International Journal on Software Tools for Technology Transfer (2018) 20:493–497
https://doi.org/10.1007/s10009-018-0501-x

INTRODUCTION

Model checking: recent improvements and applications

Dragan Bošnački[1] · **Anton Wijs**[1]

Published online: 24 July 2018

Abstract
Model checking (Baier and Katoen in Principles of model checking, MIT Press, Cambridge, 2008; Clarke et al. in Model checking, MIT Press, Cambridge, 2001) is an automatic technique to formally verify that a given specification of a concurrent system meets given functional properties. Its use has been demonstrated many times over the years. Key characteristics that make the method so appealing are its level of automaticity, its ability to determine the absence of errors in the system (contrary to testing techniques) and the fact that it produces counter-examples when errors are detected, that clearly demonstrate not only that an error is present, but also how the error can be produced. The main drawback of model checking is its limited scalability, and for this reason, research on reducing the computational effort has received much attention over the last decades. Besides the verification of qualitative functional properties, the model checking technique can also be applied for other types of analyses, such as planning and the verification of quantitative properties. We briefly discuss several contributions in the model checking field that address both its scalability and its applicability to perform planning and quantitative analysis. In particular, we introduce six papers selected from the 23rd International SPIN Symposium on Model Checking Software (SPIN 2016).

Keywords Model checking · Planning · Strategy synthesis · Probabilistic model checking · Partial-order reduction

1 Introduction

The current issue of the journal Software Tools for Technology Transfer (STTT) contains six revised and extended versions of papers presented at the 23rd International SPIN Symposium on Model Checking Software (SPIN 2016) [8]. SPIN 2016 was held in Eindhoven, The Netherlands, on 7–8 April 2016 collocated with the Joint European Conferences on Theory and Practice of Software (ETAPS). These six papers were selected by the guest editors out of the sixteen papers presented at the event, based on their ranking given by the peer reviewers.

During the last two decades the SPIN symposiums have established themselves as traditional annual forums for researchers and practitioners for the verification of software systems. The evolution of the SPIN events has to a great extent mirrored the maturing of model checking into a prevailing technology for the formal verification of software systems. The first SPIN workshop was held in Montreal in 1995. The next couple of subsequent editions of SPIN were intended as gatherings for presenting extensions and applications of the model checker SPIN [24], to which the series owes its name. Starting with the 2000 edition, the scope of the event clearly broadened to include techniques for formal verification and testing in general. In addition, the SPIN events aim to promote interaction and exchange of ideas across related software engineering areas, like static and dynamic analysis.

This special issue nicely demonstrates the current scope of the SPIN events. First of all, in addition to the SPIN model checker, contributions in this issue use the tool TAPAAL [13], the Afra model checking tool [29], the ASSET tool [40], and the CADP toolbox [19].

Second of all, the majority of the papers in this issue are on extending and applying model checking beyond its traditional set-up, i.e. the formal verification of concurrent systems w.r.t. qualitative behavioural properties. Four of the six papers are on the application of model checking to construct a strategy or plan to solve a particular scheduling or control problem constrained by time and/or resource requirements. Another paper is on on-the-fly verification of

✉ Anton Wijs
A.J.Wijs@tue.nl

Dragan Bošnački
D.Bosnacki@tue.nl

[1] Eindhoven University of Technology, Eindhoven, The Netherlands

quantitative properties via probabilistic model checking [3]. In that sense, one of the papers is more traditional in its scope, but it addresses the main drawback of model checking, i.e. its limited scalability, by contributing to the topic of partial-order reduction [22,35,39], a very effective technique to mitigate state space explosion.

The remainder of this preface is organised as follows: Section 2 discusses the use of model checking for the synthesis of strategies and plans. In Sect. 3, the verification of quantitative properties by means of probabilistic model checking is considered. Partial-order reduction to on-the-fly reduce state spaces explored by model checkers is discussed in Sect. 4. Finally, in Sect. 5, some concluding remarks are given.

2 Planning and strategy synthesis

The application of model checking to construct a plan or synthesise a strategy is not far-fetched, as model checking and planning have much in common [1,11,37,43,44]: in both cases, a (large) state space has to be explored, looking for interesting behaviour. While in traditional model checking, this behaviour is essentially undesirable, violating some functional properties, in planning the interesting behaviour is desirable and constitutes a successful plan to optimise a system while fulfilling given constraints. When synthesising a strategy, typically the notion of a controller is added to the model, and the question is whether there exists a strategy for that controller such that any possible behaviour under that strategy satisfies the specification.

In the paper *Integrating river basin DSSs with model checking* by del Mar Gallardo et al. [18], which extends their SPIN 2016 paper [17], it is demonstrated how the SPIN model checker can be applied in a decision support system (DSS) that mitigates the effects of floods in river basins. Model checking is used to synthesise management recommendations that meet the constraints given by the dam manager. A set of constraints is added to a PROMELA model that interacts with an external model for the river basin. SPIN exhaustively explores all possible manoeuvres and produces a trace, i.e. a sequence of manoeuvres, that fulfils the given constraints.

The paper *A Case Study of Planning for Smart Factories – Model Checking and Monte-Carlo Search for the Rescue* by Edelkamp and Greulich [15], which extends their SPIN 2016 paper [16], proposes to use the SPIN model checker to construct plans for multi-agent systems that control the industrial production of goods. Assembling stations use queues to buffer materials, and the core objective is to optimise the throughput of the system. The authors demonstrate that by using branch-and-bound searching, optimised plans consisting of thousands of steps can be produced in reasonable time. For comparison, they also consider using a Monte Carlo search framework and conclude that such an approach is even better in constructing plans. They conjecture that building a model checker that uses Monte Carlo search is an interesting topic to investigate in future work.

Of course, timing is crucial when synthesising strategies to control real-time systems, but its introduction makes the use of model checking more challenging. The previous contribution handles timing by carefully modelling it explicitly such that a model checker unaware of timing could still be used. An alternative is to use model checking techniques that natively support timing. Symbolic continuous-time on-the-fly methods, such as those employed in the tools Kronos [9], UPPAAL [5], Tina [6] and Romeo [20], have been employed in on-the-fly algorithms for controller synthesis [4,36]. However, for such a task, discrete-time methods turn out to be very competitive [2].

The paper *Discrete and Continuous Strategies for Timed-Arc Petri Net Games* by Jensen et al. [25], which extends their SPIN 2016 paper [26], addresses this topic and proposes an on-the-fly algorithm for the synthesis of timed controllers relative to safety objectives. It turns out that when restricting the context to the use of urgent controllers that act immediately or wait for another occurrence of the same event, then discrete-time methods can be used to determine the existence of a continuous-time safety controller.

Schedulability and resource utilisation of wireless sensor and actuator network (WSAN) applications are addressed in the paper *Modeling and Analyzing Real-Time Wireless Sensor and Actuator Networks Using Actors and Model Checking* by Khamespanah et al. [27]. This paper extends their SPIN 2016 paper [28]. Such applications can be modelled by defining a number of concurrent actors, each providing services that can be requested by other actors by sending messages. Schedulability of the operations can be checked using Timed Rebeca, and Timed Computation Tree Logic (TCTL) model checking can be performed to check more complicated properties, such as minimal resource utilisation.

3 Probabilistic model checking

To check quantitative properties of systems, for example referring to time constraints or energy consumption, models can be extended with probabilities associated with behavioural events. The potential behaviour of such systems can then be captured in Markov Chains or probabilistic transition systems (PTSs) [21], which essentially are discrete-time Markov Chains in which transitions are labelled with actions and probabilities, and communication between concurrent processes is modelled. Probabilistic model checkers, such as PRISM [30] and STORM [14], can be used to analyse these Markov Chains and determine whether they satisfy given probabilistic properties.

To express these properties, suitable temporal logics need to be defined, such as probabilistic computation tree logic (PCTL) [23].

In the paper *On-the-Fly Model Checking for Extended Action-Based Probabilistic Operators* by Mateescu and Requeno [32], which extends their SPIN 2016 paper [33], a new regular probabilistic operator is proposed to specify the probability measure of a path described by a generalised regular formula involving computations on data values. This operator subsumes the until operators of PCTL and their action-based counterparts. The authors integrate this operator into MCL (Model Checking Language) and implement an on-the-fly model checking method in the CADP toolbox.

4 Partial-order reduction

The partial-order reduction (POR) technique [22,35,39] is perhaps the most efficient technique to mitigate the state space explosion problem in model checking. In recognition of this fact the founding fathers of POR, Godefroid, Peled, Valmari, and Wolper, received the 2014 CAV award. POR exploits the observation that the state space may contain several paths that are similar, in the sense that their differences are not relevant to the property under consideration. By pruning certain transitions, the size of the state space can be reduced.

The current issue features the paper *Fair Testing and Stubborn Sets* by Valmari and Vogler [41], which extends their SPIN 2016 paper [42]. Valmari was the first to notice the necessity for the so-called *cycle proviso* to ensure the correctness of POR when cycles are present in the state space. In the presence of cycles, POR without such a proviso may incorrectly terminate after having investigated a cycle, consistently ignoring behaviour that leaves the cycle. Hence, this problem is known as the *ignoring problem*. The cycle proviso turned out to be crucial for various adaptations of POR to different search orders of the state space (such as breadth-first search [7]), as well as parallel searches, both for shared memory (in settings using multiple cores [31] and graphics processing units [34]) and distributed architectures [10,38].

In the paper by Valmari and Vogler, it is proven that a partial-order method originally proposed for trace equivalence also preserves fair testing equivalence, in which deadlocks are unified with livelocks that cannot be exited. Thus, it supports a practical fairness assumption. Compared to the original SPIN 2016 paper, the extended version presents new observations regarding the ignoring problem in this context, remarking that the preservation of trace and fair testing equivalence does not imply that the ignoring problem is addressed.

5 Conclusions

Recent improvements and applications in the field of model checking have been discussed and associated with six papers selected from SPIN 2016, that have been included in this special issue. Four of the six papers contribute work on the application of model checking techniques to construct schedules and plans for planning problems, and synthesise strategies for control problems. In addition, one paper contributes to the verification of quantitative properties, and one contributes to the topic of partial-order reduction. Together, these papers address both the strengthening of the model checking method itself and its applicability to efficiently solve problems outside its traditional scope.

Acknowledgements We are grateful to all authors for their contributions as well as to the reviewers of SPIN 2016 and of this special issue for their careful and constructive examination of the manuscripts.

References

1. Abdeddaïm, Y., Maler, O.: Job-shop scheduling using timed automata. In: Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001), Lecture Notes in Computer Science, vol. 2102, pp. 478–492. Springer, Berlin (2001)
2. Andersen, M., Larsen, H., Srba, J., Sørensen, M., Taankvist, J.: Verification of liveness properties on closed timed-Arc Petri nets. In: Proceedings of the 8th International Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2012), Lecture Notes in Computer Science, vol. 7721, pp. 69–81. Springer, Berlin (2012)
3. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
4. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K., Lime, D.: UPPAAL-Tiga: time for playing games! In: Proceedings of the 19th International Conference on Computer Aided Verification (CAV 2007), Lecture Notes in Computer Science, vol. 4590, pp. 121–125. Springer, Berlin (2007)
5. Behrmann, G., David, A., Larsen, K., Hakansson, J., Petterson, P., Yi, W., Hendriks, M.: UPPAAL 4.0. In: Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems (QEST 2006), pp. 125–126. IEEE Computer Society, Washington, DC (2006)
6. Berthomieu, B., Vernadat, F.: Time Petri nets analysis with TINA. In: Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems (QEST 2006), pp. 123–124. IEEE Computer Society, Washington, DC (2006)
7. Bošnački, D., Leue, S., Lluch-Lafuente, A.: Partial-order reduction for general state exploring algorithms. STTT **11**(1), 39–51 (2009)
8. Bošnački, D., Wijs, A. (eds.): Proceedings of the 23rd International SPIN Symposium on Model Checking of Software, Lecture Notes in Computer Science, vol. 9641. Springer, Berlin (2016)

9. Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., Yovine, S.: Kronos: a model-checking tool for real-time systems. In: Proceedings of the 10th International Conference on Computer Aided Verification (CAV 1998), Lecture Notes in Computer Science, vol. 1427, pp. 546–550. Springer, Berlin (1998)
10. Brim, L., Černá, I., Moravec, P., Šimša, J.: Distributed partial order reduction of state spaces. In: Proceedings of the 3rd International Workshop on Parallel and Distributed Methods in Verification (PDMC 2004), Electronic Notes in Theoretical Computer Science, vol. 128, pp. 63–74. Elsevier, New York (2004)
11. Brinksma, E., Mader, A., Fehnker, A.: Verification and optimisation of a PLC control schedule. STTT **4**(1), 21–33 (2002)
12. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (2001)
13. David, A., Jacobsen, L., Jacobsen, M., Jørgensen, K., Møller, M., Srba, J.: TAPAAL 2.0: integrated development environment for timed-Arc Petri nets. In: Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2012), Lecture Notes in Computer Science, vol. 7214, pp. 492–497. Springer, Berlin (2012)
14. 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 2017), Lecture Notes in Computer Science, vol. 10427, pp. 592–600. Springer, Berlin (2017)
15. Edelkamp, S., Greulich, C.: A case study of planning for smart factories-model checking and Monte-Carlo search for the rescue. Int. J. Softw. Tools Technol. Transf. (2018). https://doi.org/10.1007/s10009-018-0498-1
16. Edelkamp, S., Greulich, C.: Using SPIN for the optimized scheduling of discrete event systems in manufacturing. In: Proceedings of the 23rd International SPIN Symposium on Model Checking of Software, Lecture Notes in Computer Science, vol. 9641, pp. 57–77. Springer, Berlin (2018)
17. Gallardo, M., Merino, P., Panizo, L., Salmerón, A.: River basin management with SPIN. In: Proceedings of the 23rd International SPIN Symposium on Model Checking of Software, Lecture Notes in Computer Science, vol. 9641, pp. 78–96. Springer, Berlin (2016)
18. Gallardo, M., Merino, P., Panizo, L., Salmerón, A.: Integrating river basin DSSs with model checking. Int. J. Softw. Tools Technol. Transf. (2017). https://doi.org/10.1007/s10009-017-0478-x
19. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: CADP 2011: a toolbox for the construction and analysis of distributed processes. STTT **15**(2), 89–107 (2013)
20. Gardey, G., Lime, D., Magnin, M., Roux, O.: Romeo: a tool for analyzing time Petri nets. In: Proceedings of the 17th International Conference on Computer Aided Verification (CAV 2005), Lecture Notes in Computer Science, vol. 3576, pp. 418–423. Springer, Berlin (2005)
21. van Glabbeek, R., Smolka, S., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Inf. Comput. **121**(1), 59–80 (1995)
22. Godefroid, P., Wolper, P.: A partial approach to model checking. Inf. Comput. **110**(2), 305–326 (1994)
23. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Form. Asp. Comput. **6**(5), 512–535 (1994)
24. Holzmann, G.: The SPIN Model Checking: Primer and Reference Manual. Addison-Wesley, Boston (2003)
25. Jensen, P.-G., Larsen, K.G., Srba, J.: Discrete and continuous strategies for timed-Arc Petri net games. Int. J. Softw. Tools Technol. Transf. (2017). https://doi.org/10.1007/s10009-017-0473-2
26. Jensen, P., Larsen, K., Srba, J.: Real-time strategy synthesis for timed-Arc Petri net games via discretization. In: Proceedings of the 23rd International SPIN Symposium on Model Checking of Software, Lecture Notes in Computer Science, vol. 9641, pp. 129–146. Springer, Berlin (2018)
27. Khamespanah, E., Sirjani, M., Mechitov, K., Agha, G.: Modeling and analyzing real-time wireless sensor and actuator networks using actors and model checking. Int. J. Softw. Tools Technol. Transf. (2018). https://doi.org/10.1007/s10009-017-0480-3
28. Khamespanah, E., Sirjani, M., Mechitov, K., Agha, G.: Schedulability analysis of distributed real-time sensor network applications using actor-based model checking. In: Proceedings of the 23rd International SPIN Symposium on Model Checking of Software, Lecture Notes in Computer Science, vol. 9641, pp. 165–181. Springer, Berlin (2018)
29. Khamespanah, E., Sirjani, M., Sabahi-Kaviani, Z., Khosravi, R., Izadi, M.J.: Timed rebeca schedulability and deadlock freedom analysis using bounded floating time transition system. Sci. Comput. Program. **98**(P2), 184–204 (2015)
30. 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 2011), Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer, Berlin (2011)
31. Laarman, A., Wijs, A.: Partial-order reduction for multi-core LTL model checking. In: Proceedings of the 10th Haifa Verification Conference (HVC 2014), Lecture Notes in Computer Science, vol. 8855, pp. 267–283. Springer, Berlin (2014)
32. Mateescu, R., Requeno, J.I.: On-the-fly model checking for extended action-based probabilistic operators. Int. J. Softw. Tools Technol. Transf. (2018). https://doi.org/10.1007/s10009-018-0499-0
33. Mateescu, R., Requeno, J.: On-the-fly model checking for extended action-based probabilistic operators. In: Proceedings of the 23rd International SPIN Symposium on Model Checking of Software, Lecture Notes in Computer Science, vol. 9641, pp. 189–207. Springer, Berlin (2018)
34. Neele, T., Wijs, A., Bošnački, D., Pol, J.v.d.: Partial-order reduction for GPU model checking. In: Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA 2016), Lecture Notes in Computer Science, vol. 9938, pp. 357–374. Springer, Berlin (2016)
35. Peled, D.: All from one, one for all: on model checking using representatives. In: CAV 1993, Proceedings, vol. 697, pp. 409–423 (1993)
36. Pnueli, A., Asarin, E., Maler, O., Sifakis, J.: Controller synthesis for timed automata. In: Proceedings of the 5th IFAC Conference on System Structure and Control (SSC 1998), IFAC Proceedings Volumes, vol. 31, pp. 447–452. Elsevier, New York (1998)
37. Ruys, T.: Optimal scheduling using branch and bound with SPIN 4.0. In: Proceedings of the 10th International SPIN Workshop on Model Checking Software, Lecture Notes in Computer Science, vol. 2648, pp. 1–17. Springer, Berlin (2003)
38. Simsa, J., Bryant, R., Gibson, G., Hickey, J.: Scalable dynamic partial order reduction. In: Proceedings of the 3rd International Conference on Runtime Verification, Lecture Notes in Computer Science, vol. 7687, pp. 19–34. Springer, Berlin (2012)
39. Valmari, A.: Stubborn sets for reduced state space generation. Adv. Petri Nets **483**, 491–515 (1991)
40. Valmari, A.: A state space tool for concurrent system models expressed in C++. In: Proceedings of the 14th Symposium on Programming Languages and Software Tools (SPLST 2015), CEUR Workshop Proceedings, vol. 1525, pp. 91–105. CEUR-WS.org (2015)
41. Valmari, A., Vogler, W.: Fair testing and stubborn sets. Int. J. Softw. Tools Technol. Transf. (2018). https://doi.org/10.1007/s10009-017-0481-2
42. Valmari, A., Vogler, W.: Fair testing and stubborn sets. In: Proceedings of the 23rd International SPIN Symposium on Model Checking of Software, Lecture Notes in Computer Science, vol. 9641, pp. 225–243. Springer, Berlin (2018)

43. Wijs, A.: What to do next? analysing and optimising system behaviour in time. Ph.D. Thesis, Vrije Universiteit Amsterdam (2007)
44. Wijs, A., Fokkink, W.: From χ_t to μCRL: Combining Performance and Functional Analysis. In: Proceedings of the 10th International Conference on Engineering of Complex Computer Systems (ICECCS 2005), pp. 184–193. IEEE Computer Society, Washington, DC (2005)