From Verification to Explanation (Track Introduction)

Christel Baier[1] and Holger Hermanns[2,3]

[1] Technische Universität Dresden, Germany
[2] Universität des Saarlandes, Saarland Informatics Campus, Germany
[3] Institute of Intelligent Software, Guangzhou, China

1 Introduction

It is becoming the norm that software artefacts participate in actions and decisions that affect humans. This trend has been catching momentum for decades, and is now amplified considerably by the remarkable abilities of machine-learnt methods.

However, our understanding of what is the cause of a specific automated decision is lagging far behind. More severe, we are lacking the scientifc basis to explain how several such applications interact in cascades of automated decisions. With the increase in cyber-physical technology impacting our lives, the consequences of this gradual loss in understanding are becoming severe.

The long-term ambition of this track of ISOLA is to explore how computer aided verification techniques can be leveraged to master the explanation challenge. Our focus are algorithmic and tool-supported approaches that aim at making the behaviour of software and CPS systems understandable.

2 Context

Scientifc work on explanations has not been an explicit focus of past research in the verification community, but nevertheless there are a number of prominent techniques that can be considered individual attack points for orchestrated efforts. We review the (according to our limited understanding) major research directions below.

Explaining negative verification results. Most model-checking tools accompany negative verification results with *counterexamples* to provide an evidence why the model violates its specification. These are finite prefixes of erroneous executions of the model. Although such counterexamples can support debugging, they often tend to be long and identifying the failure in the counterexample trace becomes a non-trivial task. This observation has motivated research on how to extract the relevant information from counterexamples that is needed for debugging purposes. Following the structural equation approach of Halpern and Pearl, the causality-based analysis of counterexamples has been first proposed by Beer et al. [2] with the intention to generate and visualise user-understandable

T. Margaria and B. Steffen (Eds.): ISoLA 2020, LNCS 12479, pp. 3–9, 2021.
https://doi.org/10.1007/978-3-030-83723-5_1

explanations in terms of diagrams for selected paths. This approach has been further advanced by analysing sets of counterexample traces for the purpose of extracting causal relations and representing them by logical formulas, which are then used to generate visualisable fault trees [32,31]. Other techniques aiming at explanations of negative model-checking results rely on distance metrics for program executions to support understanding and the localisation of errors [17,18]. An approach for cause consequence analysis using temporal logics has been presented in [36,20].

Explaining positive verification results. Orthogonal to the approaches for explaining negative verification results is *vacuity detection* and the *certification* of verification results. Vacuity detection [3,27,5] is motivated by the observation that positive verification results cannot rule out cases where the model is wrong or where there is no perfect match between the formal specification and the desired requirements. It relies on a stronger ("non-vacuous") satisfaction relation than in standard model checking and aims to report sufficiently informative witnesses for non-vacuously valid temporal formulas in a given model. Another direction is the *certification* of formal verification results [28,38] where the task is to accompany positive model-checking results with a certificate that serves as evidence for the successful system verification and can be checked separately to confirm that the system indeed meets its specification. Certification techniques for probabilistic models have been proposed that are based on Farkas certificates [16,23,22] and shown to be related to the construction of witnessing subsystems as in [1,40,21]. There is also recent work on certification techniques for timed automata [43,42]. One step further is the line of research that addresses the verification and certification of formal verifiers using theorem-proving techniques, see, e.g., [13,24].

Causality Reasoning. Besides the above mentioned work on the causality-based analysis of counterexamples to extract user-understandable explanations of model-checking results, combinations of causality-based reasoning and model-checking techniques have also been used by Chockler et al. [8,9,4,7] to reason about the *degree of the responsibility* of components for the satisfaction or violation of system properties as well as related coverage metrics. First steps towards compositional causality reasoning in nonprobabilistic systems and for temporal events of a simple modal logic have been presented recently in [6].

Another research direction is to exploit synergies between causality techniques and model checking. An on-the-fly approach to detect causal relationships and to classify execution traces as good or bad with respect to the property to be checked has been presented in [32] and extended in [33] for the quantitative analysis of Markov chains. The essential idea of the *causality-based model-checking* [29,30,15] is to avoid the explicit reference to states of a system model as it is the case for standard model checking. Instead it relies on a notion of concurrent traces and causal links between them and proof rules given by trace transformers. Together with sophisticated data structures, this approach can reduce the complexity of model checking for multi-threaded programs from exponential to polynomial time.

Verfication Explanations for Humans. Many forms of graphical models have been introduced to reason about causality. Examples include cause-effect graphs that have been introduced in the context of software testing [35] or Petri nets [10], or causal graphs [12] that have been used to design tractable algorithms for deciding different forms for causes in Halpern and Pearl's structural-model approach.

In the probabilistic setting, various graphical models have been proposed in the literature to represent causal dependencies. Most prominent are Bayesian networks that rely on directed acyclic graphs where the nodes represent variables and the edges indicate conditional dependencies (see, e.g., [37]). To overcome the limitations of classical Bayesian networks that assume discrete variables and do not support reasoning about time, several extensions have been proposed in the literature to formalise how the dependencies evolve over time slices (dynamic Bayesian networks) or to reason about continuous variables (e.g., hybrid or heterogeneous Bayesian networks). Another prominent visualisable model are (dynamic) fault trees [39,11], a well-established industrial standard and graphical notation to illustrate how a hazard can be caused by a combination of so called basic events. In the context of probabilistic model checking, the generation of fault trees from probabilistic counterexamples for Markov chains has been studied by Leitner-Fischer et al. [26,34,31].

Aiming at human-understandable textual explanations, the translations of minimal critical sub-MDPs (counterexamples for MDPs as in [40]) into guarded command language has been developed in [41]. The recent paper [14] proposes an alternative approach based on structural natural language sentences to describe the behaviour that leads to a violation of system requirements.

3 Contributions in this Track

For the 2020 edition of ISOLA, the track editors have selected two contribution that represent the spectrum of research on verification methods for explanations very well.

The paper by Kölbl and Leue entitled *An Algorithm to Compute a Strict Partial Orderering of Actions in Action Trees* [25] focusses on tool support for causality checking. At its core is a novel method for computing a causal explanation. This explanation here takes the form of an ordered sequences of actions that lead to a violation of a reachability property. Earlier work in this context was able to compute the unordered set of such actions, while the present contribution additionally provides them in a strictly partial ordered form, thus giving more specific axplanatory feedback to the user. The approach is implemented in the tool QuantUM and its performance and usability is discussed.

The paper by Gros et al. entitled *TraceVis: Towards Visualization for Deep Statistical Model Checking* [19] showcases a very innovative explanation component for neural network behaviour. It starts off from deep statistical model checking (DSMC), a recently proposed approach to statistically analyse the behaviour of a neural network employed as a decision entity to solve a family of two-dimensional navigation problems, known as the Racetrack. The DSMC analysis delivers a variety of estimates of numerical nature. The present paper explores the use of visualization techniques to support human analysts and domain engineers when exploringthese results. The authors present an interactive visualization tool which enables visual exploration of Racetrack crash probabilities as well as in-depth examination of the policy traces generated by DSMC. By this, the authors succesfully demonstrate how visualization can foster the effective model-checking-based analysis for the purpose of advanced explanation support for neural network behaviour.

Acknowledgments. This initiative would not have been possible without the support by the Deutsche Forschungsgemeinschaft for the Center for Perspicuous Computing (TRR 248, Grant 389792660). It furthermore has received support by the Key-Area Research and Development Program Grant 2018B010107004 of Guangdong Province.

References

1. Erika Ábrahám, Bernd Becker, Christian Dehnert, Nils Jansen, Joost-Pieter Katoen, and Ralf Wimmer. Counterexample generation for discrete-time markov models: An introductory survey. In Marco Bernardo, Ferruccio Damiani, Reiner Hähnle, Einar Broch Johnsen, and Ina Schaefer, editors, *Formal Methods for Executable Software Models - 14th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM)*, volume 8483 of *Lecture Notes in Computer Science*, pages 65–121. Springer, 2014.
2. Ilan Beer, Shoham Ben-David, Hana Chockler, Avigail Orni, and Richard J. Trefler. Explaining counterexamples using causality. *Formal Methods in System Design*, 40(1):20–40, 2012.
3. Ilan Beer, Shoham Ben-David, Cindy Eisner, and Yoav Rodeh. Efficient detection of vacuity in temporal model checking. *Formal Methods in System Design*, 18(2):141–163, 2001.
4. Shoham Ben-David, Hana Chockler, and Orna Kupferman. Attention-based coverage metrics. In *9th Int. Haifa Verification Conf. on Hardware and Software: Verification and Testing (HVC)*, volume 8244 of *LNCS*, pages 230–245. Springer, 2013.
5. Shoham Ben-David, Fady Copty, Dana Fisman, and Sitvanit Ruah. Vacuity in practice: temporal antecedent failure. *Formal Methods in System Design*, 46(1):81–104, 2015.
6. Georgiana Caltais, Stefan Leue, and Mohammad Reza Mousavi. (de-)composing causality in labeled transition systems. In *First Workshop on Causal Reasoning for Embedded and safety-critical Systems Technologies*, volume 224 of *EPTCS*, pages 10–24, 2016.
7. Hana Chockler, Norman E. Fenton, Jeroen Keppens, and David A. Lagnado. Causal analysis for attributing responsibility in legal cases. In *15th Int. Conf. on Artificial Intelligence and Law (ICAIL)*, pages 33–42. ACM, 2015.

8. Hana Chockler and Joseph Y. Halpern. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research (JAIR)*, 22:93–115, 2004.
9. Hana Chockler, Joseph Y. Halpern, and Orna Kupferman. What causes a system to satisfy a specification? *ACM Transactions on Computational Logic*, 9(3), 2008.
10. Jörg Desel, Andreas Oberweis, Torsten Zimmer, and Gabriele Zimmermann. Validation of information system models: Petri nets and test case generation. In *IEEE Int. Conf. on Cybernetics and Simulation*, pages 3401–3406, 1997.
11. J.B. Dugan, S.J. Bavuso, and M.A. Boyd. Dyanamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, 1992.
12. Thomas Eiter and Thomas Lukasiewicz. Causes and explanations in the structural-model approach: Tractable cases. *Artifical Intelligence*, 170(6-7):542–580, 2006.
13. Javier Esparza, Peter Lammich, René Neumann, Tobias Nipkow, Alexander Schimpf, and Jan-Georg Smaus. A fully verified executable LTL model checker. *Archive of Formal Proofs*, 2014.
14. Lu Feng, Mahsa Ghasemi, Kai-Wei Chang, and Ufuk Topcu. Counterexamples for robotic planning explained in structured language. *CoRR*, arXiv:1803.08966, 2018. To appear in IEEE Int. Conf. on Robotics and Automation (ICRA'18).
15. Bernd Finkbeiner, Manuel Gieseking, and Ernst-Rüdiger Olderog. Adam: Causality-based synthesis of distributed systems. In *27th Int. Conf. on Computer Aided Verification (CAV)*, volume 9206 of *LNCS*, pages 433–439. Springer, 2015.
16. Florian Funke, Simon Jantsch, and Christel Baier. Farkas certificates and minimal witnesses for probabilistic reachability constraints. In Armin Biere and David Parker, editors, *26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 12078 of *Lecture Notes in Computer Science*, pages 324–345. Springer, 2020.
17. Alex Groce. Error explanation with distance metrics. In *10th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2988 of *LNCS*, pages 108–122. Springer, 2004.
18. Alex Groce, Daniel Kroening, and Flavio Lerda. Understanding counterexamples with explain. In *16th Int. Conf. on Computer Aided Verification (CAV)*, volume 3114 of *LNCS*, pages 453–456. Springer, 2004.
19. Timo P. Gros, David Groß, Stefan Gumhold, Jörg Hoffmann, Michaela Klauck, and Marcel Steinmetz. Tracevis: Towards visualization for deep statistical model checking. in this volume.
20. Axel Habermaier, Alexander Knapp, Johannes Leupolz, and Wolfgang Reif. Fault-aware modeling and specification for efficient formal safety analysis. In *Critical Systems: Formal Methods and Automated Verification (FMICS-AVoCS)*, volume 9933 of *LNCS*, pages 97–114. Springer, 2016.
21. Nils Jansen. *Counterexamples in probabilistic verification.* PhD thesis, RWTH Aachen University, Germany, 2015.
22. Simon Jantsch, Florian Funke, and Christel Baier. Minimal witnesses for probabilistic timed automata. In Dang Van Hung and Oleg Sokolsky, editors, *18th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 12302 of *Lecture Notes in Computer Science*, pages 501–517. Springer, 2020.
23. Simon Jantsch, Hans Harder, Florian Funke, and Christel Baier. SWITSS: computing small witnessing subsystems. In Alexander Ivrii and Ofer Strichman, editors, *20th Conference on Formal Methods in Computer-Aided Design (FMCAD).* Academic Press TU Wien, 2020.

24. Jacques-Henri Jourdan, Vincent Laporte, Sandrine Blazy, Xavier Leroy, and David Pichardie. A formally-verified C static analyzer. In *42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 247–259. ACM, 2015.
25. Martin Kölbl and Stefan Leue. An algorithm to compute a strict partial orderering of actions in action trees. in this volume.
26. Matthias Kuntz, Florian Leitner-Fischer, and Stefan Leue. From probabilistic counterexamples via causality to fault trees. In *30th Int. Conf. on Computer Safety, Reliability, and Security*, volume 6894 of *LNCS*, pages 71–84. Springer, 2011.
27. Orna Kupferman and Moshe Y. Vardi. Vacuity detection in temporal model checking. In *10th IFIP WG 10.5 Advanced Research Working Conf. on Correct Hardware Design and Verification Methods (CHARME)*, volume 1703 of *LNCS*, pages 82–96. Springer, 1999.
28. Orna Kupferman and Moshe Y. Vardi. From complementation to certification. *Theoretical Computer Science*, 345(1):83–100, 2005.
29. Andrey Kupriyanov and Bernd Finkbeiner. Causality-based verification of multi-threaded programs. In *24th Int. Conf. on Concurrency Theory (CONCUR)*, volume 8052 of *LNCS*, pages 257–272. Springer, 2013.
30. Andrey Kupriyanov and Bernd Finkbeiner. Causal termination of multi-threaded programs. In *26th Int. Conf. on Computer Aided Verification (CAV)*, volume 8559 of *LNCS*, pages 814–830, 2014.
31. Florian Leitner-Fischer. *Causality Checking of Safety-Critical Software and Systems.* PhD thesis, University of Konstanz, Germany, 2015.
32. Florian Leitner-Fischer and Stefan Leue. Causality checking for complex system models. In *14th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 7737 of *LNCS*, pages 248–267. Springer, 2013.
33. Florian Leitner-Fischer and Stefan Leue. On the synergy of probabilistic causality computation and causality checking. In *20th Int. Symp. on Model Checking Software (SPIN)*, volume 7976 of *LNCS*, pages 246–263. Springer, 2013.
34. Florian Leitner-Fischer and Stefan Leue. Probabilistic fault tree synthesis using causality computation. *Int. Journal of Critical Computer-Based Systems*, 4(2):119–143, 2013.
35. Glenford J. Myers. *The Art of Software Testing.* John Wiley & Sons, 1979.
36. F. Ortmeier, W. Reif, and G. Schellhorn. Formal safety analysis of a radio-based railroad crossing using deductive cause-consequence analysis. In *5th European Dependable Computing Conf. (EDCC)*, volume 3463 of *LNCS*. Springer, 2006.
37. Judea Pearl. *Causality: Models, Reasoning and Inference.* Cambridge University Press, 2nd edition, 2009.
38. Ali Taleghani. *Using Software Model Checking for Software Certification.* PhD thesis, University of Waterloo, Ontario, Canada, 2010.
39. W.E. Vasely and F.F. Goldberg. *Fault Tree Handbook.* US Nuclear Regulatory Commission, 2014. NUREG-0492.
40. Ralf Wimmer, Nils Jansen, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker. Minimal counterexamples for linear-time probabilistic verification. *Theoretical Computer Science*, 549:61–100, 2014.
41. Ralf Wimmer, Nils Jansen, Andreas Vorpahl, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker. High-level counterexamples for probabilistic automata. *Logical Methods in Computer Science*, 11(1), 2015.

42. Simon Wimmer, Frédéric Herbreteau, and Jaco van de Pol. Certifying emptiness of timed büchi automata. In Nathalie Bertrand and Nils Jansen, editors, *18th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 12288 of *Lecture Notes in Computer Science*, pages 58–75. Springer, 2020.
43. Simon Wimmer and Joshua von Mutius. Verified certification of reachability checking for timed automata. In Armin Biere and David Parker, editors, *26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 12078 of *Lecture Notes in Computer Science*, pages 425–443. Springer, 2020.