Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Introduction to Selected Papers from SPIN 2017

Hakan Erdogmus[1], **Klaus Havelund**[2⋆]

[1] Carnegie Mellon University Silicon Valley, Mountain View, USA
[2] Jet Propulsion Laboratory, California Inst. of Technology, Pasadena, USA

Received: date / Revised version: date

The 24th International SPIN Symposium on Model Checking of Software (SPIN 2017) [2] took place at the University of California, Santa Barbara, on July 13 and 14, 2017. The symposium brought together researchers working in automated, tool-based techniques for the analysis, verification, and validation of software systems, models, and programs. In this special issue, we bring to you extended versions of selected papers from the symposium. To assemble the issue, we issued invitations to the seven most favorably reviewed papers from SPIN 2017. The authors of five of these papers accepted the invitation. Each submission subsequently went through the regular review process of the International Journal on Software Tools for Technology Transfer, requiring three reviews per paper. We used the reviewers of the original papers when we could, and supplemented this set with additional reviewers when necessary. The five papers published in this special issue are the result of this process and represent the diversity of SPIN 2017, combining theoretical and practical contributions. They address efficient, novel, and realistic ways of performing program synthesis and model checking. The first three selections each directly target a different practical concern, and the last two make a theoretical contribution, but with a practical impact.

The first paper addresses synthesis of Java programs through program sketching. In *"EdSketch: Execution-Driven Sketching for Java"* [4], authors Jinru Hua, Yushan Zhang, Yuqun Zhang, and Sarfraz Khurshid propose a method to complete partial Java programs and test assertions that the programs' complete versions must pass, using an execution-driven approach. The main advantage of this approach is that it does not rely on manually produced models of the libraries on which the programs may depend, thus sidestepping a practical obstacle to program sketching.

In the second paper, *"An Integrated Environment for Spin-Based C Code Checking - Towards Bringing Model-Driven Code Checking Closer to Practitioners"* [6], Daniel Ratiu and Andreas Ulrich advocate a model-driven approach to verify functional requirements of C programs, similarly focussing on an impediment to their practical application. Their approach uses Spin as the underlying verification engine, but avoids hard-to-produce and obscure environment models that mix C and Promela code to define the required verification harnesses. They propose instead a higher-level and compact language, prototyped in the mbeddr platform, with the added advantage of being able to express and verify properties beyond traditional assertions.

The third paper by Michalis Kokologiannakis and Konstantinos Sagonas is a case study that tackles a real-world model checking problem. In *"Stateless Model Checking of the Linux Kernel's Read-Copy-Update (RCU)"* [5], the authors analyze a complicated memory synchronization mechanism used heavily in the Linux kernel. They use a stateless model checking tool to verify the main guarantees of this mechanism and reproduce known safety and liveness violations in a very efficient way. Their success suggests that model checking could become part of the standard assurance approach of large foundational software systems that we heavily rely on.

The last two papers are theoretical contributions in model checking with impact on performance.

In *"Model Checking with Generalized Rabin and Fin-less Automata"* [1], authors Vincent Bloemen, Alexandre Duret-Lutz, and Jaco van de Pol explore the feasibility of using two alternative forms of automata with different acceptance conditions for model checking LTL properties. The new forms tend to generate fewer states, but have higher computational complexity. The authors investigate this tradeoff by comparing the performance of

[⋆] The research performed by this author was carried out at Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

the new forms to the traditional Büchi-based form. They design parallel algorithms for the new forms and test their algorithms using existing model checking benchmarks. Although the new forms do not produce the expected improvements, the experiments provide insights on how different features of the new forms impact performance, thus paving the way for new algorithms that can capitalize on these insights.

The final paper, by John Fearnley, Sanjay Jain, Bart de Keijzer, Sven Schewe, Frank Stephan, and Dominik Wojtczak, tackles a common graph-theoretic problem pertinent to model checking and program synthesis. In *"An Ordered Approach to Solving Parity Games in Quasi-Polynomial Time and Quasi-Linear Space"* [3], the authors provide an efficient implementation of an existing algorithm for solving parity games, a class of zero-sum graph coverage games involving two antagonistic players. The significance of this works is that, if the problem of determining the winner in these types of games can be solved efficiently, the most expensive steps of model checking and program synthesis can also be performed efficiently by reducing these steps to the corresponding parity games. The paper compares the performance of the authors' implementation to competing parity game solutions using a set of standard examples, noting performance differences and their reasons across problem categories.

Hakan Erdogmus and Klaus Havelund
SPIN 2017 PC Chairs

References

1. Vincent Bloemen, Alexandre Duret-Lutz, and Jaco van de Pol. Model checking with generalized Rabin and Fin-less automata. *International Journal on Software Tools for Technology Transfer, STTT, in this issue*, 2019.
2. Hakan Erdogmus and Klaus Havelund, editors. *SPIN 2017: Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, Santa Barbara, CA, USA, 2017. ACM.
3. John Fearnley, Sanjay Jain, Bart de Keijzer, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi-polynomial time and quasi-linear space. *International Journal on Software Tools for Technology Transfer, STTT, in this issue*, 2019.
4. Jinru Hua, Yushan Zhang, Yuqun Zhang, and Sarfraz Khurshid. EdSketch: Execution-driven sketching for Java. *International Journal on Software Tools for Technology Transfer, STTT, in this issue*, 2019.
5. Michalis Kokologiannakis and Konstantinos Sagonas. Stateless model checking of the Linux kernels Read-Copy-Update (RCU). *International Journal on Software Tools for Technology Transfer, STTT, in this issue*, 2019.
6. Daniel Ratiu and Andreas Ulrich. An integrated environment for Spin-based C code checking - towards bringing model-driven code checking closer to practitioners. *International Journal on Software Tools for Technology Transfer, STTT, in this issue*, 2019.