Software and Systems Modeling (2024) 23 285–287
https://doi.org/10.1007/s10270-024-01168-y

GUEST EDITORIAL

Guest editorial for the special section on SEFM 2020 and 2021

Frank S. de Boer[1] · **Antonio Cerone**[2]

Received: 11 March 2024 / Accepted: 12 March 2024

1 Introduction

The main objective of the International Conference on Software Engineering and Formal Methods (SEFM) is to bring together practitioners and researchers from academia, industry, and government, to advance the state of the art in formal methods, to help in their large-scale application in the software industry, and to encourage their integration with other practical software engineering methods.

In general the papers presented at the SEFM conferences are selected on the basis of a rigorous single-blind peer review process by a program committee of international renowned experts in the field. This special section consists of a selection of papers presented at SEFM 2020 and 2021, the 18th and 19th editions of SEFM, which have been held virtually during the COVID pandemic. The 18th edition was virtually hosted by the national research institute for mathematics and computer science (CWI) in the Netherlands during 14–17 September 2020. The 19th edition was jointly organised in virtual mode by Carnegie Mellon University (US), Nazarbayev University (Kazakhstan) and University of York (UK) during 6–10 December 2021.

Each article was subject to the full SoSyM review cycle and authors received anonymous feedback in two or more rounds of reviewing from two to three reviewers who are experts in the field. As a result, each article has been thoroughly revised and substantially extended compared to its conference version.

We thank the anonymous reviewers for their valuable comments which in general were the basis for considerable further improvements. We also thank Martin Schindler for his persistent assistance without which this special section would not have been possible.

✉ Frank S. de Boer
F.S.de.Boer@cwi.nl

Antonio Cerone
antonio.cerone@nu.edu.kz

[1] Centrum Wiskunde & Informatica, Amsterdam, NL

[2] Department of Computer Science, School of Engineering and Digital Sciences (SEDS), Nazarbayev University, Astana, Kazakhstan

2 Selected papers

We briefly outline the papers selected for this special section.

A framework for embedded software portability and verification: from formal models to low-level code by Renata Martins Gomes, Bernhard Aichernig, and Marcel Baunach.

With the growing demand for dependability and the increasing hardware diversity in systems like the Internet of Things (IoT), new software development approaches are essential. This includes rigorous methods for verifying and automatically porting Real-Time Operating Systems (RTOS) to various devices.

This paper discusses a hardware-specific part of a kernel model formalized in Event-B, ensuring correctness according to the specification. Since hardware details are only added in subsequent modeling stages, most of the model and proofs can be reused for multiple targets. In a proof of concept, the generic model is refined for two different architectures, also ensuring safety and liveness properties and allowing for automatic low-level code generation from the model.

A lightweight approach to nontermination inference using Constrained Horn Clauses by Bishoksan Kafle, Graeme Gange, Peter Schachte, Harald Søndergaard, and Peter J. Stuckey.

Nontermination is an unwanted program property for some software systems, and a safety property for other systems. In either case, automated discovery of preconditions for nontermination is of interest. This paper introduces a fast lightweight nontermination analyser NtHorn which is able to deduce non-trivial sufficient conditions for nontermination. Constrained Horn Clauses (CHCs) established techniques for CHC program transformation and abstract interpretation then can be exploited for the purpose of nontermination analysis. NtHorn is comparable in power to the state-of-the-art nonter-

mination analysis tools, as measured on standard competition benchmark suites (consisting of integer manipulating programs), while typically solving problems faster by one order of magnitude.

Quantitative modelling and analysis of BDI agents by Blair Archibald, Muffy Calder, Michele Sevegnani, and Mengwei Xu.

Belief-Desire-Intention (BDI) agents are a popular agent architecture. This paper extends the Conceptual Agent Notation (Can)-a BDI programming language with advanced features such as failure recovery and declarative goals-to include probabilistic action outcomes, e.g. to reflect failed actuators, and probabilistic policies, e.g. for probabilistic plan and intention selection. The extension is encoded in Milner's bigraphs. This encoding allows for investigation and comparison of the probability of success (intention completion) under different probabilistic outcomes and plan/event/intention selection strategies. An application to a smart manufacturing use case shows that plan selection has limited effect compared with intention selection, and that the impact of action failures can be marginal-even when failure probabilities are large-due to the agent making smarter choices.

Lazy model checking for recursive state machines by Clemens Dubslaff, Patrick Wienhöft, and Ansgar Fehnker.

Recursive state machines (RSMs) are state-based models for procedural programs with wide-ranging applications in program verification and interprocedural analysis. This paper introduces a new model-checking algorithm for RSMs and requirements in computation tree logic (CTL) that exploits the compositional structure of RSMs by ternary model checking in combination with a lazy evaluation scheme. Implementations of the new model-checking algorithms are evaluated on randomized scalability benchmarks and on an interprocedural data-flow analysis of Java programs, showing both practical applicability and significant speedups in comparison to state-of-the-art model-checking tools for procedural programs.

P-stable abstractions of hybrid systems by Anna Becchi, Alessandro Cimatti, and Enea Zaffanella.

This paper introduces an algorithm for synthesizing P-stable abstractions which characterize the transitions between the stability regions of dynamical systems in response to external inputs. It discusses the representational power of P-stable abstractions, which provide a high-level account of the behavior of the system with respect to stability, and evaluates the effectiveness of the algorithm in synthesizing P-stable abstractions for significant systems.

Fairness, assumptions, and guarantees for extended bounded response LTL+P synthesis by Alessandro Cimatti, Luca Geatti, Nicola Gigante, Angelo Montanari, and Stefano Tonetta.

Realizability and reactive synthesis from temporal logics are fundamental problems in formal verification. The complexity of these problems for Linear Temporal Logic with Past (LTL+P) led to the identification of fragments with lower complexities and simpler algorithms. Recently, the logic of Extended Bounded Response LTL+P (LTLEBR+P for short) has been introduced. It allows one to express safety languages definable in LTL+P and it is provided with an efficient, fully symbolic algorithm for reactive synthesis.

This paper introduces and investigates the expressiveness of an extension of LTLEBR+P with fairness conditions, assumptions, and guarantees that allows the expression of properties beyond the safety fragment, retaining the efficiency of LTLEBR+P in practice. The paper further includes a fully symbolic algorithm for the realizability problem for this extension. To ensure soundness and completeness of the algorithm, the paper introduces a general framework for safety reductions in the context of realizability of (fragments of) LTL+P. The experimental evaluation shows promising results.

Counterexample classification by Cole Vick, Eunsuk Kang, and Stavros Tripakis.

In model checking, when a model fails to satisfy the desired specification, a typical model checker provides a counterexample that illustrates how the violation occurs. This paper introduces counterexample classification. The goal of this classification is to cover the space of all counterexamples into a finite set of counterexample classes, each of which describes a distinct type of violating behavior for the given specification. These classes are then presented as a summary of possible violating behaviors in the system, freeing the user from manually having to inspect or analyze numerous counterexamples to extract the same information. A prototype implementation of the proposed technique is described on top of an existing formal modeling and verification tool, the Alloy Analyzer, and is further evaluated with respect to the effectiveness of the technique on case studies involving the well-known Needham-Schroeder and TCP protocols with promising results.

Analyzing the impact of human errors on interactive service robotic scenarios via formal verification by Livia Lestingi,

Andrea Manglaviti, Davide Marinaro, Luca Marinello, Mehrnoosh Askarpour, Marcello Bersani, and Matteo Rossi.

Developing robotic applications with human-robot interaction for the service sector raises a plethora of challenges. This paper presents a model-driven framework for developing interactive service robotic scenarios, allowing designers to model the interactive scenario, estimate its outcome, deploy the application, and smoothly reconfigure it. The core of the framework is a formal model of the agents at play—the humans and the robots—and the robotic mission under analysis, which is subject to Statistical Model Checking to estimate the mission's outcome. The formal model incorporates a formalization of different human erroneous behaviors' phenotypes, whose likelihood can be tuned while configuring the scenario. Through scenarios inspired by the healthcare setting, the evaluation highlights how different configurations of erroneous behavior impact the verification results and guide the designer towards the mission design that best suits their needs.

Active model learning of stochastic reactive systems by Martin Tappler, Edi Muskardin, Bernhard Aichernig, and Ingo Pill.

Black-box systems are inherently hard to verify. Many verification techniques, like model checking, require formal models as a basis. Active automata learning allows to automatically infer formal models from system interactions. This paper presents a new approach to efficiently learn models of stochastic reactive systems. It extends L*-based learning for Markov decision processes to stochastic Mealy machines. An evaluation shows that the proposed optimizations and adaptations to stochastic Mealy machines can reduce learning costs by an order of magnitude while improving the accuracy of learned models.