DOI:
10.4204/EPTCS.311
ISSN: 2075-2180

EPTCS 311

Proceedings of the Second International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements

Natal, Brazil, August 26, 2019

Edited by: Martin Suda and Sarah Winkler

A Report on ARCADE 2019

The second ARCADE workshop took place on August 26, 2019 in Natal, Brazil, co-located with the 27th International Conference on Automated Deduction (CADE-27).

ARCADE stands for Automated Reasoning: Challenges, Applications, Directions, Exemplary achievements, and this slogan captures the aim of the workshop series: to bring together key people from various sub-communities of automated reasoning—such as SAT/SMT, resolution, tableaux, theory-specific calculi, interactive theorem proving—to discuss the present, past, and future of the field. In this abstract we report on key data

related to submissions and participation, and summarize the presentation sessions and the subsequent discussions.

Introduction

The second ARCADE workshop was held as a satellite event of CADE-27 as a forum for discussions and for sharing ideas about current challenges, new application areas, and future directions of automated reasoning. In the spirit of the first ARCADE held in 2017 in Gothenburg, the workshop encouraged short non-technical position statements revolving around challenges, applications, directions, and exemplary achievements in automated reasoning. The corresponding extended abstracts were summarized in short presentations at the workshop, with plenty of time for discussion after each talk.

Out of twelve submitted abstracts, the program committee selected eleven for presentation at the event. With 33 registrations and even more attendees, ARCADE was one of the largest workshops at CADE. The event was divided into three sessions comprising five topic blocks. Each session consisted of two or (in a single case) one short talks, followed by a discussion of the presented topics, driven by questions posed by the authors. The accepted abstracts and proposed questions were published on the workshop's web page in advance of the workshop to allow attendees to consider these points beforehand.

In this report we first recapitulate the scope and aims of ARCADE and briefly recall the history of the event. Then we present a high-level summary of the talks and associated discussions, and, finally, we end this report with a conclusion.

Scope of ARCADE

ARCADE aims to bring together experts from various sub-communities of automated reasoning to debate the present, past, and future of the field. Consequently, the workshop focuses on lively discussion: rather than presenting concluded work, authors are invited to inspire discussion. The title of the workshop is indicative of the encouraged topics:

Challenges: What are the next grand challenges for research on automated reasoning? By this we refer to problems, solving which would imply a significant impact (e.g., shift of focus) on the CADE community and beyond.

Applications: Where is automated reasoning already successfully applied in real-world (industrial) scenarios and where could automated reasoning be relevant in the future?

Directions: Based on the grand challenges and requirements from real-world applications, what are the research directions the community should promote? What bridges between the different sub-communities of CADE need to be strengthened? What new communities should be included (if any)?

Exemplary Achievements: What are the landmark achievements of automated reasoning whose influence reached far beyond the CADE community itself? What can we learn from those successes when shaping our future research?

Despite the focus of ARCADE on debate, it is also encouraged to submit an extended version of the abstracts in post-proceedings. As citations of ARCADE 2017 contributions have shown, ARCADE papers do get referenced to motivate later work. This illustrates that ARCADE indeed has the potential to shape ideas and trigger future research, and its submissions are valuable reference points for the community.

History of the Event

ARCADE 2019 was the second installation of the workshop, following the first ARCADE in 2017 in Gothenburg. At the previous workshop, out of 19 submissions the program committee selected 14 for presentation and discussion. The proceedings of ARCADE 2017 were published as Volume 51 of the EPiC series on EasyChair.

Workshop in 2019

As mentioned above, each of the five sessions consisted of one or two short talks of roughly 15 minutes, followed by a discussion of the presented topics driven by questions posed by the authors. This format stimulated lively debates.

The talks in the first session broached the issue of semantic and syntactic inference restrictions for different domains. More precisely, Christoph Weidenbach asked the question to which extent these two paradigms may be unifiable. It was remarked that in contrast to SAT, syntactic restrictions are often essential for more expressive logics; on the other hand, semantic guidance may offer significant advantages, too. Hence a unified approach bears a great potential, but a combination is clearly a non-trivial endeavour.

John Hester proposed in his talk an approach for using axiom schemata to reason in first-order set theories which are not finitely axiomatizable, such as ZFC. He posed the questions whether reasoning in NBG is more effective than ZFC, and whether axiom schemata are actually required for proofs in practice.

The second session was devoted to theorem proving approaches which exploit machine learning techniques in some sense. Claudia Schon presented a novel approach to common sense reasoning in the CoRg system. In a preprocessing phase, knowledge graphs are used as background knowledge; the background knowledge together with target formulae is then fed into a theorem prover, and afterwards a machine learning component is used to predict the relevance of different models obtained. She asked the question whether knowledge graphs can compete with knowledge bases for common sense reasoning tasks, but also raised the more general issue of deduction vs. abduction in this setting.

Sarah Winkler posed questions about the right granularity of features in machine learning for term rewriting and theorem proving, relating to work in the software verification community where hand-crafted features describing program characteristics turned out to be highly useful. In particular, she asked the question of how structural features of theorem proving problems could look like.

Automation of higher-order reasoning was the topic of the third session, where Sophie Tourret reported on the achievements and challenges in the ambitious Matryoshka project. She raised the question to which extent and on which level should theorem proving components be formally verified. A lively discussion afterwards also emphasized the importance of the question what level of higher-order reasoning is actually required in practice.

The fourth session focused on automation of ethical reasoning. Christoph Benzmüller proposed the area of machine ethics as a new application area for automated reasoning. He presented the Explicit Normative Reasoning and Machine Ethics project, which aims to develop ethical and legal governance components for intelligent autonomous systems.

In the second talk of this session, Matthew Peveler proposed modelling of ethical principles as an application area for quantified modal logic. He emphasized that logics to model ethical theories exhibit different requirements than modal logics for other purposes, and made the point that also new approaches for proof automation are required.

The fifth session circled around the challenge of making automated reasoning tools more useful in practice. Martin Riener pointed out that automated first-order theorem provers are hardly used in tool chains in the way SAT and SMT solvers are. He posed the question why this is not the case, and how ATPs would need to be improved to make them more suitable for tool chains.

Next, Martin Suda filled in for Giles Reger in making the point that tailoring AR tools to competitions like CASC does not necessarily maximize their applicability in real-world scenarios and that other measures than just the total number of solved problems from some benchmark should be taken into account. A discussion about the suitability of TPTP as reference point to judge theorem provers followed.

Further new application areas were the topic of the last session. Pedro Quaresma presented challenges for automated reasoning tools within geometry software. In particular, he proposed computer-supported reasoning in geometry as a vehicle to expose students to automated reasoning in general.

Alessandro Gianola pointed out that the verification of data-aware processes offers interesting new problems to the AR community that have hardly been tackled yet. He explained how data-aware processes already constitute an application area for experimenting with automated reasoning techniques, but also stressed that they offer genuinely new research challenges for automated reasoning.

Conclusion

In summary, ARCADE 2019 enriched CADE-27 by providing a venue for lively debate on challenges and new directions in current automated reasoning research. Despite the location being hard to reach for many researchers, attendance was considerable. Moreover, the presented topics turned out to be mostly different from those of ARCADE 2017, which suggests that there is also significant potential for future editions of this workshop.

3rd of December, 2019

Martin Suda
Sarah Winkler

Stronger Higher-Order Automation: A Report on the Ongoing Matryoshka Project

Jasmin Blanchette (*Vrije Universiteit Amsterdam, Amsterdam, The Netherlands*)
Pascal Fontaine (*Institut Montefiore, Université de Liège, Liège, Belgium*)
Stephan Schulz (*DHBW Stuttgart, Stuttgart, Germany*)
Sophie Tourret (*Max-Planck-Institut für Informatik, Saarbrücken, Germany*)
Uwe Waldmann
(*Max-Planck-Institut für Informatik, Saarbrücken, Germany*)

This extended abstract presents the contributions to automated reasoning made in the context of the project Matryoshka, funded for five years by the European Research Council.

Interactive theorem proving (ITP) is concerned with using proof assistants to write computer-checked proofs of theorems, generally expressed in a variant of higher-order logic (HOL). Proof assistants are expressive and versatile tools, and their success stories range from software and hardware verification in [23] to pure mathematics [22]. Despite the higher trustworthiness of a machine-checked proof compared with a hand-written one, only a very small fraction of today's software and mathematical proofs is verified in a proof assistant. One recurrent complaint about these tools is their lack of automation, which has let to the creation of "hammers" [16,26] that attempt to delegate the proof obligations to fully automatic theorem proving (ATP) tools.

Although some automatic theorem provers for HOL exist [14,33], they tend to underperform on problems generated from proof assistants [34]. Thus, hammers rely mostly on automatic provers for first-order logic (FOL), such as SMT (satisfiability modulo theories) solvers [11,17] and superposition-based theorem provers [27,32]. One unpleasant property of hammers is that they obfuscate the structure of terms and formulas through the application of a sequence of encodings.

This complicates the task of first-order automatic provers since it is this very structure that they exploit to find proofs efficiently.

To some extent, the ITP and ATP research communities have been working in parallel, largely ignoring each other. Hammers have contributed to bringing the two communities closer. Two years ago, a new division was created in the CASC competition, featuring benchmarks generated by Isabelle's Sledgehammer [35], reflecting the growing interest of the ATP community in ITP. Next year, IJCAR will also encompass the ITP conference.

The Matryoshka project, whose general aim is to bridge the gap between ATP and ITP by strengthening higher-order proof automation, is part of this trend. So is our colleagues' work on Leo-III [33] and Vampire [10]. Since Matryoshka's start almost two and a half years ago, progress has been achieved both in terms of improving ATP with an eye on ITP applications and of using ITP as a vehicle for ATP research, taking the form of several workshops and many publications. This year's edition of the ARCADE workshop is a good opportunity to provide to the community a comprehensive view of our contributions and to sketch general avenues for future work.

Automatic Proof at the Service of Interactive Proof

To face the challenges offered by HOL starting from FOL, we have chosen as milestones the intermediate logics *λ-free HOL* and *Boolean-free HOL*. In λ-free HOL, the only higher-order features allowed are functional variables and partial application of functions. In Boolean-free HOL, λ-expressions are additionally allowed. This fragment is encoded in λ-free HOL by relying on combinators, and λ-free HOL itself is encoded in FOL using the applicative encoding [15]. In these two intermediate logics, variables cannot be of Boolean type. This is the only difference between Boolean-free and full HOL. Both λ-free and Boolean-free HOL are interpreted using Henkin-style semantics [9].

To improve automation in proof assistants, our strategy is to develop calculi for these fragments that extend first-order calculi in a *graceful* way, meaning that the new calculi should be almost as efficient at first-order reasoning as their first-order counterparts while being additionally able to cope with higher-order reasoning steps in a reasonable manner. Our starting points are the superposition calculus and SMT's CDCL(T) calculus, the most successful first-order approaches for solving hammer benchmarks.

Toward Higher-Order Superposition

The calculus that Bentkamp, Blanchette, Cruanes, and Waldmann [8] designed for λ-free HOL resembles the first-order superposition calculus. A key ingredient is a term order for λ-free terms. We have developed two such orders that generalize the familiar lexicographic path order (Blanchette, Waldmann, and Wand [13]) and the Knuth--Bendix order with argument coefficients (Becker, Blanchette, Waldmann, and Wand [6]). Both orders are compatible with function contexts, just as their first-order counterparts. In the higher-order case, however, one would also like to have compatibility with arguments (that is, $s \succ s'$ implies $s\ t \succ s'\ t$), but this is hard to obtain. We compensate for this deficiency by restricting the superposition rule to apply at argument subterms (e.g., a or b in $\mathsf{f\ a\ b}$, but not f or $\mathsf{f\ a}$). Other superpositions are made unnecessary by an additional inference rule that adds arguments to partially applied functions.

Adding support for λ-terms imposes major modifications on the calculus (Bentkamp, Blanchette, Tourret, Vukmirović, and Waldmann [7]). Unification is no longer unitary, but infinitary, which means that inference rules generate streams of conclusions for a fixed set of premises. Handling these in practice requires dovetailing. Even worse, the structure of some kinds of variable-headed terms and λ-expressions (called *fluid* terms) can be fundamentally altered under substitution, which breaks the connection between superpositions on clauses and on their ground instances and makes the calculus incomplete. We restore completeness with a new inference rule in which fluid contexts are encoded by a fresh higher-order variable. To reach full HOL, this calculus must be further extended to handle Boolean variables, which can be replaced by arbitrary predicates under substitution, thus breaking the clausal structure of the main formula. Boolean encodings [15] and lazy clausification [21] are promising options to face this final challenge.

A prototype implementation of the λ-free calculus, based on the superposition prover Zipperposition, was implemented by Bentkamp [8]. This prover was then extended to Boolean-free HOL by Bentkamp, Vukmirović, and Tourret with promising results. Following on the same path, Vukmirović, with some guidance from Blanchette, Cruanes, and Schulz, also extended the high-performance E prover to λ-free HOL [33]. Partly by chance, the data structures in E were quite easy to adapt. Major changes include the replacement of the simple sort system by a proper type system. λ-free HOL terms are represented using E's existing term data structures. Types are represented as recursive structures and are maximally shared in a type bank inspired by E's shared term architecture. Updated unification and matching algorithms are able to return partial matches and unifiers. All of E's indexing data structures (perfect discrimination trees [24], fingerprint indices [30], and feature vector indices [31]) have successfully been adapted to λ-free HOL. As a result, the extended E runs at essentially the same speed as the first-order E. The next planned step is the extension of E to Boolean-free HOL.

Toward Higher-Order SMT

Together with Barbosa, Barrett, Reynolds, and Tinelli from the CVC4 team, the Matryoshka team, and more specifically El Ouraoui, Fontaine, and Tourret are experimenting with different approaches for extending SMT to HOL. The first, pragmatic approach, targets existing state-of-the-art SMT solvers with large code bases and complex data structures optimized for the first-order case. In this approach, the SMT solver is extended with only minimal modifications to its core data structures and algorithms. The second approach, requiring substantial modifications, entails the redesign of the essential data structures and algorithms of the solver to work directly in λ-free HOL. The approaches have been respectively implemented in the CVC4 and veriT solvers [5].

The pragmatic approach is based on the applicative encoding of HOL [15]. A preprocessing phase eliminates λ-expressions using λ-lifting: Each λ-expression is replaced by a fresh function symbol, and a quantified formula is introduced to define the symbol in terms of the original expression. Then, the applicative encoding is used lazily to cope with partial applications on the ground level. Trigger-based quantifier instantiation is adapted to handle encoded HOL terms properly. A trick that proves useful in practice is to add axioms that help the instantiation technique to go slightly beyond pure λ-free HOL. More precisely, the ``store'' axiom (stating, for any unary function `f`, the existence of a function that coincides with `f` except for one domain value) allows CVC4 to prove quite a few more formulas from the TPTP library when chosen instances are added [5]. In future work, the other instantiation methods, model-based, enumeration-based, and conflict-based, will be similarly adapted.

In the redesign approach, a suboptimal but simple classical congruence closure algorithm is extended to handle ground HOL terms. This algorithm handles partial applications natively, without relying on the applicative encoding. It is, however, necessary to add axioms to support extensionality. The trigger-based quantifier instantiation method is also adapted to cope with λ-free HOL terms. This involves the extension of the indexing techniques and of the matching algorithms. Store axioms are currently not used in the redesign approach, and we believe this partly explains (together with weaker instantiation techniques in general) the gap of efficiency between veriT (implementing the redesign approach) and CVC4 (implementing the pragmatic approach). Our ambition for the near future is to adapt the theory of the Congruence Closure with Free Variables algorithm [4] to λ-free HOL. An implementation of this algorithm would directly allow us to lift conflict-based instantiation to this logic. Redesigning enumerative instantiation techniques is also on our agenda.

Better SMT Proofs for Smooth Reconstruction

An important aspect of automated reasoning at the service of interactive proof is that, beyond checking the validity (or unsatisfiability) of a formula, the solver should provide some evidence for the result. We advocate proofs for SMT solvers. In particular, the veriT SMT solver generates fairly comprehensive proofs, and we are working on the proof production capabilities of the solver to ease the burden on tools that would want to replay proofs. Barbosa, Blanchette, and Fontaine [3] recently developed a method to output explicit proofs for many of the preprocessing steps that usually happen inside SMT solvers. These results have been further polished, along with the proof format, mostly by Fleury and Schurr [2,20]. In Isabelle, they achieve a reconstruction success rate of about 99%. Our goal is to have a reconstruction success rate of 100%, with a solver that behaves gracefully, i.e., that behaves as much as possible in the same way with proof production enabled as when it is disabled.

Interactive Proof at the Service of Automatic Proof

The formal verification of ATP leads to a better understanding of its core mechanisms and strengthen its theoretical results. It also facilitates incremental research by allowing researchers to experiment with existing formalisms and easily identify the consequences of their modifications. Sometimes the outcome of verification is a formal framework that can be instantiated by future applications.

A Verified SAT Solver

Fleury, with some help from Lammich and supervision from Blanchette and Weidenbach, formalized the conflict-driven clause learning (CDCL) calculus implemented in most modern SAT solvers [12]. Compared with other SAT solver verification works, this work emphasizes the stepwise refinement methodology and the connection between calculi variants described in the literature. It also considers clause forgetting, solver restarts, and incremental solving, which had not been the focus of formalization before.

Based on the CDCL formalization, Fleury implemented an optimized verified SAT solver called IsaSAT [18,19]. IsaSAT implements the two-watched-literal scheme and other imperative data structures. From a benchmark suite consisting of the preprocessed SAT Competitions problems from 2009 to 2017, IsaSAT solves 801 problems, compared with 368 for `versat` (the second fastest verified SAT solver) and 1388 for MiniSAT (the baseline reference for SAT solving) [18].

A Verified Ordered Resolution Prover

Schlichtkrull, Blanchette, Traytel, and Waldmann [29] formalized in the Isabelle/HOL proof assistant [25] a first-order prover based on ordered resolution with literal selection. They followed Bachmair and Ganzinger's account [1] from Chapter 2 of the *Handbook of Automated Reasoning*. The formal development covers the refutational completeness of two resolution calculi for ground clauses and general infrastructure for theorem proving processes and redundancy, culminating with a completeness proof for a first-order prover, called RP, expressed as transition rules operating on triples of clause sets. This material corresponds to the first four sections of Chapter 2. Interestingly, Bachmair and Ganzinger's main completeness result does not hold as stated, due to the improper treatment of inferences involving several copies of the same premise. The formalization uncovered numerous smaller mistakes.

In subsequent work [28], Schlichtkrull, Blanchette, and Traytel specified an executable prover that implements a fixed clause selection strategy and functional data structures, embodying the abstract prover described by Bachmair and Ganzinger. The executable prover is connected to the abstract prover through a chain of refinement steps.

A Framework for Saturation Provers (Ongoing Work)

One of the indispensable operations of realistic saturation theorem provers is deletion of subsumed formulas. Unfortunately, the well-known equivalence of dynamic and static refutational completeness holds only for derivations where all deleted formulas are *redundant*, and the usual definition of redundancy does not cover subsumed formulas. The fact that the equivalence of dynamic and static refutational completeness cannot be exploited directly is one of the main reasons why Bachmair and Ganzinger's refutational completeness proof for the RP prover is rather complicated and nonmodular.

Waldmann, Tourret, Robillard, and Blanchette are currently 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 arbitrary redundancy criteria, which in the end permits not only to cover subsumption deletion, but to model prover architectures in such a way that the static refutational completeness of the calculus immediately implies the dynamic refutational completeness of, say, an Otter loop or Discount loop prover implementing the calculus.

Acknowledgment

We thank Daniel El Ouraoui, Mathias Fleury, Hans-Jörg Schurr, and the anonymous reviewers for suggesting textual improvements. We also thank the ARCADE organizers, reviewers, and attendees for the discussions on this topic. The project receives funding from the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka).

References

1. Leo Bachmair & Harald Ganzinger (2001): *Resolution Theorem Proving*. In: Alan Robinson & Andrei Voronkov: Handbook of Automated Reasoning I. Elsevier and MIT Press, pp. 19–99, doi:10.1016/b978-044450813-3/50004-7.
2. Haniel Barbosa, Jasmin Christian Blanchette, Mathias Fleury, Pascal Fontaine & Hans-Jörg Schurr (2019): *Better SMT Proofs for Easier Reconstruction*. In: AITP 2019—Fourth Conference on Artificial Intelligence and Theorem Proving—Abstracts of the Talks—April 7-12, 2019, Obergurgl, Austria.
3. Haniel Barbosa, Jasmin Christian Blanchette & Pascal Fontaine (2017): *Scalable Fine-Grained Proofs for Formula Processing*. In: Automated Deduction—CADE 26—26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6–11, 2017, Proceedings, LNCS 10395. Springer, pp. 398–412, doi:10.1007/978-3-319-63046-5_25.
4. Haniel Barbosa, Pascal Fontaine & Andrew Reynolds (2017): *Congruence Closure with Free Variables*. In: Tools and Algorithms for the Construction and Analysis of Systems—23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings, Part II, LNCS 10206. Springer, pp. 214–230, doi:10.1007/978-3-662-54580-5_13.
5. Haniel Barbosa, Andrew Reynolds, Daniel El Ouraoui, Cesare Tinelli & Clark Barrett (2019): *Extending SMT Solvers to Higher-Order Logic*. In: Automated Deduction—CADE-27—27th International Conference on Automated Deduction, Natal, Brazil, August 23–30, 2019, Proceedings, LNCS 11716. Springer, pp. 35–54, doi:10.1007/978-3-030-29436-6_3.
6. Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann & Daniel Wand (2017): *A Transfinite Knuth–Bendix Order for Lambda-Free Higher-Order Terms*. In: Automated Deduction—CADE 26—26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6–11, 2017, Proceedings, LNCS 10395. Springer, pp. 432–453, doi:10.1007/978-3-319-63046-5_27.
7. Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirović & Uwe Waldmann (2019): *Superposition with Lambdas*. In: Automated Deduction—CADE-27—27th International Conference on Automated Deduction, Natal, Brazil, August 23–30, 2019, Proceedings, LNCS 11716. Springer, doi:10.1007/978-3-030-29436-6_4.
8. Alexander Bentkamp, Jasmin Christian Blanchette, Simon Cruanes & Uwe Waldmann (2018): *Superposition for Lambda-Free Higher-Order Logic*. In: 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, LNCS 10900. Springer, pp. 28–46, doi:10.1007/978-3-319-94205-6_3.
9. Christoph Benzmüller & Dale Miller (2014): *Automation of Higher-Order Logic*. In: Jörg H. Siekmann: Computational Logic, Handbook of the History of Logic 9. Elsevier, pp. 215–254,

doi:10.1016/B978-0-444-51624-4.50005-8.

10. Ahmed Bhayat & Giles Reger (2019): *Restricted Combinatory Unification*. In: Automated Deduction—CADE-27—27th International Conference on Automated Deduction, Natal, Brazil, August 23–30, 2019, Proceedings, LNCS 11716. Springer, pp. 74–93, doi:10.1007/978-3-030-29436-6_5.
11. Nikolaj Bjørner (2018): *Z3 and SMT in Industrial R&D*. In: Formal Methods—22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FLoC 2018, Oxford, UK, July 15–17, 2018, Proceedings, LNCS 10951. Springer, pp. 675–678, doi:10.1007/978-3-319-95582-7_44.
12. Jasmin Christian Blanchette, Mathias Fleury, Peter Lammich & Christoph Weidenbach (2018): *A Verified SAT Solver Framework with Learn, Forget, Restart, and Incrementality*. J. Autom. Reasoning 61(1–4), pp. 333–365, doi:10.1007/s10817-018-9455-7.
13. Jasmin Christian Blanchette, Uwe Waldmann & Daniel Wand (2017): *A Lambda-Free Higher-Order Recursive Path Order*. In: Foundations of Software Science and Computation Structures —20th International Conference, FoSSaCS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings, LNCS 10203. Springer, pp. 461–479, doi:10.1007/978-3-662-54458-7_27.
14. Chad E. Brown (2012): *Satallax: An Automatic Higher-Order Prover*. In: Automated Reasoning—6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26–29, 2012. Proceedings, LNCS 7364. Springer, pp. 111–117, doi:10.1007/978-3-642-31365-3_11.
15. Haskell Curry, Robert Feys & William Craig (1958): *Combinatory Logic* I. North-Holland, doi:10.1016/s0049-237x(08)x7103-4.
16. Łukasz Czajka & Cezary Kaliszyk (2018): *Hammer for Coq: Automation for Dependent Type Theory*. J. Autom. Reasoning 61(1–4), pp. 423–453, doi:10.1007/s10817-018-9458-4.
17. Morgan Deters, Andrew Reynolds, Tim King, Clark W. Barrett & Cesare Tinelli (2014): *A tour of CVC4: How it works, and how to use it*. In: Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21–24, 2014. IEEE, pp. 7, doi:10.1109/FMCAD.2014.6987586.
18. Mathias Fleury (2019): *Optimizing a Verified SAT Solver*. In: NASA Formal Methods—11th International Symposium, NFM 2019, Houston, TX, USA, May 7–9, 2019, Proceedings, LNCS 11460. Springer, pp. 148–165, doi:10.1007/978-3-030-20652-9_10.
19. Mathias Fleury, Jasmin Christian Blanchette & Peter Lammich (2018): *A Verified SAT Solver with Watched Literals using Imperative HOL*. In: Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8–9, 2018, pp. 158–171, doi:10.1145/3167080.
20. Mathias Fleury & Hans-Jörg Schurr (2019): *Reconstructing veriT proofs in Isabelle/HOL*. In: Proceedings Sixth Workshop on Proof eXchange for Theorem Proving, PxTP 2019, Natal, Brazil, August 26, 2019, EPTCS 301, pp. 36–50, doi:10.4204/EPTCS.301.6.
21. Harald Ganzinger & Jürgen Stuber (2005): *Superposition with Equivalence Reasoning and Delayed Clause Normal Form Transformation*. Inf. Comput. 199(1–2), pp. 3–23, doi:10.1016/j.ic.2004.10.010.
22. Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Rute Jason, Solovyev Alexey, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu & Roland Zumkeller (2017): *A Formal Proof Of The Kepler Conjecture*. Forum of Mathematics, Pi 5, pp. E2, doi:10.1017/fmp.2017.1.
23. Xavier Leroy (2009): *Formal Verification of a Realistic Compiler*. Commun. ACM 52(7), pp. 107–115, doi:10.1145/1538788.1538814.
24. William McCune (1992): *Experiments with Discrimination-Tree Indexing and Path Indexing for Term Retrieval*. J. Autom. Reasoning 9(2), pp. 147–167, doi:10.1007/BF00245458.
25. Tobias Nipkow, Lawrence C. Paulson & Markus Wenzel (2002): *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. LNCS 2283. Springer, doi:10.1007/3-540-45949-9.
26. Lawrence C. Paulson & Jasmin Christian Blanchette (2012): *Three Years of Experience with Sledgehammer, a Practical Link Between Automatic and Interactive Theorem Provers*. In: 8th International Workshop on the Implementation of Logics, IWIL 2010, Yogyakarta, Indonesia, October 9, 2011, EPiC Series in Computing 2. EasyChair, pp. 1–11, doi:10.29007/36dt.

27. Giles Reger & Martin Suda (2018): *Incremental Solving with Vampire*. In: Vampire 2017—Proceedings of the 4th Vampire Workshop, EPiC Series in Computing 53. EasyChair, pp. 52–63, doi:10.29007/6sj.
28. Anders Schlichtkrull, Jasmin Christian Blanchette & Dmitriy Traytel (2019): *A Verified Prover Based on Ordered Resolution*. In: Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14–15, 2019, pp. 152–165, doi:10.1145/3293880.3294100.
29. Anders Schlichtkrull, Jasmin Christian Blanchette, Dmitriy Traytel & Uwe Waldmann (2018): *Formalizing Bachmair and Ganzinger's Ordered Resolution Prover*. In: 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, pp. 89–107, doi:10.1007/978-3-319-94205-6_7.
30. Stephan Schulz (2012): *Fingerprint Indexing for Paramodulation and Rewriting*. In: Automated Reasoning—6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings, LNCS 7364. Springer, pp. 477–483, doi:10.1007/978-3-642-31365-3_37.
31. Stephan Schulz (2013): *Simple and Efficient Clause Subsumption with Feature Vector Indexing*. In: Maria Paola Bonacina & Mark E. Stickel: Automated Reasoning and Mathematics: Essays in Memory of William W. McCune, LNCS 7788. Springer, pp. 45–67, doi:10.1007/978-3-642-36675-8_3.
32. Stephan Schulz (2013): *System Description: E 1.8*. In: Logic for Programming, Artificial Intelligence, and Reasoning—19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14–19, 2013. Proceedings, LNCS 8312. Springer, pp. 735–743, doi:10.1007/978-3-642-45221-5_49.
33. Alexander Steen & Christoph Benzmüller (2018): *The Higher-Order Prover Leo-III*. In: 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, LNCS 10900. Springer, pp. 108–116, doi:10.1007/978-3-319-94205-6_8.
34. Nik Sultana, Jasmin Christian Blanchette & Lawrence C. Paulson (2013): *LEO-II and Satallax on the Sledgehammer Test Bench*. J. Applied Logic 11(1), pp. 91–102, doi:10.1016/j.jal.2012.12.002.
35. Geoff Sutcliffe (2017): *The CADE-26 automated theorem proving system competition—CASC-26*. AI Commun. 30(6), pp. 419–432, doi:10.3233/AIC-170744.
36. Petar Vukmirović, Jasmin Christian Blanchette, Simon Cruanes & Stephan Schulz (2019): *Extending a Brainiac Prover to Lambda-Free Higher-Order Logic*. In: Tools and Algorithms for the Construction and Analysis of Systems—25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part I, LNCS 11427. Springer, pp. 192–210, doi:10.1007/978-3-030-17462-0_11.

How Can We Improve Theorem Provers for Tool Chains?

Martin Riener (*The University of Manchester*)

Automated first-order theorem provers have matured as standalone tools to the point that they can be used within a larger infrastructure like Isabelle's Sledgehammer. Nevertheless, there is a significant difference to the spread of SAT solvers, that occur in simple applications like configuration management but are reliably used in tight loops of larger tool chains, not the least in SMT Solvers or instantiation/AVATAR based ATPs. We cannot expect a similar level of integration due to the higher expressiveness of general purpose theorem proving. Nonetheless, here we will identify some aspects that could improve the acceptance in industry.

Automated theorem provers have seen use as back-ends in larger software packages (various hammers, TLAPS) but are rarely used within a tool chain. For decidable theories, SMT solvers certainly have better properties but in this context, we focus mostly on their use as general purpose theorem provers (in other words, any SMT-LIB logic that includes uninterpreted functions and quantifier support). The abil-ity to produce models still distinguishes SMT solvers but the support of full first order logic confronts them with similar problems as other ATPs. Many of these problems are of a technical nature: the integration of a theorem prover into an industrial project brings several external requirements into play that are easily dealt with in interactive modes. The software might need to run in a certain operating system or avoid optional components under the GPL license. The availability of certain prover features also often depends on such optional components. For example, an arbitrary precision library might only provide integers where its differently licensedalternative also provides unbounded real numbers. In interactive use, we can often restate the problem to avoid real numbers altogether. A human can also investigate the reasons for time outs easier. We summarize these problems under the titles common availability, standardized interfaces, andreliability. The analysis is centred around CVC4 [1], E Prover [7], iProver [3], SPASS [8], Vampire [4], veriT [2], and Z3 [5].

Availability

Most commonly, provers target Linux as the main operating system and are available in source form to be compiled manually. Static Linux binaries are also often available for download at the project homepage, with the exception of Vampire. SPASS and E are packaged in single Linux distributions (Debian/Ubuntu and Fedora respectively) whereas CVC4 and Z3 are widely available. An exception areAlt-Ergo, Zipperposition and Beagle which are usually installed via Opam and sbt, the respective sourcebased package systems for OCaml and Scala.

Apart from Z3, CVC4 and veriT, which offer native binaries, the support for Windows is usually provided via the Cygwin compatibility layer. MacOS is usually supported via the source based package manager Homebrew. Due to Apple's strong discouragement of static linking, this is normally the only way of distribution.

From an industrial perspective, external factors often determine the operating system or way of distribution required. When multiple provers are required, the most common denominator is often only the compilation from source. The introduction of automated builds and testing environments for multiple operating systems is a significant task that is hard to publish on. Although rare, there are research engineer grants that could be used to hire an expert that does this maintenance. Since most provers are underan open source license, one time engineering tasks like the adaptation to a yet unsupported operating system could be announced as a Google Code project.

Interfacing

The most common way of communication with a theorem prover is via plain text. This has the advantages that the input language remains stable and the prover can be easily exchanged for an alternative. The two main input formats, SMT-LIB and TPTP, have been converging with regard to the supported logics and theories: for example, they both define integer and real arithmetic and higher-order logic. There are differences though: only SMT-LIB defines special theories like bit-vectors, data-types or arrays, even though some provers accept them with a custom syntax whereas only TPTP supports polymorphictypes and modal operators. The choice of the language therefore restricts which features are efficiently supported.

A further restriction is the fact that no prover supports all features of either SMT-LIB or TPTP. This ties the tool chain even tighter to a particular prover, making it hard to use different encodings of the same problem. On the other hand, the close integration can not be used to share data-structures or avoid repeated parsing of input.

Another distinction regards the output formats. SMT-LIB does not specify a proof format but it has a definition of models in terms of a Herbrand universe extended with abstract values. These abstract values are solely defined by the solver and again tie the tool chain tightly to a particular prover.

TPTP's proof format TSTP on the other hand specifies the proof DAG with the single semantic restriction that a conclusion logically follows from its premises. More detailed proofs can be constructed but require additional reasoning. If the proof replay is not tailored to a prover there is also a reasonable chance that it fails. TSTP only specifies a concrete format for finite models. In both cases, an industrial user needs to rely on prover specific behaviour that is often poorly documented.

Some of these issues can be mitigated with an intermediate layer such as pySMT. As an alternative, one can use an API, should it exist and when it has the correct language bindings. The API might change more frequently because it is tied more closely to the implementation.

In each case, only the ATP community can improve the situation. The efforts of harmonizing the SMT-LIB format with TIP seem to have been successful, making the benchmarks collections of both formats available to the other community. Due to the larger syntactic differences, this is not easy to achieve between TPTP and SMT-LIB but it is certainly possible continue integrating the features of the other side. The veriT SMT solver provides an excellent proof format that could be integrated into SMT-LIB. A stand-alone tool that converts between the formats would also be of great help. This could be a stepping stone of making the whole benchmark libraries of both formats available to the other. Currently, there is an overlap due to manual conversion efforts (for instance, the AUFNIRA/nasa is an adaption of the original statements using the array sort instead of an axiomatization). An full problem library that can state the problem in the logic and language required would make such a manual conversion effort unnecessary.

Reliability

The most dreaded behaviour of a theorem prover which has not been mentioned so far is when it reports "unknown", either due to a timeout or when it rejects the problem as outside of its capabilities.[1]

This can be due to an unfavourable encoding like expressing equality as a reflexive, symmetric and transitive relation. Other cases are not that obvious: arrays with Boolean values can replace a bit-vector but it depends on the decision procedures that are actually implemented which determine if the encoding is suitable to a problem. Decisions of the latter kind could be integrated into a framework like pySMT (also to compensate for missing theories) but in general this can only be solved by the creator of the tool chain.

Another way to obtain "unknown" is by enabling incomplete proof search strategies, be it particular instantiation techniques or axiom selection. In general, the number of options for theorem provers is immense and their interaction is hard to predict. Unfortunately, this situation is hard to improve. ATP developers can evaluate the effectiveness of a large number of combinations to find those that work well or eliminate some those that don't have any effect. How well this works depends strongly on how representative the benchmarks are for the application. Specialized tool chain developers might be able tune the options themselves but this can not expected from the average user.

A third possibility for a timeout is lack of knowledge about the (possible) model. A superposition based prover will not saturate the clause set generated by $\forall x : Int, y : Int.\, x < y$ and continue until it times out. Still, some learned similarity measures could select sufficiently distinct generated clauses that could be of interest for choosing a better suited set of options. A similar phenomenon occurs when SMT solvers run in a refinement loop. The clause above almost immediately produces a model $x = 0, y = 1$. A refined model would exclude these model values, obtaining a new model $x = -1, y = 2$ and so on. A human quickly recognizes that the problem has both infinitely many models and counter-models. Part of this information is also generated when the decision procedure -- here probably the Simplex algorithm -- is run and we should know more about the model theory of that decision procedure. How this knowledge can be preserved in the face of theory combination is not obvious. When we consider the clause $f(x) < f(y)$, whatever model we pick for f overshadows any monotonicity intuitions we had for the simpler case. Nevertheless, it could be beneficial to extract more information from a stopped proof search and feed this back into the refinement loop.

Summary

We have focused on topics that could be considered as obstacles for industrial integration: possible runtime environment restrictions, the need to ship with predetermined parameter schedules or the general hardships of undecidable problems. Nevertheless, it is worth highlighting the impressive advancements that have already been made in that regard. Modern theorem provers come with an extensive manual that includes their inference rules. The level of detail in proofs has become significantly more detailed. Finally, there have already been successful integrations of general first-order provers in widely used software, ranging from backends for interactive proof assistants to software synthesis and test case generation. However we hope that future development can close the gap between the research and its application in industry.

[1] For example, both CVC4 and Z3 report unknown when trying to find a model of $\exists a : Array(Int, Int), b : Array(Int, Int). a \neq b$ but they can both construct a model for the inequality of constants $c \neq d$, which is how existential quantification is defined.

References

1. Clark W. Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds & Cesare Tinelli (2011): *CVC4*. In: Ganesh Gopalakrishnan & Shaz Qadeer: Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings, Lecture Notes in Computer Science 6806. Springer, pp. 171–177, doi:10.1007/978-3-642-22110-1_14.
2. Thomas Bouton, Diego Caminha Barbosa De Oliveira, David Déharbe & Pascal Fontaine (2009): *veriT: An Open, Trustable and Efficient SMT-Solver*. In: Schmidt, pp. 151–156, doi:10.1007/978-3-642-02959-2_12.
3. Konstantin Korovin (2008): *iProver - An Instantiation-Based Theorem Prover for First-Order Logic (System Description)*. In: Alessandro Armando, Peter Baumgartner & Gilles Dowek: Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings, Lecture Notes in Computer Science 5195. Springer, pp. 292–298, doi:10.1007/978-3-540-71070-7_24.
4. Laura Kovács & Andrei Voronkov (2013): *First-Order Theorem Proving and Vampire*. In: Natasha Sharygina & Helmut Veith: Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings, Lecture Notes in Computer Science 8044. Springer, pp. 1–35, doi:10.1007/978-3-642-39799-8_1.
5. Leonardo Mendonça de Moura & Nikolaj Bjørner (2008): *Z3: An Efficient SMT Solver*. In: C. R. Ramakrishnan & Jakob Rehof: Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings, Lecture Notes in Computer Science 4963. Springer, pp. 337–340, doi:10.1007/978-3-540-78800-3_24.
6. Renate A. Schmidt (2009): *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*. Lecture Notes in Computer Science 5663. Springer, doi:10.1007/978-3-642-02959-2.
7. Stephan Schulz (2013): *System Description: E 1.8*. In: Ken McMillan, Aart Middeldorp & Andrei Voronkov: Proceedings of the 19th LPAR, Stellenbosch, LNCS 8312. Springer, doi:10.1007/978-3-642-45221-5_49.
8. Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda & Patrick Wischnewski (2009): *SPASS Version 3.5*. In: Schmidt, pp. 140–145, doi:10.1007/978-3-642-02959-2_10.

Intelligent Geometry Tools

James Davenport (*University of Bath*)
Jacques Fleuriot (*University of Edinburgh*)

Pedro Quaresma (*University of Coimbra*)
Tomás Recio (*University of Cantabria*)
Dongming Wang (*Beihang University*)

We propose building a community of "intelligent geometry" researchers, manifested by the creation of a living *Intelligent Geometry Book*, to introduce many more people to computer-supported reasoning.

On the one hand, participants at "Big Proof 2019" workshop commented that students (mostly mathematics, but also computing) were not being introduced to formal methods and computer-supported reasoning at an early enough stage. On the other hand, for hundreds if not thousands of years, geometry has been the tool of choice for introducing formal reasoning. Hence we propose that computer-supported reasoning in geometry is an ideal vehicle for exposing many more people to computer-supported reasoning than is currently the case.

The need for formal reasoning is not just a niche area in mathematics. The whole area of verified software relies on formal reasoning. Everyone who flies in or out of the UK has placed their lives in the hands of the air traffic controllers, supported by a computer system which has been formally verified, and the same is true of Line 14 (driverless) of the Paris Métro, or many railway systems. But the firms that develop such systems have an uphill struggle finding staff. They report that practically no recruit at the graduate level has been exposed to computer-supported reasoning.

Reuse of previous results is vital across science, but nowhere more than in mathematics. However, in practice re-use of *implementations of ideas* has proved much harder than re-use of *abstract ideas*. The same algorithm can be implemented many times. Some times this re-implementation is due to real improvements (as we may provide a shorter or clearer proof of a theorem), but often it is due to "engineering" mismatches: different programming languages, data formats etc.

Therefore this project aims to build a community of researchers, the *Intelligent Geometry Community*, experts in the representation and management of geometric knowledge, and knowledge-based intelligent software for exploratory and educational purposes. By pooling the experiences of the participants over the fundamental issues to be addressed by any project in this area and making results open source/open access, the network will lay the ground for advanced research projects contributing to the European research agenda. The creation of this community will be demonstrated by the creation of a living *Intelligent Geometry Book*, the *i*GeometryBook.

Hence the main challenge is to organise and harmonise the work of the participants and to integrate different techniques and tools developed by them, to facilitate collaborations within the network.

The *i*GeometryBook will be an intelligent environment, collaborative, adaptive and adaptable, containing formally represented machine-checked geometric knowledge, historically analysed, with embedded tools for computation, deduction, and knowledge processing. It is envisaged that the knowledge contained in the *i*GeometryBook will be created by the authors, with contributions from readers, using the embedded tools according to certain pre-designed mechanisms and remotely accessible and searchable with natural and visual languages interfaces.

Such a "superset of a book", freely available in all computational platforms, adaptable, collaborative and adaptive to each and every user's profiles, would bring together a whole new generation of mathematical tools with impact at all levels: exploratory research, applications on mathematics and education.

The need of information and communications technology (ICT) in a learning setting is well recognised, as said in *Mathematics Education in Europe: Common Challenges and National Policies* the use of ICT for teaching mathematics is prescribed or recommended in all European countries. The recommendations range from very specific instructions to more general guidelines [1,2,3,4,5]. In particular, three of the aims in [2] are

1. construct mathematical proofs;

2. make deductions and inferences and draw conclusions by using mathematical reasoning;
3. use technology such as calculators and computers effectively, and recognise when such use may be inappropriate.

It is our thesis that an appropriate interactive geometry tool can support, not only each aim above separately, but also the interplay of these aims.

In terms of the challenges of ARCADE, this should also respond to "How can we attract young people to our field?".

References

1. Ministério da Educação e Ciência (2013): *Programa e Metas Curriculares Matemática A - Ensino Secundário*. Technical Report. Ministério da Educação e Ciência, República Portuguesa.
2. Department for Education (2014): *Further maths AS and A level subject content for teaching in schools from 2017*. Technical Report. GOV.uk.
3. M. Isoda. (2010): *Junior high school teaching guide for the japanese course of study: Mathematics (grade 7-9)*. Technical Report. CRICED, University of Tsukuba.
4. Ministère de l'éducation nationale, de la jeunesse et de la vie associative (2011): *Programme de l'enseignement spécifique et de spécialité de mathématiques*. Bulletin officiel spécial 8, MENJVA, 13 octobre, 2011.
5. Ministero dell'istruzione, dell'università e Della Ricerca (2010): *Indicazioni nazionali riguardanti gli obiettivi specifici di apprendimento per il liceo delle scienze umane*. Decreto 7 ottobre 2010, n. 211, gu serie generale n.291 del 14-12-2010 – suppl. ordinario n. 275., MIUR, 2010.

Verification of Data-Aware Processes: Challenges and Opportunities for Automated Reasoning

Diego Calvanese (*Free University of Bozen-Bolzano*)
Silvio Ghilardi (*Università degli Studi di Milano*)
Alessandro Gianola (*Free University of Bozen-Bolzano*)
Marco Montali (*Free University of Bozen-Bolzano*)
Andrey Rivkin (*Free University of Bozen-Bolzano*)

We briefly introduce the line of research on the verification of data-aware processes, with the intention of raising more awareness of it within the automated reasoning community. On the one hand, data-aware processes constitute a concrete setting for validating and experimenting with automated reasoning techniques. On the other hand, they trigger new genuine research challenges for researchers in automated reasoning.

Introduction

Contemporary organizations rely more and more on business processes to describe, analyze, and regulate their internal work. Business process management (BPM) is now a well-assessed discipline at the intersection between operations management, computer science, and IT engineering. Its grand goal is to support managers, analysts, and domain experts in the design, deployment, enactment, and continuous improvement of processes [21].

One of the essential concepts in BPM is that of a *process model*. A process model explicitly describes which tasks have to be performed within the organization (such as *check order*) in response to external events (such as *receive order request*), and what are the allowed courses of execution (such as *deliver order* can only be executed if check order has been successfully completed). Several process modeling languages have been proposed for this purpose, such as BPMN [35], UML Activity Diagrams [26], and EPCs [1]. Verification and automated reasoning techniques are in this respect instrumental to formally analyze process models and ascertain their correctness before their actual deployment into corresponding BPM systems.

Traditionally, formal analysis of process models is limited to the process control flow, represented using variants of bounded Petri nets or finite-state transition systems (depending on how concurrency is interpreted). This, however, does not reflect the intrinsic, multi-perspective nature of processes and their corresponding models. In particular, process tasks are executed by *resources* based on *decisions* that depend on background and process-related *data*, in turn manipulated upon task execution.

In this multi-perspective spectrum, the last two decades have seen a huge body of research dedicated to the integration of *data* and *process* management to achieve a more comprehensive understanding on how data influence behavior, and how behavior impact data [38, 20, 37].

The corresponding development of formal frameworks for the verification of data-aware processes has consequently flourished, leading to a wide plethora of formal models depending on how the data and process components, as well as their interplay, is actually represented.

One stream of research followed the artifact-centric paradigm, where the main focus is that of persistent business objects (such as orders or loans) and their lifecycle [41, 8]. Here, variants of the same model are obtained depending on how such business objects are represented. Notable examples are: *(i)* relational data with different kinds of constraints [18, 6,32], *(ii)* relational data with numerical values and arithmetics [15, 19], *(iii)* tree-structured data [7]. Also more minimalistic models have been brought forward, capturing data-aware processes as a persistent data storage evolved through the application of (conditional) actions that may inject external, possibly fresh values through service calls reminiscent of uninterpreted functions. Two variants of this model have been studied, the first considering persistent relational data with constraints [5, 2], the second operating over description logic knowledge bases whose extensional data are interpreted under incomplete information, and updated in the style of Levesque functional approach [28, 14].

Another stream of research followed instead the more traditional activity-centric approach, relying on Petri nets as the underlying control-flow backbone of the process. Specifically, Petri net-based models have been enriched with: *(i)* data items locally carried by tokens [39, 31], *(ii)* data registers with numerical and non-numerical values [16], *(iii)* tokens carrying tree-structured data [4], and/or*(iv)* persistent relational data manipulated with the full power of FOL/SQL [17, 34].

Last but not least, the interplay between data and processes has been studied to build solid foundations for "many-to-many" processes, that is, processes whose tasks co-evolve multiple different objects related to each other (such as e-commerce companies where each order may correspond to multiple shipped packages, and each package may contain items from different orders). Implicit (data-driven) [3] and explicit (token-driven) [22] coreference and synchronization mechanisms have been proposed for this purpose.

On top of these formal models, several verification tasks have been studied. On the one hand, they consider different types of properties, ranging from fundamental properties such as reachability, safety, soundness and liveness, to sophisticated formulae expressed in linear- and branching-time first-order temporal logics [8]. On the other hand, they place different assumptions regarding how data can be manipulated, and whether there are read-only data whose configuration is not known. The resulting verification problems are all undecidable in general, and require to properly tame the infinity arising from the presence of data.

All in all, we believe this wide spectrum of verification problems constitutes an extremely interesting application area for automated reasoning techniques. On the one hand, data-aware processes constitute a concrete setting for experimenting symbolic techniques developed within automated reasoning, so as to enable reasoning on the evolution of data without explicitly representing them. In addition, given the applied flavor of BPM, the feasibility of assumptions and conditions imposed towards guaranteeing good computational properties (such as decidability or tractability) can be assessed in the light of enduser-oriented modeling languages and their corresponding modeling methodologies. On the other hand, data-aware processes trigger new, genuine research challenges for researchers in automated reasoning, arising from the subtle, yet necessary interplay between control-flow aspects and volatile and persistent data with constraints.

To substantiate this claim, we briefly describe next one particular verification problem where automated reasoning techniques are very promising.

The Concrete Case of Relational Artifact Systems

Artifact systems formalize data-aware processes using three main components: *(i)* a read-only database that stores fixed, background information; *(ii)* a working memory that stores the evolving state of artifacts throughout their lifecycle; *(iii)* actions that inspect the read-only memory and the working memory, and consequently update the working memory. Different variants of this model, obtained via a careful tuning of the relative expressive power of its three components, have been studied towards decidability of verification problems parameterized over the read-only database (see, e.g., [18, 15, 7, 19, 11, 12, 9]). These are verification problems where a property is checked for every possible configuration of the read-only database, thus guaranteeing that the overall process operates correctly no matter how the read-only data are instantiated.

In the most recent variants of this model, the read-only database is equipped with key and foreign key constraints relating the content of different relations. At the same time, the working memory is relational, with each relation representing an artifact, in principle capable of storing unboundedly many tuples denoting instances of that artifact [19, 32].

In [11], we took inspiration from this approach, studying the model of so-called *relational artifact systems* (RASs). Notably, we connected RASs to the well-established model of array-based systems within the SMT tradition [24]. This is done in two steps. First, the schema of a read-only database is represented in a functional, algebraic fashion, where relations and constraints are captured using multiple sorts and unary functions. Second, each artifact relation within the working memory is treated as a set of arrays, where each array accounts for one component of the corresponding artifact relation. A tuple (i.e., artifact instance) in an artifact relation is then reconstructed by accessing all such arrays with the same index.

With these notions at hand, from a logical point of view the behavior of a RAS is specified via: *(i)* second order variables for artifacts components; *(ii)* first order variables for "data", ranging both on the sorts of the read-only database and on numerical (real, integer) domains. Thus, suitable combinations of (linear) arithmetics and EUF can be employed for reasoning about RAS systems. Non-determinism in system evolution is captured via first-order parameters, that is, further existentially quantified variables occurring in transition formulae, whereas second-order variables updates are functionally determined by such non-determinism at the first-order level.

On the top of this formal model, various problems arise that can be effectively attacked using techniques and solutions within the automated reasoning community in general, and the SMT community in particular. We briefly discuss next some of them.

1. By focusing on model checking of safety properties via symbolic backward reachability [24, 25], the main problem is that of avoiding the existential prefix to grow in an uncontrolled way. This, in turn, calls for some form of symbol elimination. This is rather easily achieved for second-order variables – at least when backward search is employed – because, as mentioned above, updates are often functional modulo first-order parameters; however, it is not clear what happens if alternative search strategies are employed, or other relevant properties are checked. At the first-order level, specific challenges instead arise already in this setting. In fact, while numerical existentially quantified variables can be eliminated via well-known methods (such as predicate abstraction [23], interpolation [33, 30], model elimination [29, 36], or even quantifier elimination), the manipulation of variables ranging over theread-only

database appears to require completely different techniques, like cover computation [27]. Thanks to cover computation, one can in particular overcome the fact that quantifier elimination is not directly applicable to variables pointing to elements of the read-only database. More technically, the computation of covers is nothing but quantifier elimination in the model completions of the theory used to capture the schema and the constraints of the read-only database schema, as shown in [12]. The idea of using model completions when quantifier elimination is not directly available is presentalso in [40]. Notably, differently from quantifier elimination in linear arithmetics, cover computationin the restricted "unary" case required for RASs turns out to be tractable [27, 12].

2. Different types of properties are usually required to be verified in the context of data-aware processes, where safety is one of the most typical. Nevertheless, a comprehensive research dedicated to SMT-based techniques for the effective verification of data-aware processes should also consider richer forms of verification going beyond safety (e.g., liveness and fairness), and richer classes of artifact systems incorporating concrete data types and arithmetic operations that should explicitly appear in the specification language.
3. Database instances are typically built on top of *finite* structures (although they may contain elementsfrom "value" sorts ranging over infinite domains). Depending on the specific setting under investigation, this feature may require to introduce specific techniques from finite model theory. In particular, advanced applications will presumably require: from the foundational perspective, to investigate suitable versions of the finite model property; from the applied perspective, to integrate common solvers with model finders.
4. Interesting variants of RASs arise when the data stored therein are interpreted under incomplete information, and in the presence of complex ontological constraints expressing background, structural knowledge of the organisational domain. Transferring model checking techniques such as those re-called in point 1 above to this richer setting is not at all trivial, as reasoning must now be carried outtackling two dimensions at once: the temporal dimension along which the artifact systems evolves, and the structural dimension constraining the manipulated data objects and their mutual relationships.
5. Towards enabling the concrete exploitation of verification techniques, logic-based formalisms used to formalize RASs or other types of data-aware processes need to be connected to end user-oriented process modeling languages. This interconnection paves the way towards practical reasoning tasks that are relevant for end users, but have not yet addressed by the automated reasoning community. In addition, by considering specific modeling guidelines and methodologies, interesting subclasses of general formal models such as that of RASs may naturally emerge. It would be then important to assess whether such subclasses come with interesting computational guarantees for the corresponding automated reasoning tasks.

We tried to address only some of the most simple problems from the above list. In particular, we have used RASs as a basis for formalizing a data-aware extension of the de-facto process modeling standard BPMN [10], and used the resulting approach to conduct an initial benchmark using some process models from [32], with very encouraging results [13, 11, 10].

To sum up, we believe that by employing both well-established and relatively new techniques, the automated reasoning community is ready to face the challenges raised by the emerging area of verification of data-aware processes, providing foundational, algorithmic, and applied advancements.

References

1. W.M.P. van der Aalst (1999): *Formalization and Verification of Event-driven Process Chains*. Information and Software Technology 41(10), pp. 639–650, doi:10.1016/S0950-5849(99)00016-6.
2. P. A. Abdulla, C. Aiswarya, M. F. Atig, M. Montali & O. Rezine (2016): *Recency-Bounded Verification of Dynamic Database-Driven Systems*. In: Proc. PODS. ACM Press, pp. 195–210, doi:10.1145/2902251.2902300.
3. A. Artale, A. Kovtunova, M. Montali & W. M. P. van der Aalst (2019): *Modeling and Reasoning over Declarative Data-Aware Processes with Object-Centric Behavioral Constraints*. In: Proc. BPM. Springer, pp. 139–156, doi:10.1007/978-3-030-26619-6_11.

4. E. Badouel, L. Hélouët & C. Morvan (2016): *Petri Nets with Structured Data*. Fundam. Inform. 146(1), pp. 35–82, doi:10.3233/FI-2016-1375.
5. B. Bagheri Hariri, D. Calvanese, G. De Giacomo, A. Deutsch & M. Montali (2013): *Verification of Relational Data-centric Dynamic Systems with External Services*. In: Proc. PODS, pp. 163–174, doi:10.1145/2463664.2465221.
6. F. Belardinelli, A. Lomuscio & F. Patrizi (2012): *An Abstraction Technique for the Verification of Artifact-Centric Systems*. In: Proc. of KR. Available at http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4531.
7. M. Bojańczyk, L. Segoufin & S. Toruńczyk (2013): *Verification of database-driven systems via amalgamation*. In: Proc. of PODS, pp. 63–74, doi:10.1145/2463664.2465228.
8. D. Calvanese, G. De Giacomo & M. Montali (2013): *Foundations of Data Aware Process Analysis: A Database Theory Perspective*. In: Proc. PODS, pp. 1–12, doi:10.1145/2463664.2467796.
9. D. Calvanese, S. Ghilardi, A. Gianola, M. Montali & A. Rivkin (2018): *Verification of Data-Aware Processes via Array-Based Systems (Extended Version)*. Technical Report arXiv:1806.11459. arxiv.org. Available at https://arxiv.org/abs/1806.11459.
10. D. Calvanese, S. Ghilardi, A. Gianola, M. Montali & A. Rivkin (2019): *Formal Modeling and SMT-Based Parameterized Verification of Data-Aware BPMN*. In: Proc. BPM. Springer, pp. 157–175, doi:10.1007/978-3-030-26619-6_12.
11. D. Calvanese, S. Ghilardi, A. Gianola, M. Montali & A. Rivkin (2019): *From Model Completeness to Verification of Data Aware Processes*. In: Description Logic, Theory Combination, and All That. Springer, pp. 212–239, doi:10.1007/978-3-030-22102-7_10.
12. D. Calvanese, S. Ghilardi, A. Gianola, M. Montali & A. Rivkin (2019): *Model Completeness, Covers and Superposition*. In: Proc. of CADE, pp. 142–160, doi:10.1007/978-3-030-29436-6_9.
13. D. Calvanese, S. Ghilardi, A. Gianola, M. Montali & A. Rivkin (To appear): *SMT-based Verification of Data-Aware Processes: a Model-Theoretic Approach*. Mathematical Structures in Computer Science.
14. D. Calvanese, M. Montali & A. Santoso (2015): *Verification of Generalized Inconsistency-Aware Knowledge and Action Bases*. In: Proc. IJCAI. AAAI Press, pp. 2847–2853. Available at http://ijcai.org/Abstract/15/403.
15. E. Damaggio, A. Deutsch & V. Vianu (2012): *Artifact Systems with Data Dependencies and Arithmetic*. ACM TODS 37(3), pp. 22:1–22:36, doi:10.1145/2338626.2338628.
16. M. de Leoni, P. Felli & M. Montali (2018): *A Holistic Approach for Soundness Verification of Decision-Aware Process Models*. In: Proc. ER, pp. 219–235, doi:10.1007/978-3-030-00847-5_17.
17. R. De Masellis, C. Di Francescomarino, C. Ghidini, M. Montali & S. Tessaris (2017): *Add Data into Business Process Verification: Bridging the Gap between Theory and Practice*. In: Proc. AAAI. AAAI Press, pp. 1091–1099. Available at http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14627.
18. A. Deutsch, R. Hull, F. Patrizi & V. Vianu (2009): *Automatic Verification of Data-Centric Business Processes*. In: Proc. of ICDT, pp. 252–267, doi:10.1145/1514894.1514924.
19. A. Deutsch, Y. Li & V. Vianu (2016): *Verification of Hierarchical Artifact Systems*. In: Proc. PODS. ACM Press, pp. 179–194, doi:10.1145/2902251.2902275.
20. M. Dumas (2011): *On the Convergence of Data and Process Engineering*. In: Proc. of ADBIS, pp. 19–26, doi:10.1007/978-3-642-23737-9_2.
21. M. Dumas, M. La Rosa, J. Mendling & H. A. Reijers (2013): *Fundamentals of Business Process Management*. Springer, doi:10.1007/978-3-642-33143-5.
22. D. Fahland (2019): *Describing Behavior of Processes with Many-to-Many Interactions*. In: Proc. of PETRI NETS, LNCS 11522. Springer, pp. 3–24, doi:10.1007/978-3-030-21571-2_1.
23. C. Flanagan & S. Qadeer (2002): *Predicate abstraction for software verification*. In: Proc. of POPL, pp. 191–202, doi:10.1145/503272.503291.
24. S. Ghilardi, E. Nicolini, S. Ranise & D. Zucchelli (2008): *Towards SMT Model Checking of Array-Based Systems*. In: Proc. of IJCAR, pp. 67–82, doi:10.1007/978-3-540-71070-7_6.
25. S. Ghilardi & S. Ranise (2010): *Backward Reachability of Array-based Systems by SMT Solving: Termination and Invariant Synthesis*. Logical Methods in Computer Science 6(4), doi:10.2168/LMCS-6(4:10)2010.

26. Object Management Group (2013): *OMG Unified Modeling Language 2.5*. Http://www.omg.com/uml/.
27. S. Gulwani & M. Musuvathi (2008): *Cover Algorithms and Their Combination*. In: Proc. of ESOP, Held as Part of ETAPS, pp. 193–207, doi:10.1007/978-3-540-78739-6_16.
28. B. Bagheri Hariri, D. Calvanese, G. De Giacomo, R. De Masellis, P. Felli & M. Montali (2012): *Verification of Description Logic Knowledge and Action Bases*. In: Proc. ECAI, pp. 103–108, doi:10.3233/978-1-61499-098-7-103.
29. K. Hoder & Nikolaj Bjørner (2012): *Generalized Property Directed Reachability*. In: Proc. of SAT, pp. 157–171, doi:10.1007/978-3-642-31612-8_13.
30. L. Kovács & A. Voronkov (2009): *Interpolation and Symbol Elimination*. In: Proc. of CADE, pp. 199–213, doi:10.1007/978-3-642-02959-2_17.
31. S. Lasota (2016): *Decidability Border for Petri Nets with Data: WQO Dichotomy Conjecture*. In: Proc. of PETRI NETS, LNCS 9698. Springer, pp. 20–36, doi:10.1007/978-3-319-39086-4_3.
32. Y. Li, A. Deutsch & V. Vianu (2017): *VERIFAS: A Practical Verifier for Artifact Systems*. PVLDB 11(3), pp. 283–296, doi:10.14778/3157794.3157798.
33. K.L. McMillan (2006): *Lazy Abstraction with Interpolants*. In: Proc. of CAV, pp. 123–136, doi:10.1007/11817963_14.
34. M. Montali & A. Rivkin (2017): *DB-Nets: on The Marriage of Colored Petri Nets and Relational Databases*. TOPNOC 12, pp. 91–118, doi:10.1007/978-3-662-55862-1_5.
35. OMG (2009): *Business Process Model and Notation (BPMN) - Version 2.0, Beta 1*.
36. O. Padon, N. Immerman, S. Shoham, A. Karbyshev & M. Sagiv (2016): *Decidability of inferring inductive invariants*. In: Proc. of POPL, pp. 217–231, doi:10.1145/2837614.2837640.
37. M. Reichert (2012): *Process and Data: Two Sides of the Same Coin?*. In: Proc. of the On the Move Confederated Int. Conf. (OTM 2012), LNCS 7565. Springer, doi:10.1007/978-3-642-33606-5_2.
38. C. Richardson (2010): *Warning: Don't Assume Your Business Processes Use Master Data*. In: Proc. of BPM, LNCS 6336. Springer, doi:10.1007/978-3-642-15618-2_3.
39. F. Rosa-Velardo & D. de Frutos-Escrig (2011): *Decidability and complexity of Petri nets with unordered data*. Theor. Comput. Sci. 412(34), pp. 4439–4451, doi:10.1016/j.tcs.2011.05.007.
40. V. Sofronie-Stokkermans (2016): *On Interpolation and Symbol Elimination in Theory Extensions.*. In: Proc. of IJCAR, Lecture Notes in Computer Science. Springer, pp. 273–289, doi:10.1007/978-3-319-40229-1_19.
41. V. Vianu (2009): *Automatic Verification of Database-Driven Systems: a New Frontier*. In: Proc. of ICDT, pp. 1–13, doi:10.1145/1514894.1514896.