Report from Dagstuhl Seminar 19371

Deduction Beyond Satisfiability

Edited by

Carsten Fuhs[1], Philipp Rümmer[2], Renate Schmidt[3], and Cesare Tinelli[4]

1 **Birkbeck, University of London, GB,** `carsten@dcs.bbk.ac.uk`
2 **Uppsala University, SE,** `philipp.ruemmer@it.uu.se`
3 **University of Manchester, GB,** `renate.schmidt@manchester.ac.uk`
4 **University of Iowa – Iowa City, US,** `cesare-tinelli@uiowa.edu`

Abstract

Research in automated deduction is traditionally focused on the problem of determining the satisfiability of formulas or, more generally, on solving logical problems with yes/no answers. Thanks to recent advances that have dramatically increased the power of automated deduction tools, there is now a growing interest in extending deduction techniques to attack logical problems with more complex answers. These include both problems with a long history, such as quantifier elimination, which are now being revisited in light of the new methods, as well as newer problems such as minimal unsatisfiable cores computation, model counting for propositional or first-order formulas, Boolean or SMT constraint optimization, generation of interpolants, abductive reasoning, and syntax-guided synthesis. Such problems arise in a variety of applications including the analysis of probabilistic systems (where properties like safety or liveness can be established only probabilistically), network verification (with relies on model counting), the computation of tight complexity bounds for programs, program synthesis, model checking (where interpolation or abductive reasoning can be used to achieve scale), and ontology-based information processing. The seminar brought together researchers and practitioners from many of the often disjoint sub-communities interested in the problems above. The unifying theme of the seminar was how to harness and extend the power of automated deduction methods to solve problems with more complex answers than binary ones.

Seminar September 8–13, 2019 – http://www.dagstuhl.de/19371

2012 ACM Subject Classification Theory of computation → Automated reasoning, Computing methodologies → Knowledge representation and reasoning, Theory of computation → Logic and verification

Keywords and phrases abduction, automated deduction, interpolation, quantifier elimination, synthesis

Digital Object Identifier 10.4230/DagRep.9.9.23

Edited in cooperation with Warren Del-Pinto

1 Executive Summary

Carsten Fuhs
Philipp Rümmer
Renate Schmidt
Cesare Tinelli

This report contains the program and outcomes of Dagstuhl Seminar 19371 on "Deduction Beyond Satisfiability" held at Schloss Dagstuhl, Leibniz Center for Informatics, during September 10–15, 2017. It was the thirteenth in a series of Dagstuhl Deduction seminars held biennially since 1993.

Research in automated deduction has traditionally focused on solving decision problems, which are problems with a binary answer. Prominent examples include proving the unsatisfiability of a formula, proving that a formula follows logically from others, checking the consistency of an ontology, proving safety or termination properties of programs, and so on. However, automated deduction methods and tools are increasingly being used to address problems with more complex answers, for instance to generate programs from formal specifications, compute complexity bounds, or find optimal solutions to constraint satisfaction problems.

In some cases, the required extended functionality (e.g., to identify unsatisfiable cores) can be provided relatively easily from current deduction procedures. In other cases (e.g., for Craig interpolation, or to find optimal solutions of constraints), elaborate extensions of these procedures are needed. Sometimes, altogether different methods have to be devised (e.g., to count the number of models of a formula, compute the set of all consequences of an ontology, identify missing information in a knowledge base, transform and mine proofs, or analyze probabilistic systems). In all cases, the step from yes/no answers to such extended queries and complex output drastically widens the application domain of deductive machinery. This is proving to be a key enabler in a variety of areas such as formal methods (for software/hardware development) and knowledge representation and reasoning.

While promising progress has been made, many challenges remain. Extending automated deduction methods and tools to support these new functionalities is often intrinsically difficult, and challenging both in theory and implementation. The scarcity of interactions between the involved sub-communities represents another substantial impediment to further advances, which is unfortunate because these sub-communities often face similar problems and so could greatly benefit from the cross-fertilization of ideas and approaches. An additional challenge is the lack of common standards for interfacing tools supporting the extended queries. Developing common formalisms, possibly as extensions of current standard languages, could be as transformative to the field as the introduction of standards such as TPTP and SMT-LIB has been in the past.

This Dagstuhl seminar brought together researchers working on deduction methods and tools that go beyond satisfiability and other traditional decision problems; specialists that work on advanced techniques in deduction and automated reasoning such as model counting, quantifier elimination, interpolation, abduction, or optimization; and consumers of deduction technology who need answers to more complex queries than just yes/no questions.

The unifying theme of the seminar was how to harness and extend the power of automated deduction methods to solve a variety of non-decision problems with useful applications. Research questions addressed at the seminar were the following:

- What kind of information should be passed to a "beyond satisfiability" deduction tool, and what information should be returned to the user? The goal should be to enhance the understanding of related concepts in different subfields and applications, and to converge towards common formalisms.
- How can current ideas, results and systems in one sub-community of researchers and practitioners benefit the needs of other communities?
- What are outstanding challenges in using and building deduction tools to attack logical problems with complex answers?

2 Table of Contents

3 Overview of Talks

3.1 Safe Decomposition of Startup Requirements: Verification and Synthesis

Alberto Griggio (Bruno Kessler Foundation – Trento, IT)

Joint work of Alberto Griggio, Alessandro Cimatti, Luca Geatti, Stefano Tonett

The initialization of complex cyber-physical systems often requires the interaction of various components that must start up with strict timing requirements on the provision of signals (power, refrigeration, light, etc.). In order to safely allow an independent development of components, it is necessary to ensure a safe decomposition, i.e. the specification of local timing requirements that prevent later integration errors due to the dependencies.

We propose a high-level formalism to model local timing requirements and dependencies. We consider the problem of checking the consistency (existence of an execution satisfying the requirements) and compatibility (absence of an execution that reach an integration error) of the local requirements, and the problem of synthesizing a region of timing constraints that represent all possible correct refinements of the original specification. We show how the problems can be naturally translated into a reachability and synthesis problem for timed automata with shared variables. Exploiting the linear structure of the requirements, we propose an encoding of the problem into SMT. We evaluate the SMT-based approach using Mathsat and show how it scales better compared to the automata-based approach using Uppaal and nuXmv.

3.2 Proof Checking in Zipperposition

Alexander Bentkamp (Free University Amsterdam, NL)

Joint work of Alexander Bentkamp, Simon Cruanes

Beyond a simple yes/no answer, automated theorem provers can typically generate proof output. To detect soundness bugs and to ensure better guarantees on the correctness of these proofs, we developed a proof checker for Zipperposition, a prover for polymorphic higher-order logic. The checker reduces the proof into a series of ground problems, which can be decidably and efficiently checked by a simple SMT prover.

3.3 Guiding High-Performance SAT Solvers with Unsat-Core Predictions

Nikolaj S. Bjørner (Microsoft Research – Redmond, US)

Joint work of Nikolaj S. Bjørner, Daniel Selsam
Main reference Daniel Selsam, Nikolaj Bjørner: "Guiding High-Performance SAT Solvers with Unsat-Core Predictions", in Proc. of the Theory and Applications of Satisfiability Testing – SAT 2019 – 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings, Lecture Notes in Computer Science, Vol. 11628, pp. 336–353, Springer, 2019.
URL http://dx.doi.org/10.1007/978-3-030-24258-9_24

The NeuroSAT neural network architecture was introduced by Selsam et al for predicting properties of propositional formulae. When trained to predict the satisfiability of toy problems, it was shown to find solutions and unsatisfiable cores on its own. However, the authors saw "no obvious path" to using the architecture to improve the state-of-the-art. In this work, we train a simplified NeuroSAT architecture to directly predict the unsatisfiable cores of real problems. We modify several state-of-the-art SAT solvers to periodically replace their variable activity scores with NeuroSAT's prediction of how likely the variables are to appear in an unsatisfiable core. The modifications led to speedups of 10 percent on unseen and diverse problems and 20 percent on problems form a scheduling domain. Our results demonstrate that NeuroSAT can provide effective guidance to high-performance SAT solvers on real problems. The talk introduces the techniques for representing CNF problems as graphical neural networks, our choice of training approach and some caveats in whether the measured improvements can be obtained in alternative ways. NeuroCore was developed in collaboration with Daniel Selsam and develops on the NeuroSAT architecture also developed by Selsam and collaborators.

References

1 Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, David L. Dill: Learning a SAT Solver from Single-Bit Supervision. ICLR 2019.
2 Daniel Selsam, Nikolaj Bjørner: Guiding High-Performance SAT Solvers with Unsat-Core Predictions. SAT 2019: 336-353

3.4 Proof reconstruction in conflict-driven satisfiability

Maria Paola Bonacina (Università degli Studi di Verona, IT)

Joint work of Maria Paola Bonacina, Stéphane Graham-Lengrand, Natarajan Shankar
Main reference Maria Paola Bonacina, Stéphane Graham-Lengrand, Natarajan Shankar: "Proofs in conflict-driven theory combination", in Proc. of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8-9, 2018, pp. 186–200, ACM, 2018.
URL http://dx.doi.org/10.1145/3167096

Proofs of unsatisfiability, or, equivalently, validity, are an important output of automated reasoning methods, and their transformation, exchange, and standardization is a key factor for the interoperability of different automated reasoning systems. In theorem proving proof reconstruction is the task of extracting a proof from the final state of a derivation after generating the empty clause, and for several theorem-proving methods it is a standard, though nontrivial, task. In SAT solving the conflict-driven clause learning procedure (CDCL) generates proofs by resolution: while this is true in principle, in practice, SAT-solver proofs are so huge that their definition, generation, and manipulation is an active research topic. In SMT solving, which represents a middle ground between first-order theorem proving and SAT solving, proof generation is also crucial, and while it receives increasing attention, the field does not seem to have a standard output format for proofs. This talk aims at contributing to this discussion by presenting approaches to proof reconstruction in CDSAT (Conflict-Driven SATisfiability), the paradigm for satisfiability modulo theories and assignments (SMT/SMA) developed by the author with Stéphane Graham-Lengrand and Natarajan Shankar.

3.5 Tractable QBF and model counting via Knowledge Compilation

Florent Capelli (INRIA Lille, FR)

Joint work of Florent Capelli, Simone Bova, Stefan Mengel, Friedrich Slivovsky
Main reference Florent Capelli, Stefan Mengel: "Tractable QBF by Knowledge Compilation", in Proc. of the 36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany, LIPIcs, Vol. 126, pp. 18:1–18:16, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2019.
URL http://dx.doi.org/10.4230/LIPIcs.STACS.2019.18

We show how knowledge compilation can be used as a tool for solving QBF and more. More precisely, we show that one can apply quantification on certain data structures used in knowledge compilation which in combination with the fact that restricted classes of CNF-formulas can be compiled into these data structures can be used to show fixed-parameter tractable results for QBF. In particular, we rediscover a result by Hubie Chen (ECAI 04) on FPT-tractability of QBF on bounded treewidth CNF and generalise it to aggregation problems such as counting or enumerating the models of the input quantified CNF.

This talk will review joint work with Simone Bova, Stefan Mengel and Friedrich Slivovsky.

3.6 Forgetting-Based Abductive Reasoning and Inductive Learning in Ontologies

Warren Del-Pinto (University of Manchester, GB)

Joint work of Warren Del-Pinto, Renate A. Schmidt
Main reference Warren Del-Pinto, Renate A. Schmidt: "ABox Abduction via Forgetting in ALC", in Proc. of the The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 – February 1, 2019, pp. 2768–2775, AAAI Press, 2019.
URL http://dx.doi.org/10.1609/aaai.v33i01.33012768

We present an approach to performing abductive reasoning in large description logic (DL) ontologies. The approach is based on the use of forgetting. Characteristics of the hypotheses obtained and important considerations such as redundancy elimination will be discussed alongside experimental results.

We also discuss the potential interaction between this form of abductive reasoning and inductive learning in DL ontologies. In particular: how do the characteristics of these hypotheses lend themselves to several learning problems in this setting?

3.7 Proofs in SMT

Pascal Fontaine (LORIA & INRIA – Nancy, FR)

Joint work of Pascal Fontaine, Haniel Barbosa, Jasmin Blanchette, Hans-Jörg Schurr
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

Satisfiability Modulo Theories (SMT) solvers are successfully used for various applications, notably in verification platforms and as back-ends for interactive theorem provers. In both cases, proofs are valuable. In the first, they help to improve confidence, and can also be used for certification in an industrial context. In the second, proofs can be used to reconstruct theorems within the kernel of proof assistants.

Ideally, proofs for SMT should be full and detailed, and at the same time they should not be too large. The overhead of outputting proofs should preferably be small. The talk briefly reviews the various aspects for outputting proofs in SMT, notably in the preprocessing phase, and discusses some issues that still need to be tackled.

3.8 Loop Acceleration for Under-Approximating Program Analysis

Florian Frohn (MPI für Informatik – Saarbrücken, DE)

Joint work of Florian Frohn, Jürgen Giesl, Marc Brockschmidt, Matthias Naaf

In the last years, under-approximating loop acceleration techniques have successfully been used to analyze programs operating on integers. Essentially, such techniques replace single-path loops with non-deterministic straight-line code that under-approximates the effect of the loops. The key to the success of this approach is its ability to construct symbolic under-approximations that cover program traces of arbitrary length. Applications include proving reachability, deducing lower bounds on the worst-case runtime complexity, and proving non-termination.

In this talk, two novel acceleration techniques will be presented. Furthermore, we will discuss several open problems related to loop acceleration. Finally, we will discuss an empirical evaluation of the presented approach.

3.9 From Derivational Complexity to Runtime Complexity of Term Rewriting

Carsten Fuhs (Birkbeck, University of London, GB)

License Ⓒ Creative Commons BY 3.0 Unported license
© Carsten Fuhs
Main reference Carsten Fuhs: "Transforming Derivational Complexity of Term Rewriting to Runtime Complexity", in Proc. of the Frontiers of Combining Systems – 12th International Symposium, FroCoS 2019, London, UK, September 4-6, 2019, Proceedings, Lecture Notes in Computer Science, Vol. 11715, pp. 348–364, Springer, 2019.
URL http://dx.doi.org/10.1007/978-3-030-29007-8_20

Derivational complexity of term rewriting considers the length of the longest rewrite sequence for arbitrary start terms, whereas runtime complexity restricts start terms to basic terms. Recently, there has been notable progress in automatic inference of upper and lower bounds for runtime complexity. We propose a novel transformation that allows an off-the-shelf tool for inference of upper or lower bounds for runtime complexity to be used to determine upper or lower bounds for derivational complexity as well. Our approach is applicable to derivational complexity problems for innermost rewriting and for full rewriting. We have implemented the transformation in the tool AProVE and conducted an extensive experimental evaluation. Our results indicate that bounds for derivational complexity can now be inferred for rewrite systems that have been out of reach for automated analysis thus far. At the Dagstuhl seminar, we also discussed extensions and possible applications of our approach.

3.10 Decision Procedures Beyond Satisfiability

Jürgen Giesl (RWTH Aachen, DE)

License Ⓒ Creative Commons BY 3.0 Unported license
© Jürgen Giesl
Joint work of Jürgen Giesl, Florian Frohn, Peter Giesl, Marcel Hark
URL https://doi.org/10.1016/j.ipl.2018.06.012, https://doi.org/10.1007/978-3-030-25543-5_24, https://doi.org/10.1007/978-3-030-29436-6_16
Main reference Florian Frohn, Jürgen Giesl: "Constant runtime complexity of term rewriting is semi-decidable", Inf. Process. Lett., Vol. 139, pp. 18–23, 2018.
URL http://dx.doi.org/10.1016/j.ipl.2018.06.012
Main reference Florian Frohn, Jürgen Giesl: "Termination of Triangular Integer Loops is Decidable", in Proc. of the Computer Aided Verification – 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 11562, pp. 426–444, Springer, 2019.
URL http://dx.doi.org/10.1007/978-3-030-25543-5_24
Main reference Jürgen Giesl, Peter Giesl, Marcel Hark: "Computing Expected Runtimes for Constant Probability Programs", in Proc. of the Automated Deduction – CADE 27 – 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings, Lecture Notes in Computer Science, Vol. 11716, pp. 269–286, Springer, 2019.
URL http://dx.doi.org/10.1007/978-3-030-29436-6_16

We give an overview on our recent work on finding (sub-)classes of programs with decidable termination or complexity properties. All of our decision procedures give results that go beyond binary "yes/no" answers.

Our first result is that it is semi-decidable whether the runtime complexity of a term rewrite system is constant. In case of constant runtime complexity, our semi-decision procedure also computes the exact worst-case runtime.

In our second result, we consider affine integer programs and show that termination is decidable for programs that consist of a single loop where the update matrix is triangular. Moreover, our procedure can also compute witnesses for eventual non-termination.

Finally, we regard a class of probabilistic programs with constant probabilities (so-called CP programs) and present a procedure to decide (positive) almost sure termination and to compute asymptotically tight bounds on their expected runtime. Based on this, we developed an algorithm to infer the exact expected runtime of any CP program.

3.11 Spacer on Jupyter

Arie Gurfinkel (University of Waterloo, CA)

Joint work of Arie Gurfinkel, Nikolaj Bjørner

Constrained Horn Clauses (CHC) is a fragment of First Order Logic modulo constraints that captures many program verification problems as constraint solving. Safety verification of sequential programs, modular verification of concurrent programs, parametric verification, and modular verification of synchronous transition systems are all naturally captured as a satisfiability problem for CHC modulo theories of arithmetic and arrays.

Of course, satisfiability of CHC is undecidable. Thus, solving them is a mix of science, art, and a dash of magic. In this talk, we have presented a tutorial on using a CHC solver Spacer that is build into SMT solver Z3. The tutorial is illustrated by a Jupyter notebook that shows how different verification problems are modeled by CHC and solved by Spacer.

The corresponding notebook is available at: https://spacerexamples-ariegurfinkel.notebooks.azure.com/

3.12 Abstract Execution

Reiner Hähnle (TU Darmstadt, DE)

Joint work of Reiner Hähnle, Steinhöfel, Dominic
Main reference Dominic Steinhöfel, Reiner Hähnle: "Abstract Execution", in Proc. of the Formal Methods – The Next 30 Years – Third World Congress, FM 2019, Porto, Portugal, October 7-11, 2019, Proceedings, Lecture Notes in Computer Science, Vol. 11800, pp. 319–336, Springer, 2019.
URL http://dx.doi.org/10.1007/978-3-030-30942-8_20

We propose a new static software analysis principle called Abstract Execution, generalizing Symbolic Execution: While the latter analyzes all possible execution paths of a specific program, abstract execution analyzes a partially unspecified program by permitting abstract symbols representing unknown contexts. For each abstract symbol, we faithfully represent each possible concrete execution resulting from its substitution with concrete code. There is a wide range of applications of abstract execution, especially for verifying relational properties of schematic programs. We implemented abstract execution in a deductive verification framework and proved correctness of eight well-known statement-level refactoring rules, including two with loops. For each refactoring we characterize the preconditions that make it semantics-preserving. Most preconditions are not mentioned in the literature.

3.13 Reasoning about Expected Runtimes of Probabilistic Programs (and Quantitative Separation Logic)

Benjamin Kaminski (RWTH Aachen, DE)

Joint work of Benjamin Kaminski, Kevin Batz, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll
Main reference Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, Federico Olmedo: "Weakest Precondition Reasoning for Expected Runtimes of Randomized Algorithms", J. ACM, Vol. 65(5), pp. 30:1–30:68, 2018.
URL http://dx.doi.org/10.1145/3208102

We present a weakest-precondition-style calculus a la Dijkstra tailored to reasoning about the expected runtime of probabilistic programs. We put a particular focus on inductive invariant-style reasoning for loops. These invariants are *quantitative* and thus lie inherently beyond the scope of Boolean predicates. Major problems in this context are e.g. to (a) synthesize inductive invariants completely from scratch or (b) shape non-inductive candidates into an inductive invariants.

We also present a quantitative separation logic for reasoning about quantitative aspects of randomized programs that have access to dynamic memory.

3.14 Algorithmic Proof Analysis by CERES

Alexander Leitsch (TU Wien, AT)

Proofs are more than just validations of theorems; they can contain interesting mathematical information like bounds or algorithms. However this information is frequently hidden and proof transformations are required to make it explicit. One such transformation on proofs in sequent calculus is cut-elimination (i.e. the removal of lemmas in formal proofs to obtain proofs made essentially of the syntactic material of the theorem). In his famous paper "Untersuchungen über das logische Schließen" Gentzen showed that for cut-free proofs of prenex end-sequents Herbrand's theorem can be realized via the midsequent theorem. In fact Gentzen defined a transformation which, given a cut-free proof p of a prenex sequent S, constructs a cut-free proof p' of a sequent S' (a so-called Herbrand sequent) which is propositionally valid and is obtained by instantiating the quantifiers in S. These instantiations may contain interesting and compact information on the validity of S. Generally, the construction of Herbrand sequents requires cut-elimination in a given proof p (or at least the elimination of quantified cuts). The method CERES (cut-elimination by resolution) [3] provides an algorithmic tool to compute a Herbrand sequent S' from a proof p (with cuts) of S even without constructing a cut-free version of p [5]. A prominent and crucial principle in mathematical proofs is mathematical induction. However, in proofs with mathematical induction Herbrand's theorem typically fails. To overcome this problem we replace induction by recursive definitions and proofs by proof schemata [1], [2], [4]. An extension of CERES to proof schemata (CERESs) allows to compute inductive definitions of Herbrand expansions (so-called Herbrand systems) representing infinite sequences of Herbrand sequents, resulting in a form of Herbrand's theorem for inductive proofs.

References

1 A. Leitsch, N. Peltier, D. Weller. CERES for first-order schemata. LogCom: Journal of Logic and Computation, vol. 27/7 1897-1954 (2017).
2 C. Dunchev, A. Leitsch, M. Rukhaia, D. WellerCut-Elimination and Proof Schemata. In Logic, Language and Computation, 117-136, Lecture Notes in Computer Science 8984 (2015).
3 M. Baaz and A. Leitsch. Methods of Cut-Elimination, Trends in Logic vol. 34, Springer, (2011) .
4 D. Cerna and A. Leitsch. Schematic Cut Elimination and the Ordered Pigeonhole Principle. IJCAR 2016: 241-256 (2016).
5 A. Leitsch and A. Lolic. Extraction of Epansion Trees. J.Autom.Reasoning 63(1): 95-126 (2019)

3.15 Efficient SAT-Based Reasoning Beyond NP

Joao Marques Silva (Federal University – Toulouse, FR)

The performance improvements made to SAT solvers over the last two decades have reshaped the organization of reasoners for different problem domains, within and beyond NP. This talk provides an overview of recent work on tackling decision and related problems beyond NP. A few successful examples include Maximum Satisfiability (MaxSAT), Propositional Abduction, Quantified Boolean Formulas (QBF), among others.

3.16 Correct-by-Decision Solving and Applications

Alexander Nadel (Intel Israel – Haifa, IL)

Main reference Alexander Nadel: “A Correct-by-Decision Solution for Simultaneous Place and Route”, in Proc. of the Computer Aided Verification – 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 10427, pp. 436–452, Springer, 2017.
URL http://dx.doi.org/10.1007/978-3-319-63390-9_23

To reduce a problem, provided in a human language, to constraint solving, one normally maps it to a set of constraints, written in the language of a suitable logic. We highlight a different paradigm, called Correct-by-Decision (CBD), in which the original problem is converted into a set of constraints and a decision strategy, where the decision strategy is essential for guaranteeing the correctness of the modeling. We have successfully applied CBD at Intel for designing scalable SAT-based solutions for several sub-stages of the Physical Design stage of chip design [1, 2, 3]. During our talk, we will walk through an example CBD application for solving the problem of routing under constraints and discuss some open questions, related to CBD.

References

1 Amit Erez and Alexander Nadel. Finding bounded path in graph using SMT for automatic clock routing. In Daniel Kroening and Corina S. Pasareanu, editors, *Computer Aided Verification – 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, volume 9207 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2015.

2 Alexander Nadel. Routing under constraints. In Ruzica Piskac and Muralidhar Talupur, editors, *2016 Formal Methods in Computer-Aided Design, FMCAD 2016, Mountain View, CA, USA, October 3-6, 2016*, pages 125–132. IEEE, 2016.
3 Alexander Nadel. A correct-by-decision solution for simultaneous place and route. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification – 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, pages 436–452. Springer, 2017.

3.17 A Resolution-Based Calculus for Preferential Logics

Claudia Nalon (University of Brasilia, BR)

Joint work of Claudia Nalon, Dirk Pattinson
Main reference Cláudia Nalon, Dirk Pattinson: "A Resolution-Based Calculus for Preferential Logics", in Proc. of the Automated Reasoning – 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Lecture Notes in Computer Science, Vol. 10900, pp. 498–515, Springer, 2018.
URL http://dx.doi.org/10.1007/978-3-319-94205-6_33

Preferential logics are part of a family of conditional logics intended for counterfactual reasoning, allowing to infer and to withdraw conclusions in the presence of new facts. Sequent or tableau calculi for such logics are notoriously hard to construct, and often require additional syntactic structure. Various conditional logics require nested sequents, labelled sequents or special transition formulae, together with non-trivial proofs of either semantic completeness or cut elimination. In this talk, we present a recently developed resolution-based calculus for the preferential logic S and argue that its pure syntactic nature makes it well suited for automation.

3.18 Logically Constrained Rewriting over Bit Vectors

Naoki Nishida (Nagoya University, JP)

Joint work of Naoki Nishida, Yoshiaki Kanazawa
Main reference Yoshiaki Kanazawa, Naoki Nishida, and Masahiko Sakai: "On representation of structures and unions in logically constrained rewriting", IEICE Technical Report SS2018-38, IEICE, Vol. 118, No. 385, pp. 67–72, January 1019 (in Japanese).
URL https://www.ieice.org/ken/paper/20190116P1Jg/eng/

Recently, several methods for verifying imperative programs by means of transformations into Term Rewrite Systems (TRSs, for short) have been investigated. In particular, constrained rewrite systems are popular as sources of such transformations, since logical constraints used for modeling control flows can be separated from terms that represent intermediate states of the execution of target programs. In the existing methods, data types that can be used in target programs are restricted to the integers and their one-dimensional arrays [1], and hence we are not allowed to use other primitive data types, structures, or unions.

In this talk, we briefly introduce Logically Constrained TRSs (LCTRSs, for short) over the bit vectors, which are obtained from C programs with structures and unions. Such LCTRSs can be models of automotive embedded systems, and are useful to verify the corresponding programs. Our framework of rewriting induction, a verification method of equivalence of two functions, works for not only LCTRSs over the integers but also LCTRSs over the bit vectors.

References

1 Carsten Fuhs, Cynthia Kop, and Naoki Nishida. Verifying procedural programs via constrained rewriting induction. *ACM Transactions on Computational Logic*, 18(2):14:1–14:50, June 2017.

3.19 Using SMT solvers to reason about firewalls

Ruzica Piskac (Yale University – New Haven, US)

Joint work of William T. Hallahan, Ennan Zhai, Ruzica Piskac
Main reference William T. Hallahan, Ennan Zhai, Ruzica Piskac: "Automated repair by example for firewalls", in Proc. of the 2017 Formal Methods in Computer Aided Design, FMCAD 2017, Vienna, Austria, October 2-6, 2017, pp. 220–229, IEEE, 2017.
URL http://dx.doi.org/10.23919/FMCAD.2017.8102263

This work present a systematic effort that can automatically repair firewalls, using the programming by example approach. We encode firewall behavior as a set of first-order logic formulas. In our approach, after administrators observe undesired behavior in a firewall, they may provide input/output examples that comply with the intended behavior. Based on the given examples, we automatically synthesize new firewall rules for the existing firewall. This new firewall correctly handles packets specified by the examples, while maintaining the rest of the behavior of the original firewall. Through a conversion of the firewalls to SMT formulas, we offer formal guarantees that the change is correct. Our evaluation results from real-world case studies show that our tool can efficiently find repairs.

3.20 Inductive Inference with Recursion Analysis in Separation Logic

Quang Loc Le (Teesside University – Middlesbrough, GB)

Inductive inference is a vital ingredient of a proof system for reasoning about recursive data structures (e.g., lists and trees) and functions. Especially, supporting an automated inductive theorem prover in a substructural logic, e.g. separation logic, is notoriously hard. In this work, we consider inductive inference for entailment problem in separation logic combined with inductive definitions and arithmetic. We present a novel inductive entailment prover where inductive inference is based on both circular reasoning (in the spirit of Brotherston) and mathematical induction (based on Noetherian principle). The essence of our proposal is a recursion analysis for automatically generating induction rules such that inductive proofs could be obtained locally and efficiently. We have implemented our proposal in a prototype tool and evaluated it over a set of challenging entailment problems taken from a recent competition for solvers in separation logic. The experimental results show that our prover is both effective and efficient. Indeed, it outperformed all existing state-of-the-art entailment solvers.

3.21 Probabilistic Symbolic Execution using Separation Logic

Quoc-Sang Phan (Synopsys Inc. – Mountain View, US)

Probabilistic symbolic execution is a technique to calculate the probability that a program reaches a certain point, which is useful in, for example, reliability analysis. So far this technique has been widely used in programs with numerical inputs, but it enjoys little success when the program makes extensive use of dynamically allocated data structures, such as lists and trees. In this talk, I will present our work-in-progress approach to this problem using separation logic.

References

1 Antonio Filieri, Marcelo F. Frias, Corina S. Pasareanu, Willem Visser. *Model Counting for Complex Data Structures*. SPIN 2015.

2 Long H. Pham, Quang Loc Le, Quoc-Sang Phan, Jun Sun, and Shengchao Qin. *Enhancing Symbolic Execution of Heap-based Programs with Separation Logic for Test Input Generation*. ATVA 2019.

3 Long H. Pham, Quang Loc Le, Quoc-Sang Phan, and Jun Sun. *Concolic Testing Heap-Manipulating Programs*. FM 2019.

3.22 Code commutation

Albert Rubio (Complutense University of Madrid, ES)

Joint work of Elvira Albert, Miguel Gómez-Zamalloa, Miguel Isabel, Albert Rubio
Main reference Elvira Albert, Miguel Gómez-Zamalloa, Miguel Isabel, Albert Rubio: "Constrained Dynamic Partial Order Reduction", in Proc. of the Computer Aided Verification – 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 10982, pp. 392–410, Springer, 2018.
URL http://dx.doi.org/10.1007/978-3-319-96142-2_24

Two pieces of code commute if they reach the same state in any of the two orders of execution. They can commute unconditionally if they commute in any possible initial state or conditionally if it is only for a subset of the initial states. We will present the use of this property in the context of the dynamic partial order technique. The property can be analyzed automatically using SMT solvers, but it becomes challenging when the code involved in the two pieces of code is complex.

3.23 Bit-Vector Interpolation and Quantifier Elimination by Lazy Reduction

Philipp Rümmer (Uppsala University, SE)

Joint work of Backeman, Peter; Zeljic, Aleksandar
Main reference Peter Backeman, Philipp Rümmer, Aleksandar Zeljic: "Bit-Vector Interpolation and Quantifier Elimination by Lazy Reduction", in Proc. of the 2018 Formal Methods in Computer Aided Design, FMCAD 2018, Austin, TX, USA, October 30 – November 2, 2018, pp. 1–10, IEEE, 2018.
URL http://dx.doi.org/10.23919/FMCAD.2018.8603023

The inference of program invariants over machine arithmetic, commonly called bit-vector arithmetic, is an important problem in verification. Techniques that have been successful for unbounded arithmetic, in particular Craig interpolation, have turned out to be difficult to generalise to machine arithmetic: existing bit-vector interpolation approaches are based either on eager translation from bit-vectors to unbounded arithmetic, resulting in complicated constraints that are hard to solve and interpolate, or on bit-blasting to propositional logic, in the process losing all arithmetic structure. We present a new approach to bit-vector interpolation, as well as bit-vector quantifier elimination (QE), that works by lazy translation of bit-vector constraints to unbounded arithmetic. Laziness enables us to fully utilise the information available during proof search (implied by decisions and propagation) in the encoding, and this way produce constraints that can be handled relatively easily by existing interpolation and QE procedures for Presburger arithmetic. The lazy encoding is complemented with a set of native proof rules for bit-vector equations and non-linear (polynomial) constraints, this way minimising the number of cases a solver has to consider.

3.24 Forgetting for Computing Snap-Shots of Ontologies: Progress and Challenges

Renate Schmidt (University of Manchester, GB)

Main reference Jieying Chen, Ghadah Alghamdi, Renate A. Schmidt, Dirk Walther, Yongsheng Gao: "Ontology Extraction for Large Ontologies via Modularity and Forgetting", in Proc. of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019, pp. 45–52, ACM, 2019.
URL http://dx.doi.org/10.1145/3360901.3364424

Forgetting is non-standard reasoning technology to restrict the information in a knowledge base by excluding some of the terms and symbols in the signature. My presentation focussed on the use of forgetting for ontology extraction. Because ontologies can be very large, it is useful to have tools that extract and create snap-shots of an ontology. The creation of ontology extracts is an essential operation for the reuse, creation, evaluation, curation, decomposition, integration and general use of ontologies. For example, it allows ontology modellers to create and work with extracts of an ontology that succinctly summarise the information relating to particular terms in the ontology. These could be used to create new smaller ontologies tailor-made for a particular purpose required by a new application of specific vendors.

After a brief introduction of forgetting the presentation discussed a trial of current forgetting tools in an industry collaboration with SNOMED International and the challenges that this research has thrown up.

For the SNOMED use case the ability to produce extracts for very small signatures compared to the signature of the whole ontology (1% or less) was required. The smaller the extract signature, the more work forgetting tools have in order to compute an extract. The research found that what helps is pre-computing a module and then applying forgetting; in addition signature extension was needed to allow modellers to use existing refsets of concept names. A novel workflow was developed consisting of four stages: (a) signature extension, (b) ontology module extraction, (c) forgetting, and (d) feedback by domain experts, which was evaluated on the SNOMED CT and NCIt ontologies. The investigation used three different modularisation approaches (locality-based, semantic and minimal subsumption modularisation) and three forgetting tools (NUI, LETHE and FAME).

Discussion of the various challenges that remain to be addressed has revealed useful suggestions for improving the ontology extraction process and an alternative faster approach.

3.25 Efficient Validation of FOLID Cyclic Induction Reasoning

Sorin Stratulat (University of Lorraine – Metz, FR)

Checking the soundness of the cyclic induction reasoning for first-order logic with inductive definitions (FOLID) may be costly; the standard checking method is decidable but based on a doubly exponential complement operation for Büchi automata. In this talk, I will present a semi-decidable polynomial method whose most expensive steps recall the comparisons with multiset path orderings. In practice, it has been integrated in the Cyclist prover and successfully checked all the proofs generated with the standard method and included in its distribution.

FOLID cyclic proofs may also be hard to certify. Our method helps to represent the cyclic induction reasoning as being well-founded, where the ordering constraints are automatically built from the analysis of the proofs. Hence, it creates a bridge between the two induction reasoning methods and opens the perspective to use the certification methods adapted for well-founded induction proofs.

3.26 Rule-Based Nonmonotonic Reasoning with Probabilities

Andrzej Szalas (University of Warsaw, PL)

Main reference Andrzej Szałas: "Decision-Making Support Using Nonmonotonic Probabilistic Reasoning". In: Czarnowski I., Howlett R., Jain L. (eds) Intelligent Decision Technologies 2019. Smart Innovation, Systems and Technologies, vol 142. Springer, Singapore
URL https://doi.org/10.1007/978-981-13-8311-3_4

The talk will be focused on a decision-making support by rule-based nonmonotonic reasoning enhanced with probabilities. As a suitable rule-based tool we will analyze Answer Set Programming (ASP) and explore its probabilistic extension permitting the use of probabilistic expressions of two types. The first type represents an externally given prior probability distribution on literals in an answer set program P. The second type represents a posterior distribution conditioned on individual decisions and choices made, together with their consequences represented by answer sets of P.

The ability to compare aspects of both the prior and posterior probabilities in the language of the program P has interesting uses in filtering solutions/decisions one is interested in. A formal characterization of this probabilistic extension to ASP as well as some examples demonstrating its potential use will also be discussed.

The discussed techniques do not increase the complexity of standard ASP-based reasoning.

3.27 A Fixpoint Logic and Dependent Effects for Temporal Property Verification

Tachio Terauchi (Waseda University – Tokyo, JP)

Joint work of Yoji Nanjo, Hiroshi Unno, Eric Koskinen, Tachio Terauchi
Main reference Yoji Nanjo, Hiroshi Unno, Eric Koskinen, Tachio Terauchi: "A Fixpoint Logic and Dependent Effects for Temporal Property Verification", in Proc. of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018, pp. 759–768, ACM, 2018.
URL http://dx.doi.org/10.1145/3209108.3209204

Existing approaches to temporal verification of higher-order functional programs have either sacrificed compositionality in favor of achieving automation or vice-versa. In this paper we present a dependent-refinement type & effect system to ensure that welltyped programs satisfy given temporal properties, and also give an algorithmic approach—based on deductive reasoning over a fixpoint logic–to typing in this system. The first contribution is a novel type-and-effect system capable of expressing dependent temporal effects, which are fixpoint logic predicates on event sequences and program values, extending beyond the (non-dependent) temporal effects used in recent proposals. Temporal effects facilitate compositional reasoning whereby the temporal behavior of program parts are summarized as effects and combined to form those of the larger parts. As a second contribution, we show that type checking and typability for the type system can be reduced to solving first-order fixpoint logic constraints. Finally, we present a novel deductive system for solving such constraints. The deductive system consists of rules for reasoning via invariants and well-founded relations, and is able to reduce formulas containing both least and greatest fixpoints to predicate-based reasoning.

3.28 Abduction in DL by translation to FOL

Sophie Tourret (MPI für Informatik – Saarbrücken, DE) and Christoph Weidenbach (MPI für Informatik – Saarbrücken, DE)

Joint work of Sophie Tourret, Patrick Koopman, Christoph Weidenbach

Description Logics (DL) are the languages of choice to reason about real world knowledge as represented in web ontologies. One recurrent issue with ontologies is their incompleteness. Abductive reasoning is one way to repair such faulty systems by automatically computing possible explanations for observations that, although not entailed by the ontology, do not contradict it. Most existing abduction techniques for DL have a limited expressiveness and do not scale well. By relying on state-of-the-art tools for abduction in first-order logic, based on SMT, we want to improve on the current situation.

3.29 On Hierarchical Symbol Elimination and Applications

Viorica Sofronie-Stokkermans

Joint work of Viorica Sofronie-Stokkermans, Dennis Peuter

We present possibilities of symbol elimination in extensions of a theory T_0 with additional function symbols whose properties are axiomatised using a set of clauses which we established in [1] and [2]. We analyze situations in which we can perform such tasks in a hierarchical way, relying on existing mechanisms for symbol elimination in T_0. This is for instance possible if the base theory T_0 allows quantifier elimination. We discuss possibilities of extending such methods to situations in which the base theory does not allow quantifier elimination but has a model completion which does (or, in some cases, has a co-theory in which symbol elimination is possible). We discuss the way these results can be used e.g. for abduction, interpolant generation (cf. e.g. [1], [2]), and invariant strengthening [3].

References

1 V. Sofronie-Stokkermans. On Interpolation and Symbol Elimination in Theory Extensions N. Olivetti and A. Tiwari (eds), *Automated Reasoning – 8th International Joint Conference, IJCAR 2016*, LNCS 9706, pages 273–289, Springer (2016)

2 V. Sofronie-Stokkermans. On Interpolation and Symbol Elimination in Theory Extensions. Logical Methods in Computer Science 14(3) (2018)

3 D. Peuter and V. Sofronie-Stokkermans. On Invariant Synthesis for Parametric Systems. In: Fontaine P (ed.), *Automated Deduction – CADE 27 – 27th International Conference on Automated Deduction*, LNCS 11716, pages 385–405, Springer (2019).

3.30 Saturation Theorem Proving: From Inference Rules to Provers

Uwe Waldmann (MPI für Informatik – Saarbrücken, DE)

Joint work of Uwe Waldmann, Jasmin Blanchette, Simon Robillard, Sophie Tourret

One of the indispensable operations of realistic saturation theorem provers is (backward and forward) deletion of subsumed formulas. In presentations of proof calculi, however, subsumption deletion is usually discussed only informally, and in the rare cases where there is a formal exposition, it is typically clumsy. The main reason for this is the fact that the well-known equivalence of dynamic and static refutational completeness holds only for derivations where all deleted formulas are redundant, but using a standard notion of redundancy, a clause C does not make an instance Cσ redundant.

We are working on a generic framework for formal refutational completeness proofs of abstract provers that implement saturation proof calculi. The framework relies on a modular extension of lifted redundancy criteria, which in the end permits not only to cover subsumption deletion, but to model entire prover architectures in such a way that the static refutational completeness of a calculus immediately implies the dynamic refutational completeness of, say, an Otter loop or Discount loop prover implementing the calculus.

A formal proof in Isabelle is currently under development.

3.31 Loop Detection by Logically Constrained Rewriting

Sarah Winkler (University of Verona, IT)

Joint work of Naoki Nishida, Sarah Winkler
Main reference Naoki Nishida, Sarah Winkler: "Loop Detection by Logically Constrained Term Rewriting", in Proc. of the Verified Software. Theories, Tools, and Experiments – 10th International Conference, VSTTE 2018, Oxford, UK, July 18-19, 2018, Revised Selected Papers, Lecture Notes in Computer Science, Vol. 11294, pp. 309–321, Springer, 2018.
URL http://dx.doi.org/10.1007/978-3-030-03592-1_18

Logically constrained rewrite systems (LCTRSs) constitute a very general rewriting formalism that captures simplfication processes in various domains, as well as computation in imperative programs. In both of these contexts, nontermination is a critical source of errors. This talk discusses loop criteria for LCTRSs that are implemented in the tool Ctrl. The usefulness of these criteria is illustrated by applications in four domains: checking (a) LLVM peephole optimizations as well as (b) simplification rules of SMT solvers for potential loops, (c) detecting looping executions of C programs, and (d) establishing nontermination of integer transition systems.

Participants

- Alexander Bentkamp
Free University Amsterdam, NL
- Nikolaj S. Bjørner
Microsoft Research – GRedmond, US
- Maria Paola Bonacina
Università degli Studi di Verona, IT
- Florent Capelli
INRIA Lille, FR
- Warren Del-Pinto
University of Manchester, GB
- Rayna Dimitrova
University of Leicester, GB
- Pascal Fontaine
LORIA & INRIA – Nancy, FR
- Florian Frohn
MPI für Informatik – Saarbrücken, DE
- Carsten Fuhs
Birkbeck, University of London, GB
- Jürgen Giesl
RWTH Aachen, DE
- Alberto Griggio
Bruno Kessler Foundation – Trento, IT
- Arie Gurfinkel
University of Waterloo, CA
- Reiner Hähnle
TU Darmstadt, DE
- Matthias Heizmann
Universität Freiburg, DE
- Benjamin Kaminski
RWTH Aachen, DE
- Laura Kovács
TU Wien, AT
- Quang Loc Le
Teesside University – Middlesbrough, GB
- Alexander Leitsch
TU Wien, AT
- Anthony W. Lin
TU Kaiserslautern, DE
- Joao Marques-Silva
Federal University – Toulouse, FR
- David Monniaux
VERIMAG – Grenoble, FR
- Alexander Nadel
Intel Israel – Haifa, IL
- Claudia Nalon
University of Brasilia, BR
- Naoki Nishida
Nagoya University, JP
- Quoc Sang Phan
Synopsys Inc. – Mountain View, US
- Ruzica Piskac
Yale University – New Haven, US
- Albert Rubio
Complutense University of Madrid, ES
- Philipp Rümmer
Uppsala University, SE
- Andrey Rybalchenko
Microsoft Research – Cambridge, GB
- Renate Schmidt
University of Manchester, GB
- Martina Seidl
Johannes Kepler Universität Linz, AT
- Viorica Sofronie-Stokkermans
Universität Koblenz-Landau, DE
- Sorin Stratulat
University of Lorraine – Metz, FR
- Andrzej Szalas
University of Warsaw, PL
- Tachio Terauchi
Waseda University – Tokyo, JP
- Cesare Tinelli
University of Iowa – Iowa City, US
- Sophie Tourret
MPI für Informatik – Saarbrücken, DE
- Andrei Voronkov
University of Manchester, GB & EasyChair
- Uwe Waldmann
MPI für Informatik – Saarbrücken, DE
- Christoph Weidenbach
MPI für Informatik – Saarbrücken, DE
- Thomas Wies
New York University, US
- Sarah Winkler
University of Verona, IT