Schedule for Dagstuhl workshop 25231 "Certifying Algorithms for Automated Reasoning"

Time (ca)	MONDAY JUN 2	TUESDAY JUN 3	WEDNESDAY JUN 4	THURSDAY JUN 5	FRIDAY JUN 6	Time (ca)
08:45	**8:45-9:00:** Welcome					08:45
09:00	**9:00-9:50**	**9:00-9:50**	**9:00-9:50**	**9:00-9:30**	**9:00-9:30**	09:00
09:15	Certified SAT solving	Certified automated planning	Certified QBF solving	Janota: Graph symmetries	Weidenbach: Certification in SCL	09:15
09:30	Fazekas	Helmert	Seidl	**9:30-10:00**	**9:30-10:00**	09:30
09:45	**9:50-10:40**	**9:50-10:40**	**9:50-10:40**	Szeider: Symmetries in SAT & QBF	Sutcliffe: Proof verification GDV	09:45
10:00	Certified subgraph solving	Certified SMT solving	Certified first-order theorem proving	**10:00-10:30**	**10:00-10:30**	10:00
10:15	McCreesh	Barbosa	Rawson	Anders: Faster certified symmetries	Schindler: Certified optimal planning	10:15
10:30	***COFFEE BREAK***	***COFFEE BREAK***	***COFFEE BREAK***	***COFFEE BREAK***	***COFFEE BREAK***	10:30
10:45						10:45
11:00	**11:10-12:00**	**11:10-12:00**	**11:10-12:00**	**11:00-11:30**	**11:00-11:30**	11:00
11:15	Certified constraint programming	Certified model counting and	Formally verified	Reynolds: Proofs for cvc5	Fleury: Consuming CaDiCaL proofs	11:15
11:30	McIlree	knowledge compilation	proof checking	**11:30-12:00**	**11:30-12:00**	11:30
11:45		Bryant	Myreen & Tan	Gurfinkel: Speculative SAT mod SAT	Oliveras: Symbolic conflict analysis	11:45
12:00						12:00
12:15	***LUNCH***	***LUNCH***	***LUNCH***	***LUNCH***	***LUNCH***	12:15
12:30						12:30
12:45					***WORKSHOP ENDS***	12:45
13:00						13:00
13:15					***LEGEND***	13:15
13:30					*Long talk (50 min)*	13:30
13:45					*Short talk (30 min)*	13:45
14:00	**14:10-15:00**		***WEDNESDAY***		*Other*	14:00
14:15	Proof logging for					14:15
14:30	algebraic algorithms		***AFTERNOON***	**14:30-15:30**		14:30
14:45	Kaufmann			Panel discussion		14:45
15:00	***COFFEE & CAKE***	**15:00-15:30**	***FREE***	*The future of certifying algorithms*		15:00
15:15		Bogaerts: Certified pareto optimality				15:15
15:30		***COFFEE & CAKE***		***COFFEE & CAKE***		15:30
15:45	**15:40-16:30**					15:45
16:00	Proof logging for MIP	**16:00-16:30**		**16:00-16:30**		16:00
16:15	Gleixner	Beyer: Verification witnesses		Baader: Certifying subsumptions		16:15
16:30	**16:30-17:00**	**16:30-17:00**		**16:30-17:00**		16:30
16:45	Koops: Certified PB optimization	Biere: Certified HW model checking		Abraham: Certifying real algebra		16:45
17:00	**17:00-17:30**	**17:00-17:30**				17:00
17:15	Oertel: Certified preprocessing	Seisenberger: Railway verification		***FREE TIME FOR***		17:15
17:30	**17:30-18:00**	**17:30-18:00**		***DISCUSSIONS***		17:30
17:45	Presentation of participants	Reeves: Trimming SMT proofs				17:45
18:00	**DINNER**	**DINNER**	**DINNER**	**DINNER**		18:00
18:15						18:15

Dagstuhl Workshop 25231

Certifying Algorithms for Automated Reasoning

Sunday, June 1st, 2025 – Friday, June 6th, 2025

- Nikolaj S. Bjørner, Microsoft - Redmond, US
- Marijn J. H. Heule, Carnegie Mellon University - Pittsburgh, US
- Daniela Kaufmann, TU Wien, AT
- Jakob Nordström, University of Copenhagen, DK & Lund University, SE

Please feel free to direct any questions to us!

General Information

- Please be aware that you will need the username and password of your DOOR account (i.e., the credentials you used to register for the seminar) while in Dagstuhl.
- The Dagstuhl reception is located in the facility opposite the manor house. The reception is open from 15:00 to 19:00 on Sunday and from 8:00 to 16:00 on other days. If it is closed when you arrive, please use the access code in your voucher to enter the building and then follow the self-service check-in procedure described at the reception.
- Departure is on Friday June 6. Dagstuhl kindly asks you to clear your room by 9:00 and to pay your bill for accommodation, meals and your private expenses on the day of your departure before lunch.

Meals

- Breakfast is served from 7:30 to 8:45.
- Lunch is served at 12:15.
- Dinner is served at 18:00.
- During the morning break, coffee and tea are available outside the seminar room.
- In the afternoon, coffee and cake are served between 15:00 and 16:00 in the dinner hall.
- In the evening, a cheese platter is served from 20:00 in the castle lounge.

Sunday, June 1st, 2025

15:00 Dagstuhl reception opens

18:00 Buffet dinner

Welcome!

20:00 Informal gathering in the lounge in the castle (if desired)

Beverages and a small assortment of snacks are available on a cash honour system.

Monday, June 2nd, 2025

07:30–08:45	Breakfast	
08:45–09:00	All participants Welcome	
09:00–09:50	Katalin Fazekas Certified SAT Solving	Morning session
09:50–10:40	Ciaran McCreesh Proof Logging for Subgraph-Finding Algorithms	
10:40–11:10	Coffee break	
11:10–12:00	Matthew McIlree Certified Constraint Programming	
12:15	Lunch	
14:10–15:00	Daniela Kaufmann Certifying Ideal Membership Tests	Afternoon session
15:00–15:40	Coffee and cake	
15:40–16:30	Ambros Gleixner Proof logging for Mixed-Integer Programming	
16:30–17:00	Wietze Koops Practically Feasible Proof Logging for Pseudo-Boolean Optimization	
17:00–17:30	Andy Oertel Certifying Presolving/Preprocessing for 0-1 Integer Linear Programming and MaxSAT	
17:30–18:00	Presentation of participants	
18:00	Dinner	
20:00	Cheese platter	

Tuesday, June 3rd, 2025

07:30–08:45	Breakfast	
09:00–09:50	Malte Helmert Certified Automated Planning	Morning session
09:50–10:40	Haniel Barbosa SMT Proof Production, Checking and Reconstruction	
10:40–11:10	Coffee break	
11:10–12:00	Randal E. Bryant Checkable Proofs for Model Counting and Knowledge Compilation	
12:15	Lunch	
15:00–15:30	Bart Bogaerts Certifying Pareto Optimality in Multi-Objective Maximum Satisfiability	Afternoon session
15:30–16:00	Coffee and cake	
16:00–16:30	Dirk Beyer Certifying Software Verification	
16:30–17:00	Armin Biere Certifying Hardware Model Checking	
17:00–17:30	Monika Seisenberger Certifying RUP proofs in the context of Railway Verification	
17:30–18:00	Joseph Reeves Theory Lemma Trimming for SMT Proof Skeletons	
18:00	Dinner	
20:00	Cheese platter	

Wednesday, June 4th, 2025

07:30–08:45	Breakfast	
09:00–09:50	Martina Seidl Certified QBF Solving	Morning session
09:50–10:40	Michael Rawson Certified First-Order Theorem Proving: confessions, excuses and a few ways out.	
10:40–11:10	Coffee break	
11:10–12:00	Yong Kiam Tan & Magnus O. Myreen The Past, Present, and Future of Verified Proof Checkers	
12:15	Lunch	
18:00	Dinner	
20:00	Cheese platter	

Thursday, June 5th, 2025

07:30–08:45 Breakfast

Morning session

09:00–09:30 Mikoláš Janota
Graph Symmetries, Patterns, and Encodings

09:30–10:00 Stefan Szeider
Certifying Dynamic Symmetry Breaking in SAT and QBF

10:00–10:30 Markus Anders
Faster Certified Symmetry Breaking in SAT

10:30–11:00 Coffee break

11:00–11:30 Andrew Reynolds
Engineering Complete SMT Proofs in cvc5 with Ethos/Eunoia

11:30–12:00 Arie Gurfinkel
Speculative SAT modulo SAT

12:15 Lunch

Afternoon session

14:30–15:30 Panel discussion *The future of certifying algorithms*

15:30–16:00 Coffee & cake

16:00–16:30 Franz Baader
First Results on How to Certify Subsumptions Computed by the EL Reasoner ELK Using the Logical Framework with Side Conditions

16:30–17:00 Erika Abraham
The certification problem for real algebra

17:00–18:00 Free time for discussions

18:00 Dinner

20:00 Cheese platter

Friday, June 6th, 2025

07:30–08:45	Breakfast	
09:00–09:30	Christoph Weidenbach Certification in SCL	Morning session
09:30–10:00	Geoff Sutcliffe Proof Verification with GDV and LambdaPi - It's a Matter of Trust	
10:00–10:30	Tanja Schindler Proof Logging for Optimal Planning	
10:30–11:00	Coffee break	
11:00–11:30	Mathias Fleury How to consume CaDiCaL proofs?	
11:30–12:00	Albert Oliveras PB Symbolic Conflict Analysis	
12:15	Lunch	

Abstracts

Erika Abraham

The certification problem for real algebra

SMT solvers' traditional functionality is to check the satisfiability of quantifier-free formulas of first-order logic over different theories.
With their increasing efficiency and usage, this original functionality is being extended in different directions. One if them is the ability to provide some kind of assurance for the correctness of the computations, most prominently in the form of certificates.
Whereas some SMT solvers can already provide certificates for a wide range of theories, the theory of (quantifier-free non-linear) real algebra poses a hard challenge, and a solution seems to be yet completely out of reach.
In this talk, we discuss why this problem is especially hard, and which directions could be considered to make some progress.

Markus Anders

Faster Certified Symmetry Breaking in SAT

Symmetry breaking is a standard technique in many areas of automated reasoning. Recently, the possibility for proof logging symmetry breaking techniques in SAT solvers has become available by means of the dominance rule and VeriPB proof system [Bogaerts et al., 2023]. It turns out however, that the proposed logging and checking techniques pose a severe bottleneck for efficient, modern symmetry handling algorithms. In this talk, I will give a brief overview of symmetry handling algorithms and related proof logging techniques. Then, I will discuss recent developments to improve logging and checking performance, as well as some of the remaining challenges.

Franz Baader

First Results on How to Certify Subsumptions Computed by the EL Reasoner ELK Using the Logical Framework with Side Conditions

The generation of proof certificates and the use of proof checkers is nowadays standard in first-order automated theorem proving and related areas. They have, to the best of our knowledge, not yet been employed in Description Logics, where the focus was on detecting and repairing errors in the ontology, rather than on catching erroneous consequences created by an incorrect reasoner. This paper reports on first steps towards remedying this deficit for subsumptions computed by the DL reasoner Elk. We use an existing tool for generating proofs of consequences from Elk, and transform these proofs into a format that is accepted as certificates by our proof checker. The checker is obtained as an instance of a generic certification tool based on the Logical Framework with Side Conditions (LFSC), by formalizing the inference rules of Elk in LFSC. We report on the results of applying this approach to the classification of a large number of real-world OWL 2 EL ontologies.

Haniel Barbosa

SMT Proof Production, Checking and Reconstruction

SMT solvers can be hard to trust, since it generally means assuming their large and complex codebases do not contain bugs leading to wrong results. Machine-checkable certificates, via proofs of the logical reasoning the solver has performed, address this issue by decoupling confidence in the results from the solvers implementation. In this talk we will describe the extensive proof infrastructure of the state-of-the-art SMT solver cvc5, which has enabled the production of proofs in a number of complex domains. We will also show how these proofs are checked or reconstructed in different formats by different systems, from ad-hoc high-performance proof checkers to proof assistants such as Lean.

Dirk Beyer

Certifying Software Verification

Over the last years, certifying software verification has become an established practice in the area of automatic software verification: An independent validator re-establishes verification results of a software verifier using verification certificates (also called witnesses), which are stored in a standardized exchange format. In addition to validation, such exchangeable information about proofs and alarms found by a verifier can be shared across verification tools, and users can apply independent third-party tools to visualize and explore certificates to help them comprehend the causes of bugs or the reasons why a given program is correct. To achieve the goal of making verification results more accessible to engineers, it is necessary to consider certificates as first-class exchangeable objects, stored independently from the source code and checked independently from the verifier that produced them, respecting the important principle of separation of concerns. We present the conceptual principles of software-verification certificates and illustrate the contents of such certificates.

Material:

- Software Verification Witnesses 2.0 `https://doi.org/10.1007/978-3-031-66149-5_11`
- Verification Witnesses `https://doi.org/10.1145/3477579`

Armin Biere

Certifying Hardware Model Checking

Design faults in hardware design are costly. Thus hardware model checking has routinely been applied during the chip design process for decades. However, both academic and industrial model checkers are complex software tools and arguably hard to get correct. To increase trust in model checkers we therefore propose a model checking certification flow. The model checker produces a witness circuit which simulates the original model and for safety properties has an inductive property implying the original property. Checking simulation and inductiveness can be done by SAT solving. We have applied this idea to different model checking techniques, particularly preprocessing techniques. The single safety property track of the hardware model checking competition in 2024 required all participants to produce such certificates. The competition showed that certification is possible and cheap, i.e., both with respect to certificate production and checking. Furthemore the winner of the competition surpases the previous state-of-the-art, while producing machine checked witnesses. This is joint work with Emily Yu, Nils Froleyks, Mathias Preiner and Keijo Heljanko.

Bart Bogaerts

Certifying Pareto Optimality in Multi-Objective Maximum Satisfiability

Due to the wide employment of automated reasoning in the analysis and construction of correct systems, the results reported by automated reasoning engines must be trustworthy. For Boolean satisfiability (SAT) solversand more recently SAT-based maximum satisfiability (MaxSAT) solverstrustworthiness is obtained by integrating proof logging into solvers, making solvers capable of emitting machine-verifiable proofs to certify correctness of the reasoning steps performed. In this work, we enable for the first time proof logging based on the VeriPB proof format for multi-objective MaxSAT (MO-MaxSAT) optimization techniques. Although VeriPB does not offer direct support for multiobjective problems, we detail how preorders in VeriPB can be used to provide certificates for MO-MaxSAT algorithms computing a representative solution for each element in the non-dominated set of the search space under Pareto-optimality, without extending the VeriPB format or the proof checker. By implementing VeriPB proof logging into a stateof-the-art multi-objective MaxSAT solver, we show empirically that proof logging can be made scalable for MO-MaxSAT with reasonable overhead.

Randal E. Bryant

Checkable Proofs for Model Counting and Knowledge Compilation

Knowledge compilers convert Boolean formulas, given in conjunctive normal form (CNF), into representations that

enable efficient evaluation of unweighted and weighted model counts, as well as a variety of other useful properties. Certifying the correctness of a knowledge compilers output, requires proving that 1) the generated formula is logically equivalent to the input formula, and 2) the generated formula satisfies the structural properties that enable efficient model counting.
Our Certified Partitioned-Operation Graph (CPOG) proof framework provides a way to encode the output of a knowledge compiler as well as a set of steps providing a checkable proof of correctness. Most recently, we have extended this framework to Skolem CPOG (SCPOG) supporting projected knowledge compilation, where a subset of the variables is abstracted away via existential quantification. Doing so requires a method to encode Skolem assignments, describing instantiations of the quantified variables.
We have developed formally verified checkers for both CPOG and SCPOG, one in Lean4 and the other in CakeML/HOL. In doing so, we formally verified the soundness of the frameworks.

Katalin Fazekas

Certified SAT Solving

This talk explores certified SAT solving, which is crucial for establishing trust in the reasoning steps and results of Boolean Satisfiability (SAT) solvers. We will cover the related fundamental concepts of SAT solving and discuss how proof-producing solvers and external checkers enable certification. A key focus will be on certifying incremental SAT solving, an essential technique that allows solvers to efficiently tackle sequences of related problems while maintaining correctness guarantees.

Mathias Fleury

How to consume CaDiCaL proofs?

CaDiCaL is able to produce 3 different kind of proofs, DRAT, LRAT, and veriPB. In this talk we will discuss how (as a user) you can access to the steps and process them.

Ambros Gleixner

Proof logging for Mixed-Integer Programming

Standard solvers for mixed-integer linear programming rely define feasibility and optimality of solutions within numerical tolerances and the correctness of their results, even within these tolerances, is subject to roundoff errors stemming from the unsafe use of floating-point arithmetic. By contrast, starting with version 10, the open-source MIP solver SCIP ships a numerically exact solving mode without tolerances and can produce an independently verifiable proof log for most of the exact solving techniques. Besides giving an overview on these recent advances and remaining limitations in software for verified MIP solving, we try to gauge to what extent floating-point MIP solvers can be used directly to produce verifiably correct proof logs. Our computational study with a pure LP-based branch-and-bound version of SCIP confirms the expectation that in the overwhelming majority of cases, all critical decisions during the solving process are correct. When errors do occur on numerically challenging instances, they typically affect only a small, typically single-digit, amount of leaf nodes that would require further processing.

Arie Gurfinkel

Speculative SAT modulo SAT

State-of-the-art model-checking algorithms like IC3/PDR are based on unidirectional modular SAT solving for finding and/or blocking counterexamples. Modular SAT solvers divide a SAT-query into multiple sub-queries, each solved by a separate SAT solver (called a module), and propagate information (lemmas, proof obligations, blocked clauses, etc.) between modules. While modular solving is key to IC3/PDR, it is obviously not as effective as monolithic solving, especially when individual sub-queries are harder to solve than the combined query. This is partially addressed in SAT modulo SAT (SMS) by propagating unit literals back and forth between the modules and using information

from one module to simplify the sub-query in another module as soon as possible (i.e., before the satisfiability of any sub-query is established). However, bi-directionality of SMS is limited because of the strict order between decisions and propagation – only one module is allowed to make decisions, until its sub-query is SAT. In this talk, I will describe our generalization of SMS, called SPEC SMS, that speculates decisions between modules. This makes it bi-directional – decisions are made in multiple modules, and learned clauses are exchanged in both directions. We further extend DRUP proofs and interpolation, these are useful in model checking, to SPEC SMS. We have implemented SPEC SMS in Z3 and show that it performs exponentially better on a series of benchmarks that are provably hard for SMS.
This is joint work with Hari Govind V K, Isabel Garcia-Contreras and Sharon Shoham.

Malte Helmert

Certified Automated Planning

In my talk, I will introduce the classical planning problem and explain its relevance to the seminar by contrasting it with SAT. For those that haven't seen planning before, I hope to provide some basic understanding of the problem and why it is of interest. For those familiar with planning, I hope to give one or two additional perspectives on the problem and its complexity. Time permitting, I will also update the seminar participants on research in the planning community that tackles the main motivating questions of the seminar, in particular discussing results and open challenges for certifying planning algorithms.

Mikoláš Janota

Graph Symmetries, Patterns, and Encodings

With the objective of breaking symmetries in graphs, we define a graph's adjacency matrix to be canonical, if and only if it is lexicographically smallest among matrices of all isomorphic graphs. To break symmetries we introduce the notion of a pattern, which compactly represents a set of non-canonical graphs. We discuss how patterns can be reasoned about and encoded. Finally, we look at some experimental results for graphs with a small number of vertices.

Daniela Kaufmann

Certifying Ideal Membership Tests

Deciding ideal membership is a central problem in computer algebra, with wide-ranging applications in geometry, verification, and symbolic computation. While Gröbner bases provide a complete method for deciding ideal membership, their outputs are often complex, making certification difficult. I present a framework for certifying ideal membership tests using a practical algebraic calculus (PAC) that allows tracking polynomial manipulations. The calculus supports different levels of granularity, allowing proofs to be either concise or detailed; depending on whether the emphasis is on debugging or efficient proof checking.

Wietze Koops

Practically Feasible Proof Logging for Pseudo-Boolean Optimization

Certifying solvers have long been standard in Boolean satisfiability (SAT), allowing for proof logging and checking with limited overhead. However, developing similar tools for combinatorial optimization has remained a challenge. A recent promising approach covering a wide range of paradigms is pseudo-Boolean proof logging, but this has mostly consisted of proof-of-concept works far from delivering the performance required for real-world deployment.
In this work, we present an efficient toolchain based on VeriPB and CakePB for formally verified pseudo-Boolean optimization, and implement proof logging for the full range of techniques in the state-of-the-art solvers RoundingSat and Sat4j. Our experimental evaluation shows that proof logging and checking performance in this much more expressive paradigm is now quite close to the level of SAT solving, and hence clearly practically feasible.
This is joint work with Daniel Le Berre, Magnus Myreen, Jakob Nordström, Andy Oertel, Yong Kiam Tan, and Marc Vinyals.

Ciaran McCreesh

Proof Logging for Subgraph-Finding Algorithms

Many interesting problems involve finding a little graph inside a bigger graph: for example, maximum clique asks for the largest set of vertices where everything is adjacent, whilst subgraph isomorphism asks whether a specific pattern occurs inside a given target graph. Although these problems are computationally hard in theory, in practice solvers can often handle these problems extremely quickly, even on graphs with thousands of vertices. However, these solvers are not always perfect, and sometimes contain bugs that lead to wrong answers being produced. I'll explain how, using the VeriPB proof system, we can augment these solvers to produce correctness certificates, allowing us to be confident they have definitely given the right answers. To do this, we'll need to be able to justify a wide range of algorithmic inference steps, including colour bounds, all-different filtering, and degree reasoning; perhaps surprisingly, VeriPB is able to do all of these efficiently, despite not having any notion of what a graph is.

Matthew McIlree

Certified Constraint Programming

Constraint programming (CP) is a powerful paradigm for expressing and solving satisfaction and optimisation problems involving finite domain variables and high-level constraints. But the implementation and engineering of CP algorithms can be extremely complex, error-prone, and difficult to test. We are much more likely to trust the output of a solver if it can provide some kind of certificate of correctness via proof logging.

In this talk, I will discuss the current state of research into adding proof logging to CP solvers. I'll cover how we can prove unsatisfiability and optimality; what makes this different from established proof logging technology for SAT solvers; and the efforts towards devising efficient justification procedures for the huge variety of propagation algorithms available in the modern CP repertoire.

Yong Kiam Tan & Magnus O. Myreen

The Past, Present, and Future of Verified Proof Checkers

This survey talk will be split into two parts.

First, we will survey various automated reasoning theories/domains where verified proof checkers have been built. We will also present some of our ongoing work, and we will argue that verification can help enable the design of more complex proof systems/checkers while preserving trust in the overall certification process.

Then, we will take a deeper dive into verification infrastructure available in various theorem provers. Special focus will be given to HOL4 and the CakeML project—we have used CakeML to build several end-to-end verified proof checkers with machine-code level correctness guarantees. We will discuss synergies between what CakeML can bring to proof checking and what proof checking can bring to CakeML.

Webpage: https://cakeml.org/checkers.html

Andy Oertel

Certifying Presolving/Preprocessing for 0-1 Integer Linear Programming and MaxSAT

It is well known that reformulating the original problem can be crucial for the performance of mixed-integer programming (MIP) and maximum satisfiability (MaxSAT) solvers. While the idea in both the MIP and MaxSAT community is the same, the presolving reductions in MIP and preprocessing in MaxSAT apply different techniques to reformulate the problem. To ensure the correctness of the reformulations, all transformations must preserve the feasibility status and optimal value of the problem, but there is currently no established methodology to express and verify the equivalence of two optimization problems.

In this talk, it is presented how pseudo-Boolean proof logging can be used to certify the correctness of a wide range of modern MIP-based presolving and MaxSAT preprocessing techniques. By combining and extending the VeriPB and CakePB tools, we obtain a formally verified end-to-end proof checking tool chain to verify the correctness of reformulations of pseudo-Boolean problems.

This talk is based on the following two papers. The first paper was published at CPAIOR 2024 together with Alexander Hoen, Ambros Gleixner, and Jakob Norström. The second paper was published at IJCAR 2024 together with Hannes Ihalainen, Yong Kiam Tan, Jeremias Berg, Matti Järvisalo, Magnus O. Myreen, and Jakob Nordström.

Albert Oliveras

PB Symbolic Conflict Analysis

Pseudo-Boolean solvers can be used to deal with optimization by successively solving a series of problems that contain an additional pseudo-Boolean constraint expressing that a better solution is required. A key point for the success of this simple approach is that lemmas that are learned for one problem can be reused for subsequent ones.
In this talk we show how, by using a simple symbolic conflict analysis procedure, not only can lemmas be reused between problems but also strengthened, thus further pruning the search space traversal. In addition, we show how this technique automatically allows one to infer upper bounds in maximization problems, thus giving an estimation of how far the solver is from finding an optimal solution. An open problem is to use existing proof checking tools to certify the output of this procedure.

Michael Rawson

Certified First-Order Theorem Proving: confessions, excuses and a few ways out.

Automated Theorem Provers (ATPs) have been around a long time, but their proof certification ecosystem is nowhere near as well-developed as, say, SAT solvers. There are several reasons for this: a plurality of proof calculi, equisatisfiable inferences, and theories, to name a few. I will outline ATP systems and their proof certification, explain some difficult areas, present some recent developments, and offer a few paths to salvation.

Joseph Reeves

Theory Lemma Trimming for SMT Proof Skeletons

Automated reasoning tools require high trust, prompting modern solvers to produce proof certificates for verification. In propositional satisfiability (SAT), proof generation and checking are relatively inexpensive, but in satisfiability modulo theories (SMT), justifying theory lemmas can introduce significant overhead. Recent approaches mitigate this issue by having the SMT solver produce a proof skeleton containing only the propositional reasoning in the DRAT format and unjustified theory lemmas, whose justifications are deferred to the checking phase. Preprocessing, a key element of SMT solving that can be challenging to justify a posteriori, is not justified nor checked. We extend these approaches by including proofs for preprocessing; by reducing the checker workload via iteratively eliminating theory lemmas from proof skeletons through SAT solving and proof trimming; and by proposing two justification methods for theory lemmas: one batches justifications for parallelization, while the other not only checks the theory lemma justifications but also integrates them into a fully detailed proof that could be checked with standard approaches. Experimental results on SMT-LIB benchmarks show the benefits of our approach in reducing solving time when producing proof skeletons that can be effectively checked externally. In particular, the extended trimming techniques can significantly reduce the number of theory lemmas to be checked beyond standard trimming, thereby improving sequential and parallel checking times.

Andrew Reynolds

Engineering Complete SMT Proofs in cvc5 with Ethos/Eunoia

Abstract: Over the past 5 years, the SMT solver cvc5 has been instrumented to produce proofs for a majority of its theories. This talk reports on a new milestone for this work, namely that all mainstream features of cvc5 are 100% proof producing and checkable in an external proof checker (Ethos). In detail, Ethos is a high performance proof checker written in around 10k lines of C++. Its native input language is Eunoia, a logical framework for defining proof systems that is heavily inspired by the forthcoming SMT-LIB version 3.0 language. To engineer complete proofs

for cvc5, I will discuss the introduction of a "safe mode" of cvc5, which defines a subset of the features of cvc5 that are free of known bugs and have complete proof support. The internal proof calculus of cvc5, now known as the Cooperating Proof Calculus (CPC), has been formalized in around 6500 lines of Eunoia definitions. Notably, this formalization now covers all mainstream theories of cvc5, including those currently used by industrial users of cvc5.

Martina Seidl

Certified QBF Solving

Over the last years, much progress has been made in theory and practice of solving quantified Boolean formulas (QBFs). In principle, it is also well understood how to certify solving results found on solvers based on different solving paradigms like QCDCL as well as abstraction- and expansion-based solving. QBF certification is strongly inspired by approaches successfully used in SAT. Nevertheless, state-of-the-art solvers support certification to a limited extent only.
In this talk, an overview is given on the state of the art of certified QBF solving and the challenges that need to be addressed to obtain a fully operational certification workflow.

Monika Seisenberger

Certifying RUP proofs in the context of Railway Verification

We report on two recent advances we made in the context of applying SMT solving in the area of Railway Verification using Z3.
The first concerns a certified RUP checker that has been extracted using the Theorem Prover Rocq. We formalised the RUPchecker in Rocq, provided a soundness proof and extracted the checker from it. The procedure also allows to produce the corresponding Unitresolution proof, but to do so is not required for the correctness of the checked result.
The second application demonstrates the formalisation of graph theory to be applied to a setting in geographic data verification. The construction of such a custom theory is possible with the recent introduction of the user propagator interface in the Z3 solver.
This is joint work with Harry Bryant, Alec Critten, Andrew Lawrence, and Anton Setzer.

Geoff Sutcliffe

Proof Verification with GDV and LambdaPi - It's a Matter of Trust

Automated Theorem Proving (ATP) is concerned with the development and use of software that automates sound reasoning. An ATP system can be required to output a proof that serves as a certificate for the systems claim. To ensure that a proof is correct, verification can be required. If the verifier outputs evidence in a form that can be independently checked, that evidence serves as a certificate for the verifiers claim. The sequence of finding a proof, verifying the proof, and certifying the verification, builds an increasing level of trust in the system. This talk traces one such path for TPTP format proofs generated by ATP systems, via the GDV derivation verifier, and ending at the LambdaPi checker.

Stefan Szeider

Certifying Dynamic Symmetry Breaking in SAT and QBF

SAT Modulo Symmetries (SMS) performs dynamic symmetry breaking for graph generation by detecting and excluding non-canonical graphs during CDCL search. In this talk, I will focus on the certification mechanisms that ensure the correctness and completeness of this approach across different settings.
We will consider three types of certificates: (1) nc-certificates (non-canonicity certificates), which are permutations proving that a partially defined graph cannot be extended to a lexicographically minimal solution; (2) DRAT proofs where the symmetry-breaking clauses learned via the minimality check are added as axioms to the proof, with these axioms themselves certified by their corresponding nc-certificates; and (3) uniform proofs for the QBF setting, where

SMS handles quantified graph search problems with formal verification through LDQ-resolution.
These certification mechanisms provide mathematical guarantees for the correctness of SMS and enable independent verification of results in challenging combinatorial problems, from confirming the Murty-Simon conjecture to computing Ramsey graphs with formal proofs.
Joint work with Mikolas Janota, Markus Kirchweger, and Tomas Peitl.

Christoph Weidenbach

Certification in SCL

Clause Learning vom Simple models (SCL) is an approach to automated reasoning that generates only non-redundant clauses. For propositional logic it is very similar to CDCL. For first-order logic (FOL, with or without theories) it deduces non-redundant clauses from a ground (partial) model assumption built by propagation and decisions. Propagation in FOL is infinite and even for logics having the final model property worst case exponential. So straight forward propagation based certification does not work. On the other hand learned clauses can be the result of many resolution steps, similar to CDCL. So resolution based certification does not work either.
