Report from Dagstuhl Seminar 15381

Information from Deduction: Models and Proofs

Edited by

Nikolaj S. Bjørner[1], Jasmin Christian Blanchette[2], Viorica Sofronie-Stokkermans[3], and Christoph Weidenbach[4]

1 **Microsoft Corporation – Redmond, US,** nbjorner@microsoft.com
2 **INRIA Lorraine – Nancy, FR,** jasmin.blanchette@inria.fr
3 **Universität Koblenz-Landau, DE,** sofronie@uni-koblenz.de
4 **MPI für Informatik – Saarbrücken, DE,** weidenbach@mpi-inf.mpg.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 15381 "Information from Deduction: Models and Proofs". The aim of the seminar was to bring together researchers working in deduction and applications that rely on models and proofs produced by deduction tools. Proofs and models serve two main purposes: (1) as an upcoming paradigm towards the next generation of automated deduction tools where search relies on (partial) proofs and models; (2) as the actual result of an automated deduction tool, which is increasingly integrated into application tools. Applications are rarely well served by a simple yes/no answer from a deduction tool. Many use models as certificates for satisfiability to extract feasible program executions; others use proof objects as certificates for unsatisfiability in the context of high-integrity systems development. Models and proofs even play an integral role within deductive tools as major methods for efficient proof search rely on refining a simultaneous search for a model or a proof. The topic is in a sense evergreen: models and proofs will always be an integral part of deduction. Nonetheless, the seminar was especially timely given recent activities in deduction and applications, and it enabled researchers from different subcommunities to communicate with each other towards exploiting synergies.

Seminar September 13–18, 2015 – http://www.dagstuhl.de/15381
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.5.9.18
Edited in cooperation with Carsten Fuhs

1 Executive Summary

Nikolaj S. Bjørner
Jasmin Christian Blanchette
Viorica Sofronie-Stokkermans
Christoph Weidenbach

Models and proofs are the quintessence of logical analysis and argumentation. Many applications of deduction tools need more than a simple answer whether a conjecture holds;

often additional information – for instance proofs or models – can be extremely useful. For example, proofs are used by high-integrity systems as part of certifying results obtained from automated deduction tools, and models are used by program analysis tools to represent bug traces. Most modern deductive tools may be trusted to also produce a proof or a model when answering whether a conjecture is a theorem or whether a certain problem formalized in logic has a solution. Moreover, major progress has been obtained recently by procedures that rely on refining a simultaneous search for a model and a proof. Thus, proofs and models help producing models and proofs, and applications use proofs and models in many crucial ways.

Below, we point out several directions of work related to models and proofs in which there are challenging open questions:

- *Extracting proofs from derivations.* An important use of proof objects from derivations is for applications that require certification. But although the format for proof objects and algorithms for producing and checking them has received widespread attention in the research community, the current situation is not satisfactory from a consumer's point of view.
- *Extracting models from derivations.* Many applications rely on models, and models are as important to certify non-derivability. Extracting models from first-order saturation calculi is a challenging problem: the well-known completeness proofs of superposition calculi produce perfect models from a saturated set of clauses. The method is highly non-constructive, so extracting useful information, such as "whether a given predicate evaluates to true or false under the given saturated clauses," is challenging. The question of representation is not yet well addressed for infinite models.
- *Using models to guide the search for proofs and vice versa.* An upcoming next generation of reasoning procedures employ (partial) models/proofs for proof search. They range from SAT to first-order to arithmetic reasoning and combinations thereof. It remains an open question what properties of models are crucial for successful proof search, how the models should be dynamically adapted to the actual problem, and how the interplay between the models and proof search progress through deduction should be designed.
- *External applications of models and proofs.* Models and proofs are used in various ways in applications. So far application logics and automated proof search logics have been developed widely independently. In order to get more of a coupling, efforts of bringing logics closer together or the search for adequate translations are needed.

This Dagstuhl seminar allowed to bring together experts for these topics and invited discussion about the production and consumption of proofs and models. The research questions pursued and answered include:

- To what extent is it possible to design common exchange formats for theories, proofs, and models, despite the diversity of provers, calculi, and formalisms?
- How can we generate, process, and check proofs and models efficiently?
- How can we search for, represent, and certify infinite models?
- How can we use models to guide proof search and proofs to guide model finding?
- How can we make proofs and models more intelligible, yet at the same time provide the level of detail required by certification processes?

2 Table of Contents

3 Overview of Talks

3.1 Formal Verification of Pastry Using TLA+

Noran Azmy (MPI für Informatik – Saarbrücken, DE)

Peer-to-peer protocols for maintaining distributed hash tables, such as Pastry or Chord, have become popular for certain Internet applications. While such protocols promise certain properties concerning correctness and performance, verification attempts using formal methods invariably discover border cases that violate some of those guarantees. For example, Zave discovered that no previously published version of Chord maintains the invariants claimed of the protocol. In his PhD thesis, Tianxiang Lu discovered similar correctness problems for Pastry and also developed a model, which he called LuPastry, for which he provided a partial proof of correct delivery assuming no node departures, mechanized in the TLA+ Proof System. We present the first complete proof of correct delivery for LuPastry, which we call LuPastry+.

3.2 CDCL as Saturation

Peter Baumgartner (NICTA – Canberra, AU)

Conflict driven clause learning (CDCL) is the main paradigm for building propositional logic SAT solvers. Saturation based theorem proving is the main paradigm for building first-order logic theorem provers. A natural research question is to investigate the relationships between these paradigms for, e.g., exploiting successful techniques from CDCL in first-order logic theorem proving. To this end, techniques like splitting, dependency-directed backtracking and lemma-learning techniques have been considered for integration in first-order logic resolution calculi and instance-based methods.

The paper revisits this topic from a different point of view. Instead of *integrating* its concepts, it shows how CDCL can be *simulated* by a saturation based resolution calculus. This is not trivial, as CDCL's splitting and backjumping operations are not compatible with saturation. One could, of course, add an explicit splitting rule to resolution, as mentioned above, but this would work in very restricted cases only. In contrast, our calculus approach allows for straightforward lifting to first-order logic. Moreover, in contrast to, e.g., model evolution calculi, it separates model representation from calculus. This supports the modular design of theorem provers, which, this way, e.g., may arbitrarily trade-off representational power versus efficiency, without compromising refutational completeness.

The main result for now is a refutational completeness result in presence of redundancy criteria and deletion rules. The latter are needed for a faithful simulation of CDCL.

3.3 Higher-Order Proofs and Models – Examples from Meta-Logical Reasoning and Metaphysics

Christoph Benzmueller (FU Berlin, DE)

Joint work of Benzmüller, Christoph; Woltzenlogel-Paleo, Bruno; Paulson, Lawrence; Brown, Chad; Claus, Maximilian; Sutcliffe, Geoff; Sultana, Nik; Blanchette, Jasmin
Main reference C. Benzmüller, B. Woltzenlogel Paleo, "Automating Gödel's Ontological Proof of God's Existence with Higher-order Automated Theorem Provers," in Proc. of the 21st Europ. Conf. on Artificial Intelligence (ECAI'14), Frontiers in Artificial Intelligence and Applications, Vol. 263, pp. 93–98, IOS Press, 2014.
URL http://dx.doi.org/10.3233/978-1-61499-419-0-93

Extraction and utilization of information (by hand) from higher-order logic proofs and countermodels has played an important role in my recent research. Two examples are presented, one from meta-logical reasoning and one from metaphysics.

In the first example countermodels from Nitpick were utilized in a schematic process to verify the independence of prominent modal logic axioms in Isabelle/HOL. In addition, minimality aspects of these models were proved. The independence results constituted the key steps in the verification of the well known modal logic cube.

In the second example, the higher-order prover LEO-II detected an inconsistency in Kurt Gödel's original variant of ontological argument for the existence of God. While LEO-II's (extensional higher-order RUE-resolution) proof object in fact contains the information needed for the reconstruction of a human-intuitive explanation, I failed for a long time to identify the relevant puzzle pieces. Only recently, I was able to extract (and verify) a surprisingly easily accessible abstract-level proof. It is as in many fields in mathematics: once a beautiful structure has been revealed, it can't be missed anymore. Unmated low-level formal proofs, in contrast, are lacking persuasive power.

References

1 Christoph Benzmüller and Bruno Woltzenlogel Paleo. *Automating Gödel's Ontological Proof of God's Existence with Higher- order Automated Theorem Provers*, In ECAI 2014, IOS Press, Frontiers in Artificial Intelligence and Applications, volume 263, pp. 93–98, 2014. http://dx.doi.org/10.3233/978-1-61499-419-0-93

2 Christoph Benzmüller and Bruno Woltzenlogel Paleo. *Higher-Order Modal Logics: Automation and Applications*, In Reasoning Web 2015, Springer, LNCS, number 9203, pp. 1–43, 2015. http://dx.doi.org/10.1007/978-3-319-21768-0_2

3 Christoph Benzmüller and Lawrence Paulson. *Quantified Multimodal Logics in Simple Type Theory*, In Logica Universalis, volume 7, number 1, pp. 7–20, 2013. http://dx.doi.org/10.1007/s11787-012-0052-y

4 Christoph Benzmüller. *Invited Talk: On a (Quite) Universal Theorem Proving Approach and Its Application in Metaphysics*, In TABLEAUX 2015, Springer, LNAI, volume 9323, pp. 209–216, 2015. http://dx.doi.org/10.1007/978-3-319-24312-2_15

5 Christoph Benzmüller, Maximilian Claus, and Nik Sultana. *Systematic Verification of the Modal Logic Cube in Isabelle/HOL*, In PxTP 2015, EPTCS, volume 186, pp. 27–41, 2015. http://dx.doi.org/10.4204/EPTCS.186.5

3.4 Semi-intelligible Isar Proofs from Machine-Generated Proofs

Jasmin Christian Blanchette (INRIA Lorraine – Nancy, FR)

Main reference J. C. Blanchette, S. Böhme, M. Fleury, S. J. Smolka, A. Steckermeier, "Semi-intelligible Isar Proofs from Machine-Generated Proofs," to appear in Journal of Automated Reasoning.

Sledgehammer is a component of the Isabelle/HOL proof assistant that integrates external automatic theorem provers (ATPs) to discharge interactive proof obligations. As a safeguard against bugs, the proofs found by the external provers are reconstructed in Isabelle. Reconstructing complex arguments involves translating them to Isabelle's Isar format, supplying suitable justifications for each step. Sledgehammer transforms the proofs by contradiction into direct proofs; it iteratively tests and compresses the output, resulting in simpler and faster proofs; and it supports a wide range of ATPs, including E, LEO- II, Satallax, SPASS, Vampire, veriT, Waldmeister, and Z3.

3.5 Tips and Tricks in LIA constraint solving

Martin Bromberger (MPI für Informatik – Saarbrücken, DE)

Joint work of Bromberger, Martin; Sturm, Thomas; Weidenbach, Christoph

We present tips and tricks for constraint solving in the theory of linear integer arithmetic. These tricks are sound, efficient, heuristic methods that find solutions for a large number of problems. While most complete methods search on the problem surface for a solution, these heuristics use balls and cubes to explore the interior of the problems. The heuristic methods are especially efficient for problems with a large number of integer solutions. Although it might seem that problems with a large number of integer solutions should be trivial for complete solvers, we will show the opposite by comparing state-of-the-art SMT solvers with our own solver that contains those heuristic methods.

3.6 Formally verified constraint solvers

Catherine Dubois (ENSIIE – Evry, FR)

Do you trust your solver ? In this talk, we focus on finite domains (FD) constraint solvers. We have developed a family of formally verified solvers through a generic and modular solver developed within the Coq proof assistant and proved sound and complete [CDD12]. Local consistency property, labeling strategy are parameters of this formal development. In the talk we present the main features and the current status of the development. Work in progress concerns the Coq formalization and verification of the well-known filtering algorithm [Reg94] for the alldiff constraint.

References

1 Carlier M., Dubois C., Gotlieb A. *A Certified Constraint Solver over Finite Domains*. FM 2012:116–131

2 Régin J.-C., *A Filtering Algorithm for Constraints of Difference in CSPs*. AAAI 1994:362–367

3.7 Overview of Models in Yices

Bruno Dutertre (SRI – Menlo Park, US)

We present a form of model-based theory combination recently implemented in Yices, algorithms for exists/forall solving and model generalization.

3.8 Automatic Proofs of Termination and Memory Safety for Programs with Pointer Arithmetic

Carsten Fuhs (Birkbeck, University of London, GB)

Joint work of Ströder, Thomas; Giesl, Jürgen; Brockschmidt, Marc; Frohn, Florian; Fuhs, Carsten; Hensel, Jera; Schneider-Kamp, Peter; Aschermann, Cornelius
Main reference T. Ströder, J. Giesl, M. Brockschmidt, F. Frohn, C. Fuhs, J. Hensel, P. Schneider-Kamp, "Proving Termination and Memory Safety for Programs with Pointer Arithmetic," in Proc. of the 7th Int'l Joint Conf. on Automated Reasoning (IJCAR'14), LNAI, Vol. 8562, pp. 208–223, Springer, 2014.
URL http://dx.doi.org/10.1007/978-3-319-08587-6_15

While automated verification of imperative programs has been studied intensively, proving termination of programs with explicit pointer arithmetic fully automatically was still an open problem. To close this gap, we introduce a novel abstract domain that can track allocated memory in detail. Automating our abstract domain with the help of SMT-based entailment proofs, we construct a symbolic execution graph that over-approximates all possible runs of the program and that can be used to prove memory safety. This graph is then transformed into an integer transition system, whose termination can be proved by standard techniques, e.g., based on models found by SMT solvers. We have implemented this approach in the automated termination prover AProVE and demonstrate its capability of analyzing C programs with pointer arithmetic that existing tools cannot handle.

3.9 Quantified Array Fragments: Decision Results and Applications

Silvio Ghilardi (University of Milan, IT)

Joint work of Alberti, Francesco; Ghilardi, Silvio; Sharygina, Natasha

The theory of arrays is one of the most relevant theories for software verification, this is the reason why current research in automated reasoning dedicated so much effort in establishing decision and complexity results for it. As soon as quantified formulae are concerned, however, satisfiability becomes intractable when free unary function symbols are added to mild fragments of arithmetic [6]. Nevertheless, since applications require the use of quantifiers, e.g. in order to express invariants of program loops, it becomes crucial to identify sufficiently expressive tractable quantified fragments of the theory.

In this talk we first compare and discuss some state-of-the-art literature on the subject [4], [5], [2], [3] and then we show how the results can be applied to model-checking problems in array-based systems. We finally report the status of the implementation in our tools MCMT and BOOSTER [1].

The original contributions of this talk come from joint work with F. Alberti and N. Sharygina.

References

1 F. Alberti, S. Ghilardi, and N. Sharygina. Booster : an acceleration-based verification framework for array programs. In *ATVA*, pages 18–23, 2014.
2 F. Alberti, S. Ghilardi, and N. Sharygina. Decision procedures for flat array properties. In *TACAS*, pages 15–30, 2014.
3 F. Alberti, S. Ghilardi, and N. Sharygina. A new acceleration-based combination framework for array properties. In *FroCoS*, 2015.
4 A.R. Bradley, Z. Manna, and H.B. Sipma. What's decidable about arrays? In *VMCAI*, pages 427–442, 2006.
5 P. Habermehl, R. Iosif, and T. Vojnar. A logic of singly indexed arrays. In *LPAR*, pages 558–573, 2008.
6 J.Y. Halpern. Presburger arithmetic with unary predicates is Π^1_1 complete. *J. Symbolic Logic*, 56(2):637–642, 1991.

3.10 Using Information from Deduction for Complexity Analysis

Juergen Giesl (RWTH Aachen University, DE)

License Creative Commons BY 3.0 Unported license
© Juergen Giesl
Joint work of Frohn, Florian; Giesl, Jürgen; Hensel, Jera; Aschermann, Cornelius; Ströder, Thomas
Main reference F. Frohn, J. Giesl, J. Hensel, C. Aschermann, T. Ströder, "Inferring Lower Bounds for Runtime Complexity," in Proc. of the 26th Int'l Conf. on Rewriting Techniques and Applications (RTA'15), LIPIcs, Vol. 36, pp. 334–349, Schloss Dagstuhl, 2015.
URL http://dx.doi.org/10.4230/LIPIcs.RTA.2015.334

Several techniques and tools have been developed to prove termination and to verify inductive properties of programs automatically. We report on our recent work to use information from such automatically generated proofs in order to analyze complexity of programs. More precisely, from automated termination proofs, one can infer upper bounds on a program's runtime and on the values of its variables. Moreover, from automated induction proofs, one can infer lower bounds on the runtime of a program.

3.11 SAT-based techniques for parameter synthesis and optimization

Alberto Griggio (Bruno Kessler Foundation – Trento, IT)

Joint work of Bittner, Benjamin; Bozzano, Marco; Cimatti, Alessandro; Gario, Marco; Griggio, Alberto; Mattarei, Cristian
Main reference M. Bozzano, A. Cimatti, A. Griggio, C. Mattarei, "Efficient Anytime Techniques for Model-Based Safety Analysis," in Proc. of the 27th Int'l Conf. on Computer Aided Verification (CAV'15), LNCS, Vol. 9206, pp. 603–621, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-21690-4_41

Many application domains can be described in terms of parameterized systems, where parameters are variables whose value is invariant over time, but is only partially constrained. A key challenge in this context is the estimation of the parameter valuations that guarantee the correct behavior of the system. Manual estimation of these values is time consuming and does not find optimal solutions for specific design problems. Therefore, a fundamental problem is to automatically synthesize the maximal region of parameter valuations for which the system satisfies some properties, or to find the best/most appropriate valuation with respect to a given cost function.

In this talk, we present a technique for parameter synthesis and optimization that exploits the efficiency of state-of-the art model checking algorithms based on SAT solvers. We will start from a general solution applicable in various settings, and then show how to improve the effectiveness of our procedure by exploiting domain knowledge. We demonstrate the usefulness of our technique with a set of case studies taken from the domains of diagnosability and safety analysis.

3.12 Exploit Generation for Information Flow Leaks in Object-Oriented Programs

Reiner Haehnle (TU Darmstadt, DE)

Joint work of Do, Quoc Huy; Bubel, Richard; Haehnle, Reiner
Main reference Q. H. Do, R. Bubel, R. Hähnle, "Exploit Generation for Information Flow Leaks in Object-Oriented Programs," in Proc. of the 30th IFIP TC 11 Int'l Conf. on ICT Systems Security and Privacy Protection (SEC'15), IFIP Advances in Information and Communication Technology, Vol. 455, pp. 401–415, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-18467-8_27

We present a method for automated generation of exploits for information flow leaks in object-oriented programs. Given a flow policy and a security level specification, our approach combines self-composition, symbolic execution, computation of an insecurity formula, and model generation to produce a test input that witnesses a security leak (if one exists). The method is one instance of a general framework for generating test data that witnesses a given relational program property, for example, faults, regressions, etc. A prototypic tool called KEG implementing our method for Java target programs is available. It generates security exploits in the form of executable JUnit tests.

3.13 Saturation Theorem Proving for Herbrand Models

Matthias Horbach (MPI für Informatik – Saarbrücken, DE)

In system verification, we are often interested in analyzing specific models, usually Herbrand models over a given domain. The use of efficient first-order methods like superposition in such a setting is unsound, because the introduction of Skolem constants for existential variables changes the Herbrand domain.

I will present superposition calculi that can explicitly represent existentially quantified variables in computations with respect to a given fixed domain. They give rise to new decision procedures for minimal model validity and I will demonstrate how to employ them for counter model generation in the analysis of Petri nets and LTL formulas, as well as in local reasoning.

3.14 SMT-based Reactive Synthesis

Swen Jacobs (Universität des Saarlandes, DE)

We consider reductions of the synthesis problem for distributed and parameterized reactive systems to problems in satisfiability modulo theories (SMT). Given a (possibly parametric) system architecture and an LTL specification, we use automata theory (and possibly cutoff results from parameterized verification) to reduce the synthesis problem for implementations that satisfy the specification to a set of first-order constraints. The problem is encoded such that a model of the constraints represents both the desired implementation and an additional annotation that witnesses correctness. Our experimental results with different approaches to solve such constraints suggest that this is a very hard problem for existing SMT solvers.

3.15 Interpolation Synthesis for Quadratic Polynomial Inequalities and Combination with EUF

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

Joint work of Gan, Ting; Dai, Liyun; Xia, Bican; Zhan, Naijun; Chen, Mingshuai

An algorithm for generating interpolants for formulas which are conjunctions of quadratic polynomial inequalities (both strict and nonstrict) is proposed. The algorithm is based on a key observation that quadratic polynomial inequalities can be linearized if they are concave. A generalization of Motzkin's transposition theorem is proved, which is used to generate an interpolant between two mutually contradictory conjunctions of polynomial inequalities, in a way similar to the linear inequalities case. This can be done efficiently using semi-definite programming but forsaking completeness. A combination algorithm is given for the combined theory of concave quadratic polynomial inequalities and the equality theory over uninterpreted functions symbols using a hierarchical framework for combining interpolation algorithms for quantifier-free theories. A preliminary implementation has been explored.

3.16 Obtaining Inductive Invariants with Formula Slicing

George Karpenkov (VERIMAG – Gières, FR)

Program analysis by abstract interpretation finds inductive invariants in a given abstract domain, over-approximating the reachable state-space. This over-approximation at every program location may lead to weak invariants, insufficient for proving a desired property. *Path focusing* and *large block encoding* alleviate this problem by requiring abstractions only at loop heads; at other control points, candidate invariants are expressed as first-order formulas within a decidable theory, precisely describing possible executions from the last loop head. This significantly improves the precision.

Our *formula slicing* approach goes further, by propagating first-order formulas *through* loop heads: formulas are weakened until they become inductive by replacing atomic predicates with "true".

We show that the problem of deciding the existence of a non-trivial weakening is Σ_2^p-complete. we propose the over-approximation approaches based on the existing algorithms from the literature.

The produced inductive weakenings can be conjoined to the invariant candidates expressed in the abstract domain, improving the analysis precision, as we demonstrate on a range of programs from the International Competition on Software Verification (SV-COMP).

3.17 Optimization modulo quantified linear rational arithmetic

Zachary Kincaid (University of Toronto, CA)

The optimization modulo theories (OMT) problem is to compute the supremum of the value of some given objective term over all models of a given (satisfiable) formula. Recently, techniques have been developed for optimization modulo the theories of quantifier-free linear rational (and integer) arithmetic. In principle, these techniques can be also be applied to formulas with quantifiers, since linear rational (integer) arithmetic admits quantifier elimination. However, quantifier elimination is computationally expensive, and it may be possible to avoid it. I will present an algorithm for optimization modulo quantified linear rational arithmetic that works directly on quantified linear rational arithmetic formulas.

3.18 EPR-based BMC and k-induction with Counterexample Guided Abstraction Refinement

Konstantin Korovin (University of Manchester, GB)

Joint work of Khasidashvili, Zurab; Korovin, Konstantin; Tsarkov, Dmitry
Main reference Z. Khasidashvili, K. Korovin, D. Tsarkov, "EPR-based k-induction with Counterexample Guided Abstraction Refinement," in Proc. of the 2015 Global Conf. on Artificial Intelligence (GCAI'15), EPiC Series, Vol. 36, pp. 137–150, EasyChair, 2015.
URL http://www.easychair.org/publications/paper/EPR-based_k-induction_with_Counterexample_Guided_Abstraction_Refinement

In recent years it was proposed to encode bounded model checking (BMC) into the effectively propositional fragment of first-order logic (EPR). The EPR fragment can provide for a succinct representation of the problem and facilitate reasoning at a higher level. In this talk we present a novel abstraction-refinement approach based on unsatisfiable cores and models (UCM) for BMC and k-induction in the EPR setting. We have implemented UCM refinements for EPR-based BMC and k-induction in a first-order automated theorem prover iProver [1]. We also extended iProver with the AIGER format and evaluated it over the HWMCC'14 competition benchmarks. The experimental results are encouraging. We show that a number of AIG problems can be verified until deeper bounds with the EPR-based model checking.

This talk is based on [2].

References

1 K. Korovin. *Inst-Gen – a modular approach to instantiation-based automated reasoning.* In Programming Logics, ser. LNCS, A. Voronkov and C. Weidenbach, Eds., vol. 7797. Springer, pp. 239–270, 2013.

2 Z. Khasidashvili, K. Korovin, D. Tsarkov. *EPR-based k-induction with Counterexample Guided Abstraction Refinement.* EPiC Series, EasyChair, 2015.

3.19 Hyperresolution modulo Horn Clauses – generating infinite models

Christopher Lynch (Clarkson University – Potsdam, US)

When Ordered Resolution terminates on a satisfiable set of clauses, it is not always possible to constructively find a model. On the other hand, in Hyperresolution a model can be easily computed, but Hyperresolution rarely halts.

We present a method to identify some Horn clauses that lead to nontermination, and remove them from the Hyperresolution process. Instead of resolving these clauses, unification will be performed modulo those clauses. In many cases, this will force Hyperresolution to halt, and the result will determine an infinite Herbrand model with nice properties, e.g, closed under intersection.

We first apply this result to Cryptographic Protocol Analysis, where Horn clauses used to represent Intruder Abilities cause nontermination. If Hyperresolution modulo Intruder Abilities halts then the infinite Herbrand model gives all the messages an intruder could learn.

We extend this result to a more general class of Horn clauses, which may be useful for program analysis where it is difficult or impossible to find a finite model.

The work on Cryptographic Protocol Analysis is joint with Erin Hanna, David Myers and Corey Richardson.

3.20 Confluence and Certification

Aart Middeldorp (Universität Innsbruck, AT)

Joint work of Nagele, Julian; Felgenhauer, Bertram; Middeldorp, Aart
Main reference J. Nagele, B. Felgenhauer, A. Middeldorp, "Improving Automatic Confluence Analysis of Rewrite Systems by Redundant Rules," in Proc. of the 16th Int'l Conf. on Rewriting Techniques and Applications (RTA'15), LIPIcs, Vol. 36, pp. 257–268, Schloss Dagstuhl, 2015.
URL http://dx.doi.org/10.4230/LIPIcs.RTA.2015.257

We discuss the importance of certification for confluence. A simple technique is presented that increases the power of modern (certified) confluence tools considerably.

3.21 Mining the Archive of Formal Proofs

Tobias Nipkow (TU München, DE)

Joint work of Blanchette, Jasmin C.; Haslbeck, Maximilian; Matichuk, Daniel; Nipkow, Tobias
Main reference J. C. Blanchette, M. Haslbeck, D. Matichuk, T. Nipkow, "Mining the Archive of Formal Proofs," in Proc. of the 2015 Int'l Conf. on Intelligent Computer Mathematics (CICM'15), LNCS, Vol. 9150, pp. 3–17, Springer, 2015.
URL https://doi.org/10.1007/978-3-319-20615-8_1

The Archive of Formal Proofs is a vast collection of computer-checked proofs developed using the proof assistant Isabelle. We perform an in-depth analysis of the archive, looking at various properties of the proof developments, including size, dependencies, and proof style. This gives some insights into the nature of formal proofs.

3.22 SMT-Based Methods for Difference Logic Invariant Generation

Albert Oliveras (UPC – Barcelona, ES)

Joint work of Candeago, Lorenzo; Oliveras, Albert; Rodríguez-Carbonell, Enric

We consider the problem of synthesizing difference logic invariants for a restricted class of imperative programs: the ones whose transitions can be described as conjunctions of difference logic inequalities.

Our methodology is based on the so-called constraint-based method: we consider a template for each location as a candidate invariant, to which initiation and consecution conditions are imposed. Unlike in the general case, where Farkas' lemma is used to convert these conditions into formulas over non-linear arithmetic, in our particular case we show how we can use more efficient SMT-based techniques using only difference logic arithmetic.

3.23 How to avoid proving the absence of integer overflows

Andrei Paskevich (University Paris-Sud, FR)

Joint work of Clochard, Martin; Filliâtre, Jean-Christophe; Paskevich, Andrei
Main reference M. Clochard, J.-C. Filliâtre, A. Paskevich, "How to avoid proving the absence of integer overflows," to appear in Proc. of the 7th Int'l Conf. on Verified Software: Theories, Tools, and Experiments (VSTTE'15), as volume 9593 of LNCS; pre-print available as hal-01162661, 2016.
URL https://hal.inria.fr/hal-01162661

When proving safety of programs, we must show, in particular, the absence of integer overflows. Unfortunately, there are lots of situations where performing such a proof is extremely difficult, because the appropriate restrictions on function arguments are invasive and may be hard to infer. Yet, in certain cases, we can relax the desired property and only require the absence of overflow during the first n steps of execution, n being large enough for all practical purposes. It turns out that this relaxed property can be easily ensured for large classes of algorithms, so that only a minimal amount of proof is needed, if at all. The idea is to restrict the set of allowed arithmetic operations on the integer values in question, imposing a "speed limit" on their growth. For example, if we repeatedly increment a 64-bit integer, starting from zero, then we will need at least 2 to the power of 64 steps to reach an overflow; on current hardware, this takes several hundred years. When we do not expect any single execution of our program to run that long, we have effectively proved its safety against overflows of all variables with controlled growth speed. In this talk, we give a formal explanation of this approach and show how it is implemented in the context of deductive verification.

3.24 Compositional Program Analysis using Max-SMT

Albert Rubio (UPC – Barcelona, ES)

Joint work of Brockschmidt, Marc; Larraz, Daniel; Oliveras, Albert; Rodríguez-Carbonell, Enric; Rubio, Albert
Main reference M. Brockschmidt, D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, A. Rubio, "Compositional Safety Verification with Max-SMT," to appear in Proc. of the 2015 Conf. on Formal Methods in Computer-Aided Design (FMCAD'15); pre-print available as arXiv:1507.03851v3 [cs.LO], 2015.
URL http://arxiv.org/abs/1507.03851v3

An automated compositional program verification technique for safety properties based on conditional inductive invariants is presented. For a given program part (e.g., a single loop) and a postcondition, we show how to, using a Max-SMT solver, an inductive invariant together with a precondition can be synthesized so that the precondition ensures the validity of the invariant and that the invariant implies the postcondition. From this, we build a bottom-up program verification framework that propagates preconditions of small program parts as postconditions for preceding program parts. The method recovers from failures to prove the validity of a precondition, using the obtained intermediate results to restrict the search space for further proof attempts.

Currently we are extending the framework to prove reachability properties by using conditional termination.

3.25 Exploiting Locality in Parametric Verification

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

Joint work of Damm, Werner; Horbach, Matthias; Sofronie-Stokkermans, Viorica
Main reference W. Damm, M. Horbach, V. Sofronie-Stokkermans, "Decidability of Verification of Safety Properties of Spatial Families of Linear Hybrid Automata," in Proc. of the 10th Int'l Symp. on Frontiers of Combining Systems (FroCoS'15), LNAI, Vol. 9322, pp. 186–202, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-24246-0_12

We show how hierarchical reasoning, quantifier elimination and model generation can be used to automatically provide guarantees that given parametric systems satisfy certain safety or invariance conditions. Such guarantees can be for instance expressed as constraints on parameters. Alternatively, hierarchical reasoning combined with techniques for model generation allows us to construct counterexamples which show how unsafe states can be reached.

In this talk we focus on the case of systems composed of an unbounded number of similar components (modeled as linear hybrid automata), whose dynamic behavior is determined by their relation to neighboring systems. We present a class of such systems and a class of safety properties whose verification can be reduced to the verification of (small) families of neighboring systems of bounded size, and identify situations in which such verification problems are decidable, resp. fixed parameter tractable. We illustrate the approach with an example from coordinated vehicle guidance.

3.26 Verified AC-Equivalence Checking in Isabelle/HOL

Christian Sternagel (Universität Innsbruck, AT)

Joint work of Bertram, Felgenhauer; Sternagel, Christian

We present an algebraic correctness proof of an executable AC-equivalence check that we formalized in Isabelle/HOL. This work constitutes the basis for extending our Isabelle/HOL Formalization of Rewriting (IsaFoR) with results on rewriting modulo associativity and commutativity.

3.27 Thousands of Models for Theorem Provers – The TMTP Model Library

Geoff Sutcliffe (University of Miami, US)

The TPTP World is a well established infrastructure that supports research, development, and deployment of Automated Theorem Proving (ATP) systems for classical logics. The TPTP World includes the TPTP problem library, the TSTP solution library, standards for writing ATP problems and reporting ATP solutions, tools and services for processing ATP problems and solutions, and it supports the CADE ATP System Competition (CASC).

This work describes a new component of the TPTP World – the Thousands of Models for Theorem Provers (TMTP) Model Library. This will be a corpus of models for identified sets of axioms in the TPTP, along with tools for interpreting formulae wrt models, tools for translating from from model form to another, interfaces for visualizing models, etc. The TMTP will support the development of semantically guided theorem proving ATP systems, provide examples for developers of model finding ATP systems, and provide insights into the semantic structure of axiom sets.

3.28 Conflict-based Quantifier Instantiation for SMT

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

Joint work of Reynolds, Andrew; Tinelli, Cesare; de Moura, Leonardo
Main reference A. Reynolds, C. Tinelli, L. de Moura, "Finding conflicting instances of quantified formulas in SMT," in Proc. of the 2014 Conf. on Formal Methods in Computer-Aided Design (FMCAD'14), pp. 195–202, IEEE, 2014.
URL http://dx.doi.org/10.1109/FMCAD.2014.6987613

Satisfiability Modulo Theories (SMT) solvers have been used successfully in a variety of applications including verification, automated theorem proving, and synthesis. While such solvers are highly adept at handling ground constraints in several decidable background theories, they primarily rely on heuristic quantifier instantiation methods such as E-matching to process quantified formulas. The success of these methods is often hindered by an overproduction of instances, which makes ground level reasoning difficult. This talk introduces a new technique that alleviates this shortcoming by first discovering instances of the quantified formulas that are in conflict with the current state of the solver. The solver only resorts to traditional heuristic methods when such instances cannot be found, thus decreasing its dependence upon E-matching. Extensive experimental results show that this technique significantly reduces the number of instantiations required by an SMT solver to answer "unsatisfiable" for several benchmark libraries, and consequently leads to improvements over state-of-the-art implementations.

3.29 Learn Fresh: Model-Guided Inferences

Christoph Weidenbach (MPI für Informatik – Saarbrücken, DE)

Joint work of Alagi, Gabor; Teucke, Andreas; Weidenbach, Christoph
Main reference C. Weidenbach, "Automated Reasoning Building Blocks," in R. Meyer, A. Platzer, H. Wehrheim (eds.), "Correct System Design – Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday", LNCS, Vol. 9360, pp. 172–188, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-23506-6_12

I investigate the relationship between candidate models, inferences and redundancy. It turns out that clauses learned by the CDCL calculus correspond to the result of superposition inferences and are not redundant. The result can be lifted to the Bernays Schoenfinkel class but it is open how it can be lifted to first-order logic, in general. For-first order logic abstraction mechanisms are needed that result in effective model representations enabling decision procedures for clause validity.

References

1 Gabor Alagi and Christoph Weidenbach. NRCL – a model building approach to the Bernays-Schönfinkel fragment. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems, 10th International Symposium, FroCoS 2015, Wroslav, Poland, 2015. Proceedings*, volume 9322 of *LNCS*, pages 69–84. Springer, 2015.

2 Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Lemma learning in the model evolution calculus. In *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 572–586. Springer, 2006.

3 Harald Ganzinger and Konstantin Korovin. New directions in instatiation–based theorem proving. In Samson Abramsky, editor, *18th Annual IEEE Symposium on Logic in Computer Science, LICS'03*, LICS'03, pages 55–64. IEEE Computer Society, 2003.

4 Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(t). *Journal of the ACM*, 53:937–977, November 2006.

5 Ruzica Piskac, Leonardo Mendonça de Moura, and Nikolaj Bjørner. Deciding effectively propositional logic using DPLL and substitution sets. *Journal of Automated Reasoning*, 44(4):401–424, 2010.

6 Andreas Teucke and Christoph Weidenbach. First-order logic theorem proving and model building via approximation and instantiation. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems, 10th International Symposium, FroCoS 2015, Wroslav, Poland, 2015. Proceedings*, volume 9322 of *LNCS*, pages 85–100. Springer, 2015.

7 Christoph Weidenbach. Automated reasoning building blocks. In Roland Meyer, André Platzer, and Heike Wehrheim, editors, *Correct System Design – Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday, Oldenburg, Germany, September 8-9, 2015. Proceedings*, volume 9360 of *Lecture Notes in Computer Science*, pages 172–188. Springer, 2015.

3.30 Partial Models for More Proofs

Sarah Winkler (Microsoft Research UK – Cambridge, GB)

Joint work of Sato, Haruhiko; Winkler, Sarah
Main reference H. Sato. S. Winkler, "Encoding Dependency Pair Techniques and Control Strategies for Maximal Completion," in Proc. of the 25th Int'l Conf. on Automated Deduction (CADE'15), LNAI, Vol. 9195, pp. 152–162, Springer, 2015.
URL http://dx.doi.org/10.1007/978-3-319-21401-6_10

Maximal completion constitutes a powerful and fast Knuth-Bendix completion procedure based on MaxSAT/MaxSMT solving. Recent advancements let Maxcomp improve over other automatic completion tools, and produce novel complete systems [1]: (1) Termination techniques using the dependency pair framework are encoded as satisfiability problems, including dependency graph and reduction pair processors. (2) Instead of relying on pure maximal completion, different SAT-encoded control strategies are exploited.

Maximal completion can also produce complete systems for subtheories (partial models): This is done by encoding control strategies which, for instance, give preference to rewrite systems where the number of non-joinable critical pairs is minimal.

Exploiting this feature, we use maximal completion to guide equational proof search. More precisely, we investigate how the addition of a partial model R (and a reduction order to prove it terminating) to unit equality problems from TPTP influences the behavior of provers on these problems. When restricting to reduction orders which are total on ground terms, there always exists a ground complete system extending R, hence completeness is not compromised. Experiments with the theorem prover SPASS show that supplying complete systems for subtheories is indeed beneficial, though adding the respective reduction order has more effect than the additional rewrite rules.

References

1 H. Sato and S. Winkler. Encoding Dependency Pair Techniques and Control Strategies for Maximal Completion. In *CADE*, volume 9195 of *LNCS*, pages 152–162, 2 2015.

Participants

- Noran Azmy
MPI für Informatik – Saarbrücken, DE
- Franz Baader
TU Dresden, DE
- Peter Baumgartner
NICTA – Canberra, AU
- Christoph Benzmüller
FU Berlin, DE
- Nikolaj S. Bjørner
Microsoft Corporation – Redmond, US
- Jasmin Christian Blanchette
INRIA Lorraine – Nancy, FR
- Martin Bromberger
MPI für Informatik – Saarbrücken, DE
- Catherine Dubois
ENSIIE – Evry, FR
- Bruno Dutertre
SRI – Menlo Park, US
- Carsten Fuhs
Birkbeck, Univ. of London, GB
- Silvio Ghilardi
University of Milan, IT
- Jürgen Giesl
RWTH Aachen University, DE
- Alberto Griggio
Bruno Kessler Foundation – Trento, IT
- Arie Gurfinkel
Carnegie Mellon University – Pittsburgh, US
- Liana Hadarean
University of Oxford, GB
- Reiner Hähnle
TU Darmstadt, DE
- Matthias Horbach
MPI für Informatik – Saarbrücken, DE
- Swen Jacobs
Universität des Saarlandes, DE
- Dejan Jovanovic
SRI – Menlo Park, US
- Deepak Kapur
University of New Mexico – Albuquerque, US
- George Karpenkov
VERIMAG – Gières, FR
- Zachary Kincaid
University of Toronto, CA
- Konstantin Korovin
University of Manchester, GB
- Christopher Lynch
Clarkson Univ. – Potsdam, US
- Aart Middeldorp
Universität Innsbruck, AT
- Tobias Nipkow
TU München, DE
- Albert Oliveras
UPC – Barcelona, ES
- Andrei Paskevich
University Paris-Sud, FR
- Alexander Rabinovich
Tel Aviv University, IL
- Giles Reger
University of Manchester, GB
- Albert Rubio
UPC – Barcelona, ES
- Andrey Rybalchenko
Microsoft Research UK – Cambridge, GB
- Stephan Schulz
Duale Hochschule Baden-Württemberg – Stuttgart, DE
- Viorica Sofronie-Stokkermans
Universität Koblenz-Landau, DE
- Christian Sternagel
Universität Innsbruck, AT
- Geoff Sutcliffe
University of Miami, US
- Cesare Tinelli
Univ. of Iowa – Iowa City, US
- Andrei Voronkov
University of Manchester, GB
- Christoph Weidenbach
MPI für Informatik – Saarbrücken, DE
- Sarah Winkler
Microsoft Research UK – Cambridge, GB
- Burkhart Wolff
University Paris-Sud, FR
- Jian Zhang
Chinese Academy of Sciences – Beijing, CN