Formal Methods in System Design (2021) 58:1–4
https://doi.org/10.1007/s10703-021-00380-6

Formal methods: practical applications and foundations

Editorial

Maurice H. ter Beek[1] · **Annabelle McIver**[2]

Accepted: 17 June 2021 / Published online: 6 July 2021

The papers in this special issue represent some of the best submissions to the 23rd Symposium on Formal Methods (FM 2019), which was part of the 3rd World Congress on Formal Methods held in Porto, Portugal in 2019.

The World Congress is an event occurring every 10 years, and attracts scientists from all over the world to celebrate and exchange ideas on the technical advances made and real-world experiences of the formal analysis of systems applied across a diversity of applications. The 3rd World Congress was both an optimistic look into "The Next 30 Years" as well as an opportunity to reflect on the 30 years since the first VDM symposium on the Vienna Development Method for formal software development in 1987 brought together researchers with the common goal of creating methods to produce high quality software based on rigour and reason.

FM 2019 attracted 129 submissions, out of which 39 were accepted for presentation. A total of 18 papers were selected to be significantly extended to journal-length papers; nine of the selections appear here and the remaining nine appear in a parallel special issue published in Formal Aspects of Computing. All papers underwent a thorough review process, requiring two to three rounds of revision. We briefly describe the contents of the nine papers that together constitute this special issue.

The paper *Pegasus: Sound Continuous Invariant Generation*, by Andrew Sogokon, Stefan Mitsch, Yong Kiam Tan, Katherine Cordwell and André Platzer, introduces Pegasus, a tool for automatic generation or synthesis of continuous invariants for hybrid systems, which serves the verification of safety properties of such systems. The tool is implemented in Mathematica and its interface is accessible through Mathematica and the KeYmaera X theorem prover. A prominent feature is that it allows to combine several well-known invariant generation methods, according to a differential saturation strategy, which can be partly customised via a set of configuration parameters. In addition, by integration within a theorem prover, the tool

✉ Maurice H. ter Beek
maurice.terbeek@isti.cnr.it

Annabelle McIver
annabelle.mciver@mq.edu.au

[1] ISTI–CNR, Pisa, Italy

[2] Macquarie University, Sydney, Australia

is able to provide formal guarantees that the generated invariants are correct. To validate the approach, the tool is applied to a benchmark set of 141 safety verification problems.

The paper *From LTL to Unambiguous Büchi Automata via Disambiguation of Alternating Automata*, by Simon Jantsch, David Müller, Christel Baier and Joachim Klein, presents a new translation from LTL formulae to unambiguous Büchi automata (UBA) through an intermediate representation of very weak alternating automata (VWAA). One of the main contributions of the paper is an algorithm to disambiguate ambiguous VWAA. To reduce the cost of the approach, the authors introduce heuristics with the aim of accelerating the computation of the unambiguous VWAA and the UBA resulting from it. This allows the efficient translation from LTL formulae to Büchi automata and outperforms state-of-the-art tools, as is demonstrated by an experimental comparison for a big set of LTL benchmarks of the tool implementing the novel technique (Duggi) with the state of the art (ltl2tgba).

The paper *Compositional Verification of Concurrent Systems by Combining Bisimulations*, by Frédéric Lang, Radu Mateescu and Franco Mazzanti, presents a compositional verification technique that allows for distinguishing weak and strong actions in the to-be-verified mu-calculus formulae in order to apply different reduction techniques to different parallel components of the systems before composing them. The goal is to reduce subprocesses modulo divergence-sensitive branching bisimulation if possible, or strong bisimulation otherwise, to achieve more reduction than the mono-bisimulation approach, while still preserving the validity of the formulae. The authors demonstrate that this approach is sound and show its practical potential by applying it to a large number of examples taken from software verification competitions.

The paper *Gray-box Monitoring of Hyperproperties with an Application to Privacy*, by Sandro Stucki, César Sánchez, Gerardo Schneider and Borzoo Bonakdarpour, studies monitorability of hyperproperties formalised in HyperLTL and applies the results on a particular privacy-related hyperproperty of distributed data minimisation. The authors show that under classical assumptions of monitorability, many hyperproperties that involve one quantifier alternation cannot be monitored. To overcome this problem, they relax the assumption of black-box monitoring (where no assumption about the monitored system is available) into a 'gray-box assumption', where monitoring is restricted to the behaviour that a system can generate. The gray-box approach for monitoring distributed data minimisation is implemented in a proof-of-concept tool, called Minion, whose performance is evaluated on a few small benchmarks.

The paper *Integrating Formal Specifications into Applications: The ProB Java API*, by Philipp Körner, Jens Bendisposto, Jannik Dunkelau, Sebastian Krings and Michael Leuschel, presents a new approach to the use of model-based design. Instead of generating code from a formal model, refining it, and embedding the code into the application, the actual formal model is directly embedded into the system in the way of a model checker or animator that does the verification at runtime. The approach is realised through the ProB Java API, which interacts with the model checker and animator ProB and hides ProB from the applications. The authors discuss six case studies that use the API (games, a ProB logical calculator, DSLs, a university timetable planner and an industrial application), thus demonstrating the feasibility of integrating formal models in a typical software development life cycle. All case studies use abstract B or Event-B models to be embedded into Java applications.

The paper *Automatic Verification of Concurrent Stochastic Systems,* by Marta Kwiatkowska, Gethin Norman, David Parker and Gabriel Santos, considers concurrent stochastic multi-player games, i.e., stochastic games where in each state multiple players choose at the same time the action they want to perform, instead of just a single player as in turn-based stochastic games. The authors present a comprehensive treatment of this topic

including a formal specification of the model and the zero- and nonzero-sum properties to be considered, an extension of the rPATL logic to capture such properties with the help of different types of Nash equilibria to model players' interaction, and model-checking algorithms for both verification and strategy synthesis with respect to these properties. The theoretical work is supported by an efficient implementation in the PRISM-games model checker and a comprehensive set of case studies.

The paper *Information-flow control on ARM and POWER multicore processors*, by Graeme Smith, Nicholas Coughlin and Toby Murray, presents a comprehensive Hoare logic for verifying information-flow properties on ARM and POWER multicore processors. The authors describe a rely-guarantee Hoare logic for weak memory models for information-flow security in the presence of dynamic, value-dependent security classifications where the security classification of some variables (controlled variables) can depend on the values of other variables (control variables). Control variables must be low-security. The logic is compositional and flow-sensitive, and it is proved sound with respect to an operational semantics of ARM and POWER using Isabelle/HOL and implemented in an automatic symbolic execution tool.

The paper *Static Analysis for Detecting High-Level Races in RTOS Kernels*, by Rekha Pai, Abhishek Singh, Deepak D'Souza, Meenakshi D'Souza and Prathibha Prakash, presents both a technique and a tool for verifying the lack of so-called high-level races in key data structures of operating RTOS kernels. This application domain raises unique challenges for race detection because embedded operating systems often use specialised techniques, such as disabling interrupts and priority orderings among threads and processes, to ensure isolation and atomicity. The authors propose a technique based on static analysis, which identifies disjoint-block patterns for interrupt-driven programs with callback and software interrupts. They evaluate their technique on four concrete RTOS kernels, P-RTOS, TI-RTOS, ChibiOS and FreeRTOS, which validates that their approach can detect the races efficiently. Efficiency is essential for practical application. They show that their technique has a low rate of false positives.

The paper *Abstraction and Subsumption in Modular Verification of C Programs*, by Lennart Beringer and Andrew W. Appel, presents type-theoretic extensions of a proof system for the verification of C programs in the context of the Verified Software Toolchain (VST), which is an impredicative concurrent separation logic for the C language, implemented in the Coq proof assistant, and proved sound in Coq. The authors expand VST with specification mechanisms and automation features for abstraction, guided by type-theoretic notions and principles. The extensions include existential abstraction for handling function pointers, a novel formalisation of subtyping as well as an integration of intersection types. This brings modularity principles of modern software engineering to concrete program verification.

We are very grateful to Formal Methods in System Design for supporting this special issue, and in particular to the current Editor-in-Chief Nir Piterman for his encouragement and personal involvement in completing this special issue and to the Editor-in-Chief Emeritus Daniel Kroening for originally agreeing to publish this special issue. We would like to thank all the authors of the selected papers as well as the reviewers for their contributions in compiling this special issue. The work of all concerned was made especially challenging by the exigencies of the COVID-19 pandemic during 2020 which, amongst other things, caused academics to become experts in online education practically over night. We are very thankful that, in spite of these difficulties, everyone was able to put together this excellent special issue in a timely and cheerful manner.

Pisa and Sydney — Maurice ter Beek
June 2021 — Annabelle McIver