopmann, P. and Schmidt, R. A. (2014), Count and Forget: Uniform Interpolation of $\mathcal{SHQ}$-Ontologies. In *Automated Reasoning (IJCAR 2014)*. Lecture Notes in Artificial Intelligence 8562, Springer, 434–448.

5 Zhao, Y. and Schmidt, R. A. (2016), Forgetting Concept and Role Symbols in $\mathcal{ALCOIH}\mu^{+}(\nabla,\sqcap)$-Ontologies. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press/IJCAI, 1345–1352.

6 Zhao, Y. and Schmidt, R. A. (2017), Role Forgetting for $\mathcal{ALCOQH}(\nabla)$-Ontologies Using an Ackermann Approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press/IJCAI, 1354–1361.

3.23 Towards a classification of ATP proof tasks (part 2)

Stephan Schulz (Duale Hochschule Baden-Württemberg – Stuttgart, DE)

Automated theorem provers for first-order logic search for proofs in an infinite and highly branching search space. To be successful, they critically depend on various search heuristics or strategies. Experience shows that different strategies perform well on different problems. In this work, we try to automate the process of assigning a good strategy for a given problem via machine learning. In a first step, we use extensive test data to automatically cluster problems into classes showing similar behaviour under different strategies, using a combination of PCA for dimensionality reduction and k-means clustering to group similar problems. In a second step, we then learn properties of these clusters using standard machine learning techniques and a set of signature-structural features. Initial results already suggest better performance than the previous method of hand-selecting features and feature value splits for the classification.

3.24 Compositional entailment checking for theories based on separation logic

Mihaela Sighireanu (University Paris-Diderot, FR)

Joint work of Constantin Enea, Mihaela Sighireanu, and Zhilin Wu
Main reference Constantin Enea, Mihaela Sighireanu, Zhilin Wu: "On Automated Lemma Generation for Separation Logic with Inductive Definitions", in Proc. of the Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings, Lecture Notes in Computer Science, Vol. 9364, pp. 80–96, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-24953-7_7

The core of the SPEN solver is a semi-decision procedure for checking entailment between separation logic formulas with inductive predicates. In this talk, I'll briefly present this procedure and its extensions for theories combining separation logic with arithmetic, set, and inductive types constraints.

3.25 On Symbol Elimination in Theory Extensions

Viorica Sofronie-Stokkermans (Universität Koblenz-Landau, DE)

Main reference Viorica Sofronie-Stokkermans: "On Interpolation and Symbol Elimination in Theory Extensions", in Proc. of the Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings, Lecture Notes in Computer Science, Vol. 9706, pp. 273–289, Springer, 2016.
URL http://dx.doi.org/10.1007/978-3-319-40229-1_19

Many problems in computer science (e.g. in program verification) can be reduced to checking satisfiability of ground formulae w.r.t. a theory which can be a standard theory (for instance linear arithmetic) or a complex theory (typically the extension of a base theory $\mathcal{T}_0$ with additional function symbols axiomatized by a set $\mathcal{K}$ of formulae, or a combination of theories). SMT solvers are tuned for efficiently checking satisfiability of ground formulae in increasingly complex theories; the output can be "satisfiable", "unsatisfiable" – or possibly "unknown" if incomplete methods are used, or else termination cannot be guaranteed.

More interesting is to go beyond yes/no answers, i.e. to consider parametric systems – in which the parameters can be values or functions – and infer constraints on the parameters which guarantee that certain properties are met (for instance constraints which guarantee the unsatisfiability of certain formulae). Such constraints can be obtained by performing quantifier elimination or, more generally, symbol elimination.

In this talk we present a symbol elimination method in extensions of a theory $\mathcal{T}_0$ with additional function symbols whose properties are axiomatised using a set $\mathcal{K}$ of clauses. We analyze situations in which we can perform symbol elimination in a hierarchical way, relying on existing mechanisms for symbol elimination in $\mathcal{T}_0$. This is for instance possible if the theory $\mathcal{T}_0$ allows quantifier elimination. We present various applications of this method. The results are described in [1].

References

1 Sofronie-Stokkermans, V.: On interpolation and symbol elimination in theory extensions. In: Olivetti, N. and Tiwari, A. (eds), *Proc. IJCAR 2016*, volume 9706 of *LNCS*, pages 273–289, Springer (2016)

3.26 Flexible Theorem Proving in Modal Logics

Alexander Steen (FU Berlin, DE)

License Creative Commons BY 3.0 Unported license
© Alexander Steen
Joint work of Alexander Steen, Christoph Benzmüller, Alexander Steen, Tobias Gleißner
Main reference T. Gleißner, A. Steen, C. Benzmüller: "Theorem Provers For Every Normal Modal Logic," in Proc. of the 21st Int'l Conf. on Logic for Programming (LPAR-21), Artificial Intelligence and Reasoning, Maun, Botswana, 7-12th May 2017, EPiC Series in Computing, Vol. 46, pp. 14–30, EasyChair, 2017.
URL https://doi.org/10.29007/jsb9

Computer-assisted reasoning in non-classical logics is of increasing interest in artificial intelligence (AI), computer science, mathematics and philosophy. Several powerful automated and interactive theorem proving systems have been developed over the past decades. However, with a few exceptions, most of the available systems focus on classical logics only. In particular for quantified variants there are only few systems available to date. In this talk, I present a uniform automation approach for a wide range of different modal logics. It is based on a shallow embedding into classical higher-order logic and can flexibly account for semantical variations of the desired modal logic at hand. Based on a specification of the modal logic's semantics, a procedure is presented that algorithmically translates the source problem into a classical (non-modal) HOL problem. This procedure was implemented within Leo-III and as a stand-alone pre-processing tool, ready to use in conjunction with any THF-compliant theorem prover. The choice of the concrete modal logic is thereby specified within the problem as a meta-logical statement. By combining our tool with one or more THF-compliant theorem provers we accomplish the most widely applicable modal logic theorem prover available to date, i.e. no other available prover covers more variants of propositional and quantified modal logics. Despite this generality, our approach remains competitive, at least for quantified modal logics, as our experiments demonstrate.

References

1 Christoph Benzmüller and Alexander Steen and Max Wisniewski. Leo-III Version 1.1 (System description), In IWIL Workshop and LPAR Short Presentations, EasyChair, Kalpa Publications in Computing, Volume 1, pp. 11-26, 2017.

3.27 Cyclic Proofs with Ordering Constraints

Sorin Stratulat (University of Lorraine – Metz, FR)

License Creative Commons BY 3.0 Unported license
© Sorin Stratulat
Main reference Sorin Stratulat: "Cyclic Proofs with Ordering Constraints", in Proc. of the Automated Reasoning with Analytic Tableaux and Related Methods - 26th International Conference, TABLEAUX 2017, Brasília, Brazil, September 25-28, 2017, Proceedings, Lecture Notes in Computer Science, Vol. 10501, pp. 311–327, Springer, 2017.
URL http://dx.doi.org/10.1007/978-3-319-66902-1_19

CLKID$^\omega$ is a sequent-based cyclic inference system able to reason on first-order logic with inductive definitions. The current approach for verifying the soundness of CLKID$^\omega$ proofs is based on expensive model-checking techniques leading to an explosion in the number of states.

We propose proof strategies that guarantee the soundness of a class of CLKID$^\omega$ proofs if some ordering and derivability constraints are satisfied. They are inspired from previous works about cyclic well-founded induction reasoning, known to provide effective sets of ordering constraints. A derivability constraint can be checked in linear time. Under certain conditions, one can build proofs that implicitly satisfy the ordering constraints.

3.28 Symbolic Execution and Program Synthesis

Thomas Ströder (Metro Systems GmbH – Düsseldorf, DE)

Joint work of Thomas Ströder, Jürgen Giesl, Marc Brockschmidt, Florian Frohn, Carsten Fuhs, Jera Hensel, Peter Schneider-Kamp, Cornelius Aschermann
Main reference Thomas Ströder, Jürgen Giesl, Marc Brockschmidt, Florian Frohn, Carsten Fuhs, Jera Hensel, Peter Schneider-Kamp, Cornelius Aschermann: "Automatically Proving Termination and Memory Safety for Programs with Pointer Arithmetic", J. Autom. Reasoning, Vol. 58(1), pp. 33–65, 2017.
URL http://dx.doi.org/10.1007/s10817-016-9389-x

Symbolic execution is a very powerful and flexible technique to obtain abstract representations of program behaviors. From the abstraction, we synthesize programs in simple formal languages for which sophisticated analyses of the properties we are interested in exist (of course, the program synthesis must retain all relevant properties such that results for the analyzed programs carry over to the original programs). Using this approach, we can reduce higher-order reasoning problems to pure first-order reasoning. We illustrate this approach by an example termination analysis of the strlen C program and give a brief outlook why METRO is interested in such research topics.

3.29 Recent Improvements of Theory Reasoning in Vampire

Martin Suda (TU Wien, AT)

Joint work of Giles Reger, Martin Suda, Andrei Voronkov
Main reference M. Suda, G. Reger, A. Voronkov: "Unification with abstraction and theory instantiation in saturation-based reasoning", EasyChair Preprint no. 1, EasyChair, 2017.
URL https://doi.org/10.29007/hsh2

Over the past years Vampire has been progressively improving its ability to reason with quantifiers and theories. Originally theory reasoning was only via theory axioms and evaluation but over the past year two new techniques have been introduced. The first is the recent work of AVATAR modulo theories, previously presented, for ground theory reasoning. The second, the focus of this talk, consists of two new methods for reasoning with non-ground theory clauses (where we currently focus on the theory of arithmetic). The first new method is *unification with abstraction* where the notion of unification is extended to introduce constraints where theory terms may not otherwise unify, e.g., $p(2)$ may unify with $\neg p(x+1) \vee q(x)$ to produce $2 \neq x+1 \vee q(x)$. This abstraction is performed lazily, as needed, to allow the superposition theorem prover to make as much progress as possible without the search space growing too quickly. The second new method utilises theory constraint solving (an SMT solver) to perform reasoning within a clause to find an instance where we can remove theory literals. This utilises the power of SMT solvers for theory reasoning with non-ground clauses, reasoning which is currently achieved by the addition of prolific theory axioms. Additionally, this second method can be used to discharge the constraints introduced by unification with abstraction. These methods were implemented within the Vampire theorem prover and experimental results show that they are useful for solving currently unsolved problems.

3.30 SMT-LIB 3: Bringing higher-order logic to SMT

Cesare Tinelli (University of Iowa – Iowa City, US)

Joint work of Clark Barrett, Pascal Fontaine, Cesare Tinelli

The SMT-LIB standard defines a common input/output language of commands to communicate with solvers for Satisfiability Modulo Theories (SMT) via a textual interface. The widely adopted most recent version of the standard, Version 2.6, is based on an extension of many-sorted first-order logic. Historically, has been adequate in most cases because SMT solvers are themselves based on automated reasoning techniques for first-order logic. A growing number of tools (interactive theorem provers, in particular) that leverage the power of SMT solvers, however, are based on more powerful logics. This forces the developers of these tools to implement often complex encodings of their problems in the less powerful logic of SMT-LIB 2. Given the interest of some SMT solver developers in extending their tools to higher-order logics, it would be beneficial for the field to extend the SMT-LIB 2 standard to some basic higher-order logic. This would simplify current encodings to SMT and might also improve runtime performance. This talk proposes a higher-order version of SMT-LIB based on simple type theory with rank-1 polymorphism. A distinguishing feature of the new version is that it is largely backward compatible with SMT-LIB 2, which means that applications and solvers not interested to the higher-order logic extensions are not affected. Non-backward-compatible portions are essentially orthogonal to the higher-order logic extension. They address other shortcomings of the current standard related to the way a user can specify the particular logical fragment the input problem belongs to.

3.31 Beyond Deduction

Josef Urban (Czech Technical University – Prague, CZ)

Joint work of Josef Urban, Thibault Gauthier, Jan Jakubuv, Cezary Kaliszyk, Jiri Vyskocil
Main reference Cezary Kaliszyk, Josef Urban: "Learning-Assisted Automated Reasoning with Flyspeck", J. Autom. Reasoning, Vol. 53(2), pp. 173–213, 2014.
URL http://dx.doi.org/10.1007/s10817-014-9303-3

The talk will describe several ways of applying machine learning methods in theorem proving and some ways of combining learning and deduction in feedback loops.

References

1 Jan Jakubuv, Josef Urban: ENIGMA: Efficient Learning-Based Inference Guiding Machine. CICM 2017: 292-302. 2017
2 Cezary Kaliszyk, Josef Urban, Jiri Vyskocil: Automating Formalization by Statistical and Semantic Parsing of Mathematics. ITP 2017: 12-27. 2017

3 Jan Jakubuv, Josef Urban: BliStrTune: hierarchical invention of theorem proving strategies. CPP 2017: 43-52. 2017
4 Thibault Gauthier, Cezary Kaliszyk, Josef Urban: Initial Experiments with Statistical Conjecturing over Large Formal Corpora. FM4M/MathUI/ThEdu/DP/WIP@CIKM 2016: 219-228. 2016
5 Cezary Kaliszyk, Josef Urban: MizAR 40 for Mizar 40. J. Autom. Reasoning 55(3): 245-256 (2015)
6 Cezary Kaliszyk, Josef Urban: Learning-Assisted Automated Reasoning with Flyspeck. J. Autom. Reasoning 53(2): 173-213 (2014)

Participants

- Franz Baader
TU Dresden, DE
- Christoph Benzmüller
FU Berlin, DE
- Nikolaj S. Bjorner
Microsoft Corporation – Redmond, US
- Jasmin Christian Blanchette
VU University of Amsterdam, NL
- James Brotherston
University College London, GB
- Chad E. Brown
Czech Technical University – Prague, CZ
- Hans de Nivelle
University of Wroclaw, PL
- Pascal Fontaine
LORIA & INRIA – Nancy, FR
- Carsten Fuhs
Birkbeck, University of London, GB
- Jürgen Giesl
RWTH Aachen, DE
- Georges Gonthier
INRIA Saclay – Île-de-France, FR
- Reiner Hähnle
TU Darmstadt, DE
- Swen Jacobs
Universität des Saarlandes, DE
- Moa Johansson
Chalmers University of Technology – Göteborg, SE
- Cezary Kaliszyk
Universität Innsbruck, AT
- Deepak Kapur
University of New Mexico – Albuquerque, US
- Chantal Keller
University of Paris Sud – Orsay, FR
- Cynthia Kop
University of Copenhagen, DK
- Konstantin Korovin
University of Manchester, GB
- K. Rustan M. Leino
Microsoft Corporation – Redmond, US
- Tomer Libal
INRIA Saclay – Île-de-France, FR
- Tobias Nipkow
TU München, DE
- Naoki Nishida
Nagoya University, JP
- Andrei Paskevich
University of Paris Sud – Orsay, FR
- Ruzica Piskac
Yale University – New Haven, US
- Andrei Popescu
Middlesex University – London, GB
- Andrew Joseph Reynolds
University of Iowa – Iowa City, US
- Philipp Rümmer
Uppsala University, SE
- Renate Schmidt
University of Manchester, GB
- Stephan Schulz
Duale Hochschule Baden-Württemberg – Stuttgart, DE
- Thomas Sewell
Data61 – Sydney, AU
- Natarajan Shankar
SRI – Menlo Park, US
- Mihaela Sighireanu
University Paris-Diderot, FR
- Viorica Sofronie-Stokkermans
Universität Koblenz-Landau, DE
- Alexander Steen
FU Berlin, DE
- Sorin Stratulat
University of Lorraine – Metz, FR
- Thomas Ströder
Metro Systems GmbH – Düsseldorf, DE
- Martin Suda
TU Wien, AT
- Laurent Théry
INRIA Sophia Antipolis, FR
- Cesare Tinelli
University of Iowa – Iowa City, US
- Josef Urban
Czech Technical University – Prague, CZ
- Christoph Weidenbach
MPI für Informatik – Saarbrücken, DE