



HAL
open science

Recent advances in interactive and automated analysis

Radu Mateescu

► **To cite this version:**

Radu Mateescu. Recent advances in interactive and automated analysis. France. International Journal on Software Tools for Technology Transfer, 20 (2), pp.119 - 123, 2018, 10.1007/s10009-017-0477-y . hal-01766570

HAL Id: hal-01766570

<https://inria.hal.science/hal-01766570v1>

Submitted on 13 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Recent Advances in Interactive and Automated Analysis

Radu Mateescu

Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP*, LIG, F-38000 Grenoble, France

Abstract. Computers and distributed software applications are becoming nowadays ubiquitous, and therefore their safety and reliability have increasingly important societal impact. In this context, formal methods equipped with powerful and versatile analysis tools are more important than ever in the design process. Despite the relevant scientific results and well-established tools obtained in recent years, there is a constant need of enhancing the analysis capabilities in order to handle increasingly complex systems. We briefly discuss some recent advances in the field, introducing five papers selected from the 22th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2016).

1 Introduction

This special issue of the journal *Software Tools for Technology Transfer* (STTT) contains revised and extended versions of five papers selected out of 18 tool papers presented at the 22th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16) [11], held in Eindhoven, The Netherlands during April 2-8, 2016, as part of the Joint European Conferences on Theory and Practice of Software (ETAPS). These five papers were invited by the guest editor from the best ranked regular tool papers and tool demonstration papers presented at TACAS'16, according to their relevance to STTT.

TACAS is a forum for researchers, developers and users interested in rigorously based tools and algorithms for the construction and analysis of systems. The topics covered by TACAS include: formal methods, software and hardware verification techniques, real-time, hybrid,

and stochastic systems, program analysis and testing, tool environments, and case studies.

Computers and distributed software applications are becoming nowadays ubiquitous, in particular with the advent of the Internet of Things (IoT), and therefore their safety and reliability have increasingly important societal impact. In this context, formal methods equipped with analysis tools — as essential ingredients of a rigorous design process — are more important than ever in certifying a correct and efficient functioning of complex systems. Despite the relevant scientific results and well-established tools obtained in recent years, there is a constant need of scaling up the analysis capabilities in order to handle increasingly larger systems.

The selected papers cover three topics, illustrating how recent theoretical achievements are instantiated in sophisticated analysis tools available to the community.

The first topic is interactive theorem proving, a semi-automated approach to formalize and prove mathematical statements, in particular about the correctness of programs or the behaviour of concurrent systems. This approach is supported by interactive theorem provers, such as those presented in [16,22], which assist users in developing proofs by alleviating the burden of proving tedious lemmas, ideally keeping the user interaction only for the creative aspects of the proofs.

The second topic focuses on equivalence checking, an automated verification approach for finite-state concurrent systems, which consists of comparing the behaviour of a system with the behaviour of its intended service (by abstracting away the internal actions) modulo bisimulation relations. The combination of symbolic representation of state spaces and parallel algorithms yields high-performance equivalence checkers, as illustrated in [39].

Finally, the third topic concerns the analysis of probabilistic and stochastic systems, which enables designers to obtain quantitative information about the performance of a system in addition to functional verification.

* Institute of Engineering Univ. Grenoble Alpes

This approach is supported by probabilistic model checkers, such as those presented in [26,29], that operate on symbolic representations of extended Markovian models and carry out the analysis of probabilistic temporal properties and the synthesis of strategies.

The rest of this preface is organized as follows. Section 2 discusses interactive theorem provers. Section 3 discusses parallel algorithms for equivalence checking. Section 4 discusses probabilistic model checking and strategy synthesis for stochastic systems. Section 5 gives some concluding remarks.

2 Interactive Theorem Proving

Proof assistants or interactive theorem provers (ITPs) (see [1] for a survey) enable one to state a theorem in a suitable mathematical language and then prove that theorem within the ITP. This guarantees (assuming the ITP itself is bug-free) that the proof is correct and developed in full detail. With the help of ITPs, one can mechanize the construction of very large proofs, the insightful, interesting steps being carried out by the human user and the simpler (and often tedious) steps being taken in charge by the proof assistant. There are many state-of-the-art ITPs available, for example Coq [5], PVS [33], or Isabelle/HOL [32], to name a few amongst the most popular ones. ITPs have been used to carry out mechanized proofs in mathematics, such as the 4-colour theorem [20], the Odd order theorem [21] or Cauchy’s residue theorem [30], to certify optimizing C compilers [9], and to verify complex software or hardware systems, such as component-based architectures [10], operating system kernels [17], or complete microprocessors [6].

Similarly to the development of large software systems using mainstream programming languages, the usage of ITPs for building large proofs can be substantially facilitated by IDEs (*Integrated Development Environments*). An IDE combines a variety of tools (editors, compilers, refactorers, profilers, debuggers, project and release managers) into a single unified toolbox, keeping track automatically of dependencies. Despite the fact that ITPs become increasingly popular, relatively few of them are equipped with full-fledged IDEs.

The paper *Coqoon — An IDE for Interactive Proof Development in Coq* [16] by Alexander Faithfull, Jesper Bengtson, Enrico Tassi, and Carst Tankink, which extends the TACAS’16 publication [15], presents Coqoon, an Eclipse-based IDE for proof development using Coq. This IDE provides support for Coq projects similar to the Eclipse built-in support for Java projects, by enabling Coq users to create Coq projects, structure them hierarchically, and make changes anywhere in a file, thus improving on the waterfall model used in previous proof environments for Coq, such as CoqIDE or Proof General [2]. Coqoon takes advantage of the latest features of Coq, including asynchronous and parallel processing

of proofs, and (together with an OCaml extension for Eclipse) can be used to work on large developments containing Coq plugins.

Besides the benefits of an IDE for developing and managing large proof projects, a essential ingredient of an ITP is the definition and manipulation of *tactics*, which alleviate user interaction by reducing goals to smaller and simpler ones. ITPs are equipped with tactic languages, such as Ltac [14] or HITAC [3], which enable users to combine tactics and to elaborate high-level proof strategies. However, the strategies built using these languages can easily become hard to understand, and even harder to analyse and debug.

The paper *The Tinker Tool for Graphical Tactic Development* [22] by Gudmund Grov and Yuhui Lin, which extends the TACAS’16 publication [31], proposes Tinker, an ITP-independent environment for the development and maintenance of proof tactics based on the graphical language PSGraph. Tinker provides versatile tactic debugging features and an ergonomic user interface that significantly enhance the construction of proof patterns and strategies, and consequently the productivity of mechanizing proofs using ITPs.

3 Parallel Verification

The capabilities of verification tools can be naturally increased by exploiting the computing resources of multicore computers or clusters of machines. Over the past two decades, several aspects of the verification process have been subject to parallelization or distribution:

State space manipulation. In the explicit-state setting, the Labeled Transition Systems (LTSs) modeling the behaviour of concurrent systems can be efficiently generated using clusters of machines [19] and then explored and reduced on the fly modulo τ -confluence (a form of partial order reduction adequate with branching bisimulation) using specialized distributed algorithms [18]. Symbolic manipulation of state spaces based on BDDs (Binary Decision Diagrams) can also be parallelized and subsequently used as back-end for verification tools, as for instance the Sylvan multicore multi-valued BDD package [37].

Model checking. Parallel graph exploration algorithms are successfully applied to the efficient verification of properties expressed in temporal logic. The spectrum of properties considered ranges from invariants, as in the parallel variant of the Murphi model checker [36], up to LTL properties as handled by the parallel variants of SPIN [24,23]. A recent parallelization approach for verifying LTL properties in the explicit-state setting is the so-called *swarm verification*, in which a large number of parallel processes cooperate in exploring the state space and detecting accepting cycles [40] or strongly connected components [35].

Equivalence checking. This verification approach is based on bisimulation relations [12] between LTSs, and consists essentially in computing the equivalence classes on states by using partition refinement algorithms. After the early parallelization of the classical partition refinement algorithms [34], equivalence checking has been primarily subject to distributed algorithms based on the computation of *signatures* over states and transitions in order to accelerate the convergence of partition refinement [7, 8].

An effective way to fight state explosion is the combination of parallelization techniques with symbolic representations of state spaces. This cumulates the benefits of both approaches, potentially leading to high-performance verification tools. This is the topic of the paper *Multi-core Symbolic Bisimulation Minimization* [39] by Tom van Dijk and Jaco van de Pol, an extended version of the TACAS'16 publication [38], which presents a highly efficient and scalable parallel symbolic bisimulation checker built on top of the Sylvan library.

4 Probabilistic and Stochastic Systems

The quantitative modeling and analysis of concurrent systems enables one to obtain, in addition to functional properties (e.g., absence of deadlocks, safety, or liveness), also information about the performance of the system (e.g., reachability of certain states within a number of steps and a given probability threshold). Among the various approaches for quantitative analysis, probabilistic verification [4, Chap. 10] has been thoroughly investigated, leading to popular model checkers such as PRISM [27] and STORM [13]. These tools operate on (extended) Markovian models and allow one to express properties in probabilistic temporal logics, such as PCTL (*Probabilistic CTL*) and PLTL (*Probabilistic LTL*).

Probabilistic model checkers have numerous applications in various fields (protocols, cyber-physical systems, randomized algorithms, etc.) and are continuously improved with new features and capabilities. A collection of enhancements brought to the PRISM model checker is illustrated in the paper *Advances in Probabilistic Model Checking with PRISM: Variable Reordering, Quantiles and Weak Deterministic Büchi Automata* [26] by Joachim Klein, Christel Baier, Philipp Chrszon, Marcus Daum, Clemens Dubschlaff, Sascha Klüppelholz, Steffen Märcker, and David Müller, which extends the TACAS'16 publication [25]. Three improvements to PRISM are proposed: automatic variable reordering for the symbolic engines to finely tune the performance of operations on multi-terminal BDDs; symbolic computation of quantiles in Markov decision processes; and computation of minimal weak deterministic Büchi automata for an LTL fragment for the purpose of checking bounded reward properties.

Another illustration of extending PRISM is described in the paper *PRISM-games: Verification and Strategy Synthesis for Stochastic Multi-player Games with Multiple Objectives* [29] by Marta Kwiatkowska, David Parker, and Clemens Wiltsche, which extends the TACAS'16 publication [28]. The authors present PRISM-games, a new tool built upon PRISM, dedicated to the modelling, verification and strategy synthesis for stochastic multi-player games using multi-objective and compositional approaches. These extensions, brought by the original authors of PRISM, are orthogonal to the ones presented in [26]. The set of new features and enhancements encompassed by these works significantly improve the versatility of the model checker and are likely to widen its application fields.

5 Conclusion

Some recent advances in interactive and automated analysis have been discussed and related to selected papers from TACAS'16, included in this volume. Three domains have been identified: interactive theorem provers; parallel algorithms for bisimulation checking; probabilistic and stochastic systems. The five papers presented in this volume illustrate the panel of possibilities offered to concurrent system designers by recent developments: functional verification, quantitative analysis, and proof checking, all of which concern different facets of a rigorous design process based on formal methods.

Acknowledgments

We are grateful to all authors for their contributions, as well as to the reviewers of TACAS'16 and of this special issue for their careful and constructive examination of the manuscripts.

References

1. Special section on theorem proving. *Int. J. Softw. Tools Technol. Transf.* **3**(1) (2000)
2. Aspinall, D. Proof general: A generic tool for proof development. In: *Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'00)*. LNCS, vol. 1785, pp. 38–42. Springer, Berlin (2000)
3. Aspinall, D., Denney, E., Lüth, C. A tactic language for hiproofs. In: *Proceedings of the 9th International Conference on Intelligent Computer Mathematics (AISC'08)*. LNCS, vol. 5144, pp. 339–354. Springer, Berlin (2008)
4. Baier, C., Katoen, J.-P. *Principles of Model Checking* (MIT Press, 2008)
5. Bertot, Y., Castéran, P. *Interactive Theorem Proving and Program Development – Coq'Art: The Calculus of Inductive Constructions* (Springer, Berlin, 2004)

6. Beyer, S., Jacobi, C., Kröning, D., Leinenbach, D., Paul, W.-J. Putting it all together - formal verification of the VAMP. *Int. J. Softw. Tools Technol. Transf.* **8**(4-5), 411–430 (2006)
7. Blom, S., Orzan, S. A distributed algorithm for strong bisimulation reduction of state spaces. *Int. J. Softw. Tools Technol. Transf.* **7**(1), 74–86 (2005)
8. Blom, S., van de Pol, J. Distributed branching bisimulation minimization by inductive signatures. In: *Proceedings of the 8th International Workshop on Parallel and Distributed Methods in verification (PDMC'09)*. EPCTS vol. 14, pp. 32–46 (2009)
9. Boldo, S., Jourdan, J.-H., Leroy, X., Melquiond, G. A formally-verified C compiler supporting floating-point arithmetic. In: *Proceedings of the 21st IEEE Symposium on Computer Arithmetic (ARITH'13)*, pp. 107–115. IEEE (2013)
10. Brucker, A.D., Wolff, B. A verification approach to applied system security. *Int. J. Softw. Tools Technol. Transf.* **7**(3), 233–247 (2005)
11. Chechik, M., Raskin J.-F. (eds.): *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*. LNCS vol. 9636. Springer, Berlin (2016)
12. Cleaveland, R., Sokolsky, O. Equivalence and preorder checking for finite-state systems. In: *Handbook of Process Algebra*, pp. 391–424 (Elsevier, Amsterdam, 2001)
13. Dehnert, C., Junges, S., Katoen, J.-P., Volk, M. A storm is coming: A modern probabilistic model checker. In: *Proceedings of the 29th International Conference on Computer Aided Verification (CAV'17)*. LNCS vol. 10427, pp. 592–600. Springer, Berlin (2017)
14. Delahaye, D. A tactic language for the system Coq. In: *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR'00)*, LNCS vol. 1955, pp. 85–95. Springer, Berlin (2000)
15. Faithfull, A., Bengtson, J., Tassi, E., Tankink, C. Coqoon - an IDE for interactive proof development in Coq. In: *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*. LNCS vol. 9636, pp. 316–331. Springer, Berlin (2016)
16. Alexander Faithfull, Jesper Bengtson, Enrico Tassi, and Carst Tankink. Coqoon — an ide for interactive proof development in Coq. *Int. J. Softw. Tools Technol. Transf.*, <http://dx.doi.org/10.1007/s10009-017-0477-y> (2017)
17. Ferreira, J.-F., Gherghina, C., He, G., Qin, S., Chin, W.-N. Automated verification of the freeRTOS scheduler in Hip/Sleek. *Int. J. Softw. Tools Technol. Transf.* **16**(4), 381–397 (2014)
18. Garavel, H., Mateescu, M., Serwe, W. Large-scale distributed verification using CADP: Beyond clusters to grids. In: *Proceedings of the 11th International Workshop on Parallel and Distributed Methods in verification (PDMC'12)*. ENTCS vol. 296, pp. 145–161. Elsevier (2013)
19. Garavel, H., Mateescu, R., Smarandache, I. Parallel state space construction for model-checking. In: *Proceedings of the 8th International SPIN Workshop on Model Checking of Software (SPIN'01)*. LNCS vol. 2057, pp. 217–234. Springer, Berlin (2001)
20. Gonthier, G. The four colour theorem: Engineering of a formal proof. In: *Proceedings of the 8th Asian Symposium on Computer Mathematics (ASCM'07)*. LNCS vol. 5081, pp. 333. Springer, Berlin (2007)
21. Gonthier, G., Asperti, A., Avigad, J., Bertot, Y., Cohen, C., Garillot, F., Le Roux, S., Mahboubi, A., O'Connor, R., Biha, S.-O., Pasca, I., Rideau, L., Solovyev, A., Tassi, E., Théry, L. A machine-checked proof of the odd order theorem. In: *Proceedings of the 4th International Conference on Interactive Theorem Proving (ITP'13)*. LNCS vol. 7998, pp. 163–179. Springer, Berlin (2013)
22. Grov, G., Lin, Y. The Tinker tool for graphical tactic development. *Int. J. Softw. Tools Technol. Transf.*, <http://dx.doi.org/10.1007/s10009-017-0477-y> (2017)
23. Holzmann, G.J. Parallelizing the SPIN model checker. In: *Proceedings of the 19th International Workshop on Model Checking Software (SPIN'12)*. LNCS vol. 7385, pp. 155–171. Springer, Berlin (2012)
24. Holzmann, G.J., Bosnacki, D. The design of a multi-core extension of the SPIN model checker. *IEEE Trans. Software Eng.* **33**(10), 659–674 (2007)
25. Klein, J., Baier, C., Chrszon, P., Daum, M., Dubslaff, C., Klüppelholz, S., Märcker, S., Müller, D. Advances in symbolic probabilistic model checking with PRISM. In: *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*. LNCS vol. 9636, pp. 349–366. Springer, Berlin (2016)
26. Klein, J., Baier, C., Chrszon, P., Daum, M., Dubslaff, C., Klüppelholz, S., Märcker, S., Müller, D. Advances in probabilistic model checking with PRISM: Variable reordering, quantiles and weak deterministic Büchi automata. *Int. J. Softw. Tools Technol. Transf.*, <http://dx.doi.org/10.1007/s10009-017-0477-y> (2017)
27. Kwiatkowska, M., Norman, G., Parker, D. PRISM 4.0: Verification of probabilistic real-time systems. In: *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*. LNCS vol. 6806, pp. 585–591. Springer, Berlin (2011)
28. Kwiatkowska, M., Parker, D., Wiltsche, C. PRISM-games 2.0: A tool for multi-objective strategy synthesis for stochastic games. In: *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*. LNCS vol. 9636, pp. 560–566. Springer, Berlin (2016)
29. Kwiatkowska, M., Parker, D., Wiltsche, C. PRISM-games: Verification and strategy synthesis for stochastic multi-player games with multiple objectives. *Int. J. Softw. Tools Technol. Transf.*, <http://dx.doi.org/10.1007/s10009-017-0477-y> (2017)
30. Li, W., Paulson, L.-C. A formal proof of Cauchy's residue theorem. In: *Proceedings of the 7th International Conference on Interactive Theorem Proving (ITP'16)*. LNCS vol. 9807, pp. 235–251. Springer, Berlin (2016)
31. Lin, Y., Le Bras, P., Grov, G. Developing and debugging proof strategies by tinkering. In: *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*. LNCS vol. 9636, pp. 573–579. Springer, Berlin (2016)
32. Nipkow, T., Paulson, L.C., Wenzel, M. Isabelle/HOL - A Proof Assistant for Higher-Order Logic. LNCS vol. 2283. Springer, Berlin (2002)

33. Owre, S., Rushby, J.M., Shankar, N. PVS: A prototype verification system. In: Proceedings of the 11th International Conference on Automated Deduction (CADE'92). LNCS vol. 607, pp. 748–752. Springer, Berlin (1992)
34. Rajasekaran, S., Lee, I. Parallel algorithms for relational coarsest partition problems. *IEEE Trans. Parallel Distrib. Syst.* **9**(7), 687–699 (1998)
35. Renault, E., Duret-Lutz, A., Kordon, K., Poitrenaud, D. Parallel explicit model checking for generalized Büchi automata. In: Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'15). LNCS vol. 9035, pp. 613–627. Springer, Berlin (2015)
36. Stern, U., Dill, D.L. Parallelizing the Murphi verifier. *Formal Methods in System Design* **18**(2), 117–129 (2001)
37. van Dijk, T., van de Pol, J. Sylvan: Multi-core decision diagrams. In: Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'15). LNCS vol. 9035, pp. 677–691. Springer, Berlin (2015)
38. van Dijk, T., van de Pol, J. Multi-core symbolic bisimulation minimisation. In: Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16). LNCS vol. 9636, pp. 332–348. Springer, Berlin (2016)
39. van Dijk, T., van de Pol, J. Multi-core symbolic bisimulation minimisation. *Int. J. Softw. Tools Technol. Transf.*, <http://dx.doi.org/10.1007/s10009-017-0477-y> (2017)
40. Wijs, A. Towards informed swarm verification. In: Proceedings of the 3rd International Symposium on NASA Formal Methods (NFM'11). LNCS vol. 6617, pp. 422–437. Springer, Berlin (2011)