Int J Softw Tools Technol Transfer (2016) 18:355–357
DOI 10.1007/s10009-016-0414-5

INTRODUCTION

Preface of the special issue on Model Checking of Software

Selected papers of the 20th International SPIN Symposium on Model Checking of Software

Ezio Bartocci[1] · **C. R. Ramakrishnan**[2]

Published online: 10 February 2016

Abstract Software Model Checking consists of a broad collection of techniques to tackle the complexity and the diversity in the use of software in safety-critical systems. The contributions in this special issue address some of the core problems in software model checking. The articles are based on papers selected from the 2013 SPIN Symposium on Model Checking of Software, an annual forum for practitioners and researchers interested in symbolic and state space-based techniques for the validation and analysis of software systems.

Keywords Model checking · Software verification · Formal methods

1 Background

Verification of software systems raises several unique problems. These problems range from the use of complex data and control structures in software to the variety of systems implemented in software, for example in cyber-physical systems. This special issue contains six contributions that address some of the core issues in software model checking. The papers were selected from articles presented at SPIN 2013, the 20th International Symposium on Model Checking of Software, which was held on 8–9 July 2013, in Stony Brook (NY), USA. The traditional focus of the SPIN series has been on explicit-state model checking techniques and empirical evaluations, as implemented in SPIN model checker and other related tools. During the recent years, the scope of the Symposium has broadened to also include techniques for the verification and formal testing of embedded software, security-critical software, enterprise and web applications, and other interesting software platforms. Papers at the symposium reflect the breadth of problems in software model checking, exemplified by the collection of articles in this issue.

Software systems have typically been characterized by the use of complex data and control structures. Contributions by Lopes et al. [12] and Sethi et al. [13] directly address problems with analyzing programs with such structures. Concurrent data structures are also considered by Adhikari et al. [1] by tackling the verification of quasi-linearizability property, which relaxes the traditional requirements for ensuring their correctness. State space reduction for the verification of software systems is addressed by Laarman et al. in [8] with a language-agnostic partial order reduction technique. Analysis of systems with timing and other resource constraints is addressed by Jensen et al. [7], where a technique for model checking weighted Kripke structures is presented. Finally, the verification of control software in cyber-physical systems is addressed by Bogomolov et al. [3], who present a technique for guided search to detect error paths in hybrid systems.

We would like to thank all the authors contributing to this special issue and the anonymous reviewers for their efforts in the reviewing process.

2 Selected papers

In the first paper in this collection, Lopes et al. [12] introduce a novel semi-algorithm for automatically proving the equivalence of programs containing nested loops over the theory of integer arithmetic combined with uninterpreted functions.

✉ Ezio Bartocci
ezio.bartocci@gmail.com

1 Technische Universität Wien, Vienna, Austria

2 Stony Brook University, Stony Brook, NY, USA

Uninterpreted functions are commonly used to abstract away parts of a program whose specifics are irrelevant to the proof under consideration. The key idea of their method is to rewrite the applications of uninterpreted functions with polynomials over the inputs of the application, and nested loops with closed-form solutions of the corresponding recurrences. This reduces the program equivalence problem to checking the equality of expressions over integers. The effectiveness of this technique has been shown using CORK, a tool for verifying the correctness of compiler optimizations.

In the second contribution, Sethi et al. [13] present a method to verify concurrent list-based data structures with unbounded number of elements and unbounded number of threads that can access them. The authors leverage the CoMPositional (CMP) method [4], a verification technique for systems with a parameterized number of processes. At first, they consider finite data structures, and use CMP to reduce unbounded number of threads to a finite model by keeping one thread concrete and abstracting the rest into single environment. This abstraction may be refined by the user, using assertions generated by an assertion mining tool Daikon, in order to prove correctness. They then extend this method to unbounded data structures as well. The aim of this work is to provide an automatic method that verifies, with limited human guidance, linearizability of protocols that use lists to implement a shared object.

In the third contribution, Adhikari et al. [1] present an automated method for verifying the *quasi-linearizability* property on highly concurrent data structures. Quasi-linearizability is a quantitative variation of the classical notion of *linearizability*, a requirement of correctness for concurrent systems stating that all the method calls on a shared object appear to take effect instantaneously between their invocation and the response. Quasi-linearizability relaxes this requirement allowing more flexibility in the data structure implementation to improve the runtime performance. In this paper the authors present two approaches. The first approach consists of verifying, using an existing model checking algorithm, the concurrent implementation with respect to a manually provided quasi-linearizable sequential specification. In the second approach, the quasi-linearizable sequential specification is instead automatically derived and the concurrent implementation is verified using a novel refinement checking algorithm.

In the fourth contribution [7], Jensen et al. present a novel algorithm to model check a Weighted Kripke Structure (WKS) with respect to a negation-free weighted Computation Tree Logic (WCTL) formula with upper-bound constraints on the weight. WKS is a formalism that extends the classical Kripke Structure with weights on the transition relation. This is suitable to model in a system functional and non-functional requirements such as timing and resource constraints. The authors show how to encode a WKS and a WCTL formula into a dependency graph [10] (DG), where the nodes are a set of configurations and the hyper-edges define how the assignment of a configuration is dependent on the others. They first prove that the model checking problem in this setting is equivalent to the computation of the pre-fixed point assignment in the DG and then provide an efficient symbolic, local and on-the-fly algorithm.

The fifth contribution [8] proposes a novel approach where the partial order reduction (POR) technique is independent of the modeling language. POR is a well-established technique that identifies subsets of interleavings of simultaneously enabled transitions that are necessary to capture all behaviors of interest during verification. Such subsets are computed based on the system description, using methods that are generally tailored to a particular specification language. The PINS interface of the LTSmin [9] separates specification languages from various model checking algorithms. This contribution proposes a language-agnostic POR technique over `Pins` by extending Valmari's stubborn sets [14]. The contribution compares the effectiveness of this technique with respect to the SPIN system's ample-set-based POR technique.

In the sixth contribution, Bogomolov et al. [3] introduce a guided search technique to detect error paths over the dynamic behavior of hybrid systems. This formalism, combining continuous and discrete dynamics, is very useful for the modeling and analysis of cyber-physical [11,15] and biological systems [2,6]. While the reachability analysis for hybrid systems is generally undecidable, in the last years modern tools such as SpaceEx [5] implement semi-decision algorithms based on over-approximation techniques to attack this problem. This contribution proposes the use of a cost function based on coarse-grained space abstractions to guide the reachability analysis. More specifically, the estimated cost is the length of the shortest trajectory (in the abstract state space) leading to an abstract error state. The paper shows, via a SpaceEx-based implementation, how this cost function can be computed efficiently and can be used effectively to guide the search as well.

Acknowledgments Open access funding provided by TU WIen (TUW).

References

1. Adhikari, K., Street, J., Wang, C., Liu, Y., Zhang, S.: Verifying a quantitative relaxation of linearizabilty via refinement. J. Softw. Tools Technol. Transf. (2016) (In this issue)

2. Bartocci, E., Bortolussi, L., Smolka, S.A.: Hybrid systems and biology. Inf. Comput. **236**, 1–2 (2014)
3. Bogomolov, S., Donzé, A., Frehse, G., Grosu, R., Johnson, T.T., Ladan, H., Podelski, A., Wehrle, M.: Guided search for hybrid systems based on coarse-grained space abstractions. J. Softw. Tools Technol. Transf. (2016) (In this issue)
4. Chou, C.T., Mannava, P.K., Park, S.: A simple method for parameterized verification of cache coherence protocols. In: Proc. of FMCAD 2004: Formal Methods in Computer-Aided Design, vol. 3312 of LNCS, pp. 382–398. Springer (2004)
5. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable Verification of Hybrid Systems. In: Proc. of CAV 2011: the 23rd International Conference on Computer Aided Verification, vol. 6806 of LNCS, pp. 379–395. Springer (2011)
6. Grosu, R., Batt, G., Fenton, F., Glimm, J., Le Guernic, C., Smolka, S.A., Bartocci, E.: From cardiac cells to genetic regulatory networks. In: Proc. of CAV 2011: the 14th International Conference on Computer Aided Verification, vol. 6806 of LNCS, pp. 396–411. Springer, Berlin, Heidelberg (2011)
7. Jensen, J.F., Larsen, K.G., Srba, J., Oestergaard, L.K.: Efficient model-checking of weighted ctl with upper-bound constraints. J. Softw. Tools Technol. Transf. (2016) (In this issue)
8. Laarman, A., Pater, E., van de Pol, J., Hansen, H.: Guard-based partial-order reduction. J. Softw. Tools Technol. Transf. (2016) (In this issue)
9. Laarman, A.W., van de Pol, J.C., Wehrle, M.: Multi-core ltsmin: marrying modularity and scalability. In: Proc. of NFM 2011: the Third International Symposium on NASA Formal Methods, vol. 6617 of LNCS, pp. 506–511. Springer (2011)
10. Liu, Y., Smolka, S.A.: Simple linear-time algorithms for minimal fixed points. In: Proc. of ICALP'98: the 25th International Colloquium on Automata, Languages and Programming, vol. 1443 of LNCS, pp. 53–66. Springer (1998)
11. Lividas, C., Lygeros, J., Lynch, N.A.: High-level modelling and analysis of tcas. In: IEEE Real-time systems symposium, pp. 115–125 (1999)
12. Lopes, N.P., Monteiro, J.: Automatic equivalence checking of programs with uninterpreted functions and integer arithmetic. J. Softw. Tools Technol. Transf. (2016) (In this issue)
13. Sethi, D., Talupur, M., Malik, S.: Model checking unbounded concurrent lists. J. Softw. Tools Technol. Transf. (2016) (In this issue)
14. Valmari, A.: Stubborn sets for reduced state space generation. In: Proceedings of the 10th International Conference on Applications and Theory of Petri Nets: Advances in Petri Nets, pp. 491–515. Springer, London (1991)
15. Varaiya, P.: Smart cars on smart roads: problems of control. IEEE Trans. Autom. Control **38**(2), 195–207 (1993)