Lecture Notes in Computer Science 14414

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

Haniel Barbosa · Yoni Zohar
Editors

Formal Methods: Foundations and Applications

26th Brazilian Symposium, SBMF 2023
Manaus, Brazil, December 4–8, 2023
Proceedings

Springer

Editors
Haniel Barbosa
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil

Yoni Zohar
Bar-Ilan University
Ramat Gan, Israel

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-49341-6 ISBN 978-3-031-49342-3 (eBook)
https://doi.org/10.1007/978-3-031-49342-3

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

This volume contains the papers presented at SBMF 2023: the 26th Brazilian Symposium on Formal Methods. After three consecutive virtual events due to the COVID-19 pandemic, we were happy to have SBMF again as an in-person event, held at Manaus, Brazil, from December 6 to December 8, 2023, with satellite events on December 4 and December 5, 2023.

The Brazilian Symposium on Formal Methods (SBMF) is an event devoted to the development, dissemination, and use of formal methods for the construction of high-quality computational systems, aiming to promote opportunities for researchers and practitioners with an interest in formal methods to discuss the recent advances in this area. SBMF is a consolidated scientific-technical event in the software area. Its first edition took place in 1998, and it reached the jubilee 25th edition in 2022. The proceedings of recent editions have been published mostly in Springer's Lecture Notes in Computer Science series as volumes 5902 (2009), 6527 (2010), 7021 (2011), 7498 (2012), 8195 (2013), 8941 (2014), 9526 (2015), 10090 (2016), 10623 (2017), 11254 (2018), 12475 (2020), 13130 (2021), and 13768 (2022).

The conference included four invited talks, given by Artur d'Avila Garcez (City, University of London, UK), Stéphane Graham-Lengrand (SRI International, USA), Chantal Keller (Université Paris-Saclay, France), and Vince Molnár (BME-FTSRG, Hungary). A total of 9 papers were presented at the conference and are included in this volume, with 7 of them as regular papers and 2 of them as short papers. They were selected from 16 submissions (12 regular, 4 short) that came from 7 different countries: Brazil, Spain, the UK, France, the USA, South Africa, and Argentina. The Program Committee comprised 36 members from the national and international community of formal methods. Each submission was reviewed by three Program Committee members (single-blind review). Submissions, reviews, deliberations, and decisions were handled via EasyChair, which provided good support throughout this process.

We are grateful to the Program Committee for their hard work in evaluating submissions and suggesting improvements. We are very thankful to the general chair of SBMF 2023, Edjard Mota (Universidade Federal do Amazonas, Brazil), who made everything possible for the conference to run smoothly. SBMF 2023 was organized by the Universidade Federal do Amazonas (UFAM), and promoted by the Brazilian Computer Society (SBC). We would further like to thank SBC for their sponsorship, and Springer for agreeing to publish the proceedings as a volume of Lecture Notes in Computer Science.

December 2023

Haniel Barbosa
Yoni Zohar

Organization

General Chair

Edjard Mota	Universidade Federal do Amazonas, Brazil

Program Committee Chairs

Haniel Barbosa	Universidade Federal de Minas Gerais, Brazil
Yoni Zohar	Bar-Ilan University, Israel

Steering Committee

Gustavo Carvalho	Universidade Federal de Pernambuco, Brazil
Volker Stolz	Western Norway University of Applied Sciences, Norway
Sérgio Campos	Universidade Federal de Minas Gerais, Brazil
Marius Minea	University of Massachusetts Amherst, USA
Vince Molnár	Budapest University of Technology and Economics, Hungary
Lucas Lima	Universidade Federal Rural de Pernambuco, Brazil

Program Committee

Yoni Zohar	Bar-Ilan University, Israel
Haniel Barbosa	Universidade Federal de Minas Gerais, Brazil
Katalin Fazekas	TU Wien, Austria
Mathias Preiner	Stanford University, USA
Daniela Kaufmann	TU Wien, Austria
Edjard Mota	Universidade Federal do Amazonas, Brazil
Maurice ter Beek	ISTI-CNR, Italy
Vince Molnár	Budapest University of Technology and Economics, Hungary
Mathias Fleury	University of Freiburg, Germany
Leila Ribeiro	Universidade Federal do Rio Grande do Sul, Brazil
Luís Soares Barbosa	University of Minho, Portugal
Volker Stolz	Høgskulen på Vestlandet, Norway
Nils Timm	University of Pretoria, South Africa
Thierry Lecomte	ClearSy System Engineering, France
Lucas Lima	Universidade Federal Rural de Pernambuco, Brazil
Marcel Oliveira	Universidade Federal do Rio Grande do Norte, Brazil
Gustavo Carvalho	Universidade Federal de Pernambuco, Brazil
Márcio Cornélio	Universidade Federal de Pernambuco, Brazil

Clark Barrett	Stanford University, USA
Juliano Iyoda	Universidade Federal de Pernambuco, Brazil
Sergio Campos	Universidade Federal de Minas Gerais, Brazil
Adenilso Simao	University of São Paulo, Brazil
Ahmed Irfan	SRI International, USA
Leopoldo Teixeira	Universidade Federal de Pernambuco, Brazil
David Déharbe	ClearSy System Engineering, France
Michael Leuschel	University of Düsseldorf, Germany
Giselle Reis	Carnegie Mellon University-Qatar, Qatar
Rohit Gheyi	Universidade Federal de Campina Grande, Brazil
Augusto Sampaio	Universidade Federal de Pernambuco, Brazil
Armin Biere	University of Freiburg, Germany
Sophie Tourret	INRIA and MPI for Informatics, France
Natarajan Shankar	SRI International, USA
Sidney C. Nogueira	Universidade Federal Rural de Pernambuco, Brazil
Cesare Tinelli	University of Iowa, USA
Lucas Cordeiro	University of Manchester, UK
Clare Dixon	University of Manchester, UK

Additional Reviewers

Levente Bajczi
Laura Bussi
Bence Graics
Lars Michael Kristensen

Invited Talks and Tutorial

Neurosymbolic AI to Achieve Trustworthy AI

Artur d'Avila Garcez

City, University of London, UK

Abstract. Current advances in Artificial Intelligence (AI) and Machine Learning (ML) have achieved unprecedented impact across research communities and industry. Nevertheless, concerns around trust, safety, interpretability and accountability of AI were raised by influential thinkers. Many identified the need for well-founded knowledge representation and reasoning to be integrated with Deep Learning (DL). Neurosymbolic AI has been an active area of research for many years seeking to do just that, bringing together robust learning in neural networks with reasoning and explainability via symbolic representations. Our focus is on research that integrates in a principled way neural-network learning with symbolic AI. In this keynote I will review the research in neurosymbolic AI and computation, and how it can help shed new light onto the increasingly prominent role of safety, trust, interpretability and accountability of AI. We also identify promising directions and challenges for the next decade of AI research from the perspective of neurosymbolic computation. Over the past decade, AI and in particular DL has attracted media attention, has become the focus of increasingly large research endeavours and has changed businesses. This led to influential debates on the impact of AI in academia and industry. It has been argued that the building of a rich AI system, semantically sound, explainable and ultimately trustworthy, will require a sound reasoning layer in combination with deep learning. Parallels have been drawn between Daniel Kahneman's research on human reasoning and decision making, and so-called "AI systems 1 and 2" which would in principle be modelled by deep learning and symbolic reasoning, respectively.

We seek to place 20 years of research in the area of neurosymbolic AI, known as neural-symbolic integration, in the context of the recent explosion of interest and excitement around the combination of deep learning and symbolic reasoning. We revisit early theoretical results of fundamental relevance to shaping the latest research, such as the proof that recurrent neural networks compute the semantics of logic programming, and we identify bottlenecks and the most promising technical directions for the sound representation of learning and reasoning in neural networks. As well as pointing to the various related and promising techniques, we aim to help organise some of the terminology commonly used around AI, ML and DL. This is important at this exciting time when AI becomes popularized among researchers and practitioners from other areas of Computer Science and from other fields altogether, psychology, cognitive science, economics, medicine, engineering and neuroscience.

I will survey some of the prominent forms of neural-symbolic integration. We address neural-symbolic integration from the perspectives of distributed and localist forms of representation, and argue for a focus on logical representation based on the assumption that representation precedes learning and reasoning.

We delve into the fundamentals of current neurosymbolic AI methods and systems and identify promising aspects of neurosymbolic AI to address exciting challenges for learning, reasoning, validation and explainability. Finally, based on all of the above, we propose a list of ingredients for neurosymbolic AI and discuss promising directions for future research to address the challenges of AI.

Collaborating Reasoners: Theory Combination Beyond Nelson-Oppen

Stéphane Graham-Lengrand

SRI International, USA

Abstract. The Nelson-Oppen scheme constitutes a cornerstone of SMT-solving by providing a systematic recipe for interfacing theory-specific reasoners. In this scheme, the reasoners can simply be black boxes whose only requirements are to be decision procedures for (quantifier-free) satisfiability in their respective theories. To make them collaborate, extra properties are required of the theories to be combined, rather than of the reasoners; for instance, the theories should be disjoint in that they only share the equality symbol.

In this talk, we will range over the design and the benefits of several alternative schemes where reasoners collaborate by answering more complex queries than pure satisfiability queries and/or by satisfying stronger requirements than simply being decision procedures for their underlying theories. Among such designs are the CDSAT scheme where completeness and termination of reasoners are stated in a combination-aware form, as well as several schemes, like QSMA, that rely on the reasoners' ability to produce over-and under-approximations of the input formula. The benefits include the support of non-disjoint theory combinations, additional freedom in the lemmas to be learned, new techniques for interpolation, and new techniques for supporting quantifiers.

`Sniper`: Automated Reasoning for Type Theory

Chantal Keller

Université Paris-Saclay, France

Abstract. For formal proofs to become mainstream in software and hardware development, as well as mathematical formalization, automation plays an essential role. Many systems already enjoy a high degree of automation, such as deductive verification tools for proof of programs. In the case of interactive theorem proving, provers based on Higher Order Logic now often provide hammers, which are very powerful tools that call many external automated provers in parallel and propose a meaningful proof script if possible.

For interactive provers based on Type theory, though, attempts to build hammers have given good results, but appear to be less powerful and hardly predictable than for Higher Order Logic. More generally, in such systems, a variety of automatic tactics are available, but expertise is still required to use them: one needs to know when they apply, how to combine them, and apparently small changes in a goal can completely break a tactic. We give non-exhaustive examples in the Coq proof assistant:

– the `Micromega` plugin provides various tactics to reason about integer linear arithmetic, but it is non trivial to apply them when integers live in types out of Coq's standard library, and by design it cannot be applied modulo congruence;
– the `CoqHammer` plugin provides tactics to call various first-order provers, as well as to reconstruct their proofs, but it lacks theory reasoning such as integer arithmetic, and it is very hard to predict when the provers or proof reconstruction will succeed;
– the `SMTCoq` plugin provides tactics to call various SMT solvers and reconstruct their proofs, but it is limited to goals expressed in Boolean logic and with a very specific shape;
– ...

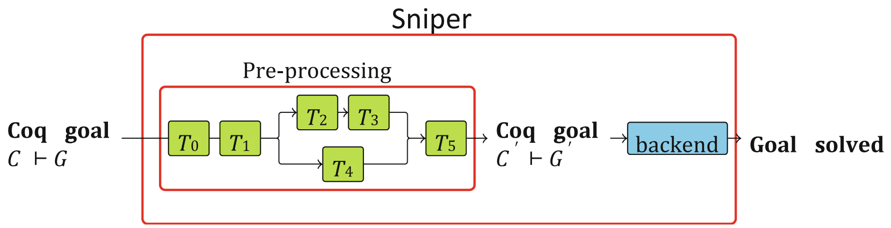
We analyze these difficulties in this way.

– Tactics for general automation (such as `CoqHammer`) are very hard to predict because there is a gap between Type theory and first-order logic that prevents anticipating if solvers and proof reconstruction will succeed.
– Tactics for more specific automation (such as `Micromega` and `SMTCoq`) are easier to predict, but apply to very specific goals, and expertise is needed to obtain or recognize such goals.

`Sniper`: Compositional Pre-processing

To reconcile the two methods, we propose a new approach that makes use of existing tactics for specific automation and tries to combine them to obtain predictive and extensible general automation. This approach is being implemented in the Coq plugin `Sniper`[1], whose development is under progress.

It is based on the following architecture:



`Sniper` pre-processes goals before calling an automatic tactic dedicated to specific automation (called *backend* in the figure) such as `SMTCoq` or `Micromega`. The key idea is that pre-processing is not a monolithic transformation, but it is a dynamic composition of fine-grained transformations (called T_1 to T_5 in the figure) that can be taken from a pool; the backend can also be any tactic that (partially) solves a given class of problems. By *dynamic*, we mean that the transformations that are used, the order in which they are applied, and the chosen backend are not fixed, but depend on the original goal.

The advantages of this approach are the following.

- It is adaptive, and can thus apply to a variety of goals.
- It should be quite predictive from the pools of transformations and backends.
- It is compositional, and contributors can easily add new transformations or backends to extend the tactic. Note that more powerful backends such as `CoqHammer` can also be used, as they become more predictive if goals are pre-processed into specific classes of problems.
- Fine-grained transformations tackle one aspect of Coq logic at a time, which make them easy to produce partial proofs (such as Coq tactics do); and partially preserve goal's structure, making some automatic backends such as `SMTCoq` more likely to succeed.

As of writing, the implementation of `Sniper` already provides a library of around fifteen certifying transformations designed for this architecture, and a prototype tactic `snipe`. Work in progress consists in making `Sniper` dynamic (as explained above) and designing an API for contributors to easily add new transformations and backends.

[1] https://github.com/smtcoq/sniper.

Acknowledgments. Sniper is common work with Louise Dubois de Prisque, Pierre Vial and Valentin Blot. It relies on SMTCoq, which is the work of many smart people, who are listed here: https://github.com/smtcoq/smtcoq/blob/coq-8.13/AUTHORS. We also thank Enzo Crance, Denis Cousineau, Assia Mahboubi and Kazuhiko Sakaguchi for fruitful discussions on this work.

References

1. Armand, M., Faure, G., Grégoire, B., Keller, C., Théry, L., Werner, B.: A modular integration of SAT/SMT aolvers to Coq through proof witnesses. In: Jouannaud, J.P., Shao, Z. (eds.) Certified Programs and Proofs. CPP 2011. LNCS, vol. 7086, pp. 135–150. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-25379-9_12
2. Besson, F.: Fast reflexive arithmetic tactics the linear case and beyond. In: Altenkirch, T., McBride, C. (eds.) Types for Proofs and Programs. TYPES 2006. LNCS, vol. 4502, pp. 48–62. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-74464-1_4
3. Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.:Hammering towards QED. J. Formalized Reasoning **9**(1),101–148 (2016). https://doi.org/10.6092/issn.1972-5787/4593
4. Blot, V., et al.: Compositional pre-processing for automatedreasoning in dependent type theory. In: Krebbers, R., Traytel, D.,Pientka, B., Zdancewic, S. (eds.) Proceedings of the 12th ACM SIGPLANInternational Conference on Certified Programs and Proofs,CPP 2023, Boston, MA, USA, January 16–17, 2023, pp. 63–77. ACM(2023). https://doi.org/10.1145/3573105.3575676
5. Czajka, L., Kaliszyk, C.: Hammer for Coq: automation for dependenttype theory. J. Autom. Reason. **61**(1–4), 423–453 (2018).https://doi.org/10.1007/s10817-018-9458-4
6. Desharnais, M., Vukmirovic, P., Blanchette, J., Wenzel, M.: Seventeenprovers under the hammer. In: Andronick, J., de Moura,L. (eds.) 13th International Conference on Interactive TheoremProving, ITP 2022, August 7–10, 2022, Haifa, Israel.LIPIcs, vol. 237, pp. 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrumfür Informatik (2022). https://doi.org/10.4230/LIPIcs.ITP.2022.8,https://www.dagstuhl.de/dagpub/978-3-95977-252-5
7. Filliâtre, J.C., Paskevich, A.: Why3 — where programs meet provers. In: Felleisen, M., Gardner, P. (eds.) Programming Languages and Systems. ESOP 2013. LNCS, vol. 7792, pp. 125–128. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-37036-6_8
8. Sakaguchi, K.: Micromega tactics for mathematical components(2019–2022), https://github.com/math-comp/mczify
9. Swamy, N., et al.: Dependent types and multimonadiceffects in F. In: Symposium on Principles of ProgrammingLanguages (POPL)

Formal Methods in Systems Engineering - Verifying SysML v2 Models

Vince Molnár

BME-FTSRG, Hungary

Abstract. Formal methods have been successfully applied to several fields in engineering, including software, hardware, and communication protocols. Systems engineering is an interdisciplinary field that focuses on how to design, integrate, and manage complex systems over their lifecycles. Models in systems engineering may capture the specification of both software and hardware components, but also processes, physical aspects, and even expected user behavior, as well as abstract descriptions of scenarios in which the system is expected to operate. Due to the integration aspect, there is a heavy emphasis on the interplay between these various viewpoints. Even though there would be plenty of use cases to apply formal methods, V&V in systems engineering is still typically performed in the form of manual reviews, and only smaller components and their implementations are analyzed with automated formal verification tools.

The Systems Modeling Language (SysML) is the de facto standard modeling language for designing and developing complex systems. The second version of SysML is a complete redesign, including changes like moving away from UML, adding an expression language, and adopting a 4D ontology-like semantics based on classification and logic. Many of these changes make SysML v2 more suitable for formal analysis than its predecessor. At the same time, the ever-increasing complexity and the increasingly popular notion of executable modeling are creating demand to automate analysis tasks. Automation is expected to save time and resources for engineers and reduce manual errors, especially in the engineering of critical systems.

In this tutorial, we provide an overview of use cases of formal methods in systems engineering, then introduce the fundamentals of the SysML v2 language, focusing on its declarative 4D semantics and how it handles temporal aspects. We take a look at formal verification approaches from the perspective of the new language, including new techniques devised for parallel programs, as well as model generation. Model execution is a central topic in the community around the new standard, so we dedicate some time to present the ongoing efforts related to execution, semantics, and formal methods. Finally, we present an early prototype for model checking SysML v2 models and discuss the challenges and open problems in the field.

Contents