Introduction to the Special Collection from iFM 2022

ROSEMARY MONAHAN, Department of Computer Science, Maynooth University, Ireland
MAURICE H. TER BEEK, Istituto di Scienza e Tecnologie dell'Informazione (ISTI), Consiglio Nazionale delle Ricerche (CNR), Italy

This special collection arose from the 17th International Conference on integrated Formal Methods (iFM) held in beautiful Lugano, Switzerland, hosted by the Software Institute of USI Università della Svizzera italiana.

CCS Concepts: • **Software and its engineering → Formal methods;**

Additional Key Words and Phrases: integrated formal methods, theorem proving, static analysis, synthesis

ACM Reference format:
Rosemary Monahan and Maurice H. ter Beek. 2023. Introduction to the Special Collection from iFM 2022. *Form. Asp. Comput.* 35, 3, Article 17 (September 2023), 2 pages.
https://doi.org/10.1145/3622995

This special collection arose from the 17th International Conference on integrated Formal Methods (iFM) held in beautiful Lugano, Switzerland, hosted by the Software Institute of USI Università della Svizzera italiana.

The **integrated Formal Methods (iFM)** conference is an annual event, which targets research in formal approaches that combine different methods for modeling and analysis. In recent years, we have witnessed a proliferation of approaches that integrate several modelling, verification, and simulation techniques, facilitating more versatile and efficient analysis of software-intensive systems. These approaches provide powerful support for the analysis of different functional and non-functional properties of the systems, complex interaction of components of different nature as well as validation of diverse aspects of system behaviour. The iFM conference series is a forum for discussing recent research advances in the development of integrated approaches to formal modelling and analysis. The conference series covers all aspects of the design of integrated techniques, including language design, verification and validation, automated tool support, and the use of such techniques in software engineering practice.

This issue includes papers evolved from some of the best submissions to the 17th International Conference on integrated Formal Methods; iFM 2022 attracted 46 submissions out of which 16 were accepted for presentation. To credit the effort of tool developers, this edition of iFM introduced for the first time EAPLS artefact badging. Seven artefact submissions achieved the available and the functional badges, while two artefacts of particularly good quality were awarded the functional and reusable badge.

Authors' addresses: R. Monahan, Department of Computer Science, Maynooth University, Maynooth, Co. Kildare, Ireland, W23 F2H6; e-mail: rosemary.monahan@mu.ie; M. H. ter Beek, Istituto di Scienza e Tecnologie dell'Informazione (ISTI), Consiglio Nazionale delle Ricerche (CNR), Via G. Moruzzi 1, 56124 Pisa, Italy; e-mail: maurice.terbeek@isti.cnr.it.

0934-5043/2023/09-ART17 $15.00
https://doi.org/10.1145/3622995

A total of four papers from those accepted for presentation were selected to be significantly extended to journal-length papers; all four papers had received EAPLS artefact badges. Three of the selections appear here. All papers underwent a thorough review process, with at least three reviews per paper and requiring two to three rounds of revision.

Formal Specification and Verification of JDK's Identity Hash Map Implementation by Martin de Boer, Stijn de Gouw, Jonas Klamroth, Christian Jung, Mattias Ulbrich and Alexander Weigl, presents the first case study of the *IdentityHashMap* class in the Java JDK. The authors specify its behaviour using the **Java Modeling Language (JML)** and prove correctness for the main insertion and lookup methods with KeY, a semi-interactive theorem prover for JML-annotated Java programs. The authors show that unit testing and bounded model checking can be leveraged to find a suitable specification more quickly. They investigate where the bottlenecks in the verification of hash maps lie for KeY by comparing the required automatic proof effort for different hash map implementations and draw conclusions for the choice of hash map implementations regarding their verifiability.

Bit-Vector Typestate Analysis by Alen Arslanagić, Pavle Subotić and Jorge A. Pérez is concerned with static analyses based on typestates, which is important in certifying correctness of code contracts. Such analyses rely on **Deterministic Finite Automata (DFAs)** to specify properties of an object. The authors focus on the analysis of contracts in low-latency environments, where many useful contracts are impractical to codify as DFAs and/or the size of their associated DFAs leads to sub-par performance. To address this bottleneck, they present a lightweight typestate analyzer, based on an expressive specification language that can succinctly specify code contracts. By implementing it in the static analyzer Infer, they demonstrate considerable performance and usability benefits when compared to existing techniques. A central insight is to rely on a sub-class of DFAs with efficient bit-vector operations.

Kaki: Efficient Concurrent Update Synthesis for SDN by Nicklas S. Johansen, Lasse B. Kær, Andreas L. Madsen, Kristian Ø. Nielsen, Jiří Srba and Rasmus G. Tollund discusses modern computer networks based on the **software-defined networking (SDN)** paradigm as they are becoming increasingly complex and require frequent configuration changes. As a result, it is essential that forwarding policies are preserved not only before and after the configuration update but also at any moment during the inherently distributed execution of such an update. The authors present Kaki, a Petri game based approach for automatic synthesis of switch batches which can be updated in parallel without violating a given (regular) forwarding policy like waypointing or service chaining. Kaki guarantees to find the minimum number of concurrent batches and it supports both splittable and nonsplittable flow forwarding. In order to achieve an optimal performance, they introduce two novel optimisation techniques based on static analysis: decomposition into independent subproblems and identification of switches that can be collectively updated in the same batch. These techniques considerably improve the performance of their tool, relying on TAPAAL's verification engine for Petri games as its backend. Experiments on a large benchmark of real networks from the Internet Topology Zoo database demonstrate that Kaki outperforms the state-of-the-art tools Netstack and FLIP.

ACKNOWLEDGMENTS

We are very grateful to Formal Aspects of Computing for supporting this special collection, and in particular to the Editor-in-Chief James Woodcock, Managing Editor John Cooke and Journal Administrator Rebecca Malone for their assistance and advice. We would like to thank all the authors of the selected papers as well as the reviewers for their contributions in compiling this special collection.