Lecture Notes in Computer Science 8702

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
   Lancaster University, UK
Takeo Kanade
   Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
   University of Surrey, Guildford, UK
Jon M. Kleinberg
   Cornell University, Ithaca, NY, USA
Alfred Kobsa
   University of California, Irvine, CA, USA
Friedemann Mattern
   ETH Zurich, Switzerland
John C. Mitchell
   Stanford University, CA, USA
Moni Naor
   Weizmann Institute of Science, Rehovot, Israel
Oscar Nierstrasz
   University of Bern, Switzerland
C. Pandu Rangan
   Indian Institute of Technology, Madras, India
Bernhard Steffen
   TU Dortmund University, Germany
Demetri Terzopoulos
   University of California, Los Angeles, CA, USA
Doug Tygar
   University of California, Berkeley, CA, USA
Gerhard Weikum
   Max Planck Institute for Informatics, Saarbruecken, Germany
Software Engineering and Formal Methods

12th International Conference, SEFM 2014
Grenoble, France, September 1-5, 2014
Proceedings
Preface

This volume contains the papers presented at SEFM 2014, the 12th International Conference on Software Engineering and Formal Methods, which was held during September 3–5, 2014, in Grenoble, France. The aim of the conference is to bring together practitioners and researchers from academia, industry, and government to advance the state of the art in formal methods, to facilitate their uptake in the software industry, and to encourage their integration with practical engineering methods. SEFM 2014 was organized by Inria and supported by Grenoble INP, University Joseph-Fourier, LIG, and CNRS.

SEFM 2014 received 138 abstracts and 106 full submissions. Papers underwent a rigorous review process, and each paper received 3 reviews. After a careful discussion phase, the international Program Committee decided to select 23 research papers and 6 tool papers. These papers cover a wide variety of topics such as program correctness, testing, static analysis, theorem proving, model checking, and automata learning. They also address a wide range of systems, including component-based, real-time, embedded, adaptive, and multi-agent.

The conference featured 3 invited talks by Patrice Godefroid (Microsoft Research, USA), Joost-Pieter Katoen (RWTH Aachen University, Germany), and Xavier Leroy (Inria, France). These talks discussed the software engineering challenges of developing trusted formal tools that scale to the size of industrial systems. Extended abstracts of the invited talks can be found in this volume.

Five international workshops were colocated with SEFM 2014: the 1st Workshop on Human-Oriented Formal Methods (HOFM 2014), the 3rd International Symposium on Modelling and Knowledge Management Applications: Systems and Domains (MoKMaSD 2014), the 8th International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2014), the 1st Workshop on Safety and Formal Methods (SaFoMe 2014), and the 4th Workshop on Formal Methods in the Development of Software (WS-FMDS 2014).

We thank the local Organizing Committee (Sophie Azzaro, Wassila Bouhadji, Myriam Etienne, Vanessa Peregrin) for taking care of the local arrangements, the Steering Committee chair Antonio Cerone and the Conference Chair Radu Mateescu for their guidance, the workshop chairs (Carlos Canal, Marc Frappier, and Akram Idani) for supervizing the workshops organization, Rim Abid for negotiating financial support, Lina Ye for acting as publicity chair, and Hugues Evrard for acting as Web master. We assembled an exciting technical program that would not have been possible without the excellent work of the Program Committee and external reviewers. Last, but not least, we thank the authors of all submitted papers, our invited speakers, and all the participants (speakers or
not) of the conference in Grenoble. All these people contributed to the success of the 2014 edition of SEFM. Finally, EasyChair made our work as program chairs substantially easier.

June 2014

Dimitra Giannakopoulou
Gwen Salaün
Organization

Program Committee

Wolfgang Ahrendt  
Chalmers University of Technology, Sweden

Bernhard K. Aichernig  
TU Graz, Austria

Dalal Alrajeh  
Imperial College London, UK

Farhad Arbab  
CWI and Leiden University, The Netherlands

Luis Barbosa  
Universidade do Minho, Portugal

Howard Barringer  
The University of Manchester, UK

Domenico Bianculli  
SnT Centre - University of Luxembourg, Luxembourg

Jonathan P. Bowen  
Birmingham City University, UK

Mario Bravetti  
University of Bologna, Italy

Tevfik Bultan  
University of California at Santa Barbara, USA

Hung Dang Van  
UET, Vietnam National University

Francisco Durán  
University of Málaga, Spain

George Eleftherakis  
The University of Sheffield International Faculty, CITY College, UK

José Luiz Fiadeiro  
Royal Holloway, University of London, UK

Mamoun Filali-Amine  
IRIT, France

Marc Frappier  
University of Sherbrooke, Canada

Martin Fränzle  
Carl von Ossietzky Universität Oldenburg, Germany

Hubert Garavel  
Inria Rhone-Alpes/VASY, France

Dimitra Giannakopoulou  
NASA Ames, USA

Stefania Gniesi  
ISTI-CNR, France

Klaus Havelund  
Jet Propulsion Laboratory, California Institute of Technology, USA

Rob Hierons  
Brunel University, UK

Mike Hinchey  
Lero, The Irish Software Engineering Research Centre, Ireland

Falk Howar  
CMU/NASA Ames, USA

Florentin Ipate  
University of Bucharest, Romania

Martin Leucker  
University of Lübeck, Germany

Peter Lindsay  
The University of Queensland, Australia

Antónia Lopes  
University of Lisbon, Portugal

Mercedes Merayo  
Universidad Complutense de Madrid, Spain

Stephan Merz  
Inria Lorraine, France

Mizuhito Ogawa  
Advanced Institute of Science and Technology, Japan
VIII Organization

Fernando Orejas  
Gordon Pace  
David Parker  
Corina Pasareanu  
Anna Philippou  
Sanjiva Prasad  
Jakob Rehof  
Leila Ribeiro  

Bernhard Rumpe  
Gwen Salaün  
Augusto Sampaio  
Gerardo Schneider  
Marjan Sirjani  
Matt Staats  
Martin Steffen  
Jing Sun  
Jun Sun  

Serdar Tasiran  
Massimo Tivoli  
Dongmei Zhang  
Jianjun Zhao  

Additional Reviewers

Alkhalaf, Muath  
Aydin, Abdulbaki  
Bessai, Jan  
Bocic, Ivan  
Bodeveix, Jeanpauk  
Bonenfant, Armelle  
Bove, Ana  
Colombo, Christian  
Cámara Moreno, Javier  
D’Souza, Deepak  
Dang, Duc-Hanh  
Decker, Normann  
Deharbe, David  
Dudenhofner, Andrej  
Duflot, Marie  
Düdder, Boris  
Fakih, Maher  
Fantechi, Alessandro  
Ferrari, Alessio  

Garnacho, Manuel  
Gerwinn, Sebastian  
Greifenberg, Timo  
Habel, Annegret  
Hojjat, Hossein  
Jafari, Ali  
Jaghouri, Mahdi  
Johansson, Moa  
Kaiser, Steffi  
Khakpour, Narges  
Khosravi, Ramtin  
Kromodimoeljo, Sentot  
Kuru, Ismail  
Kühn, Franziska  
Lefticaru, Raluca  
Li, Qin  
Lima, Lucas  
Lin, Ziyi  
Lindt, Achim
Invited Papers
Abstract. Tool-assisted verification of critical software has great potential but is limited by two risks: unsoundness of the verification tools, and miscompilation when generating executable code from the sources that were verified. A radical solution to these two risks is the deductive verification of compilers and verification tools themselves. In this invited talk, I describe two ongoing projects along this line: CompCert, a verified C compiler, and Verasco, a verified static analyzer based on abstract interpretation.
500 Machine-Years of Software Model Checking and SMT Solving

Patrice Godefroid
Microsoft Research
pg@microsoft.com

Abstract. I will report on our experience running SAGE for over 500-machine years in Microsoft’s security testing labs. SAGE is a whitebox fuzzing tool for security testing. It performs symbolic execution dynamically at the binary (x86) level, generates constraints on program inputs, and solves those constraints with an SMT solver in order to generate new inputs to exercise new program paths or trigger security vulnerabilities (like buffer overflows). This process is repeated using novel state-space exploration techniques that attempt to sweep through all (in practice, many) feasible execution paths of the program while checking simultaneously many properties. This approach thus combines program analysis, testing, model checking and automated theorem proving (constraint solving).

Since 2009, SAGE has been running 24/7 on average 100+ machines automatically “fuzzing” hundreds of applications. This is the largest computational usage ever for any SMT solver, with over 4 billion constraints processed to date. In the process, SAGE found many new security vulnerabilities (missed by blackbox fuzzing and static program analysis) and was credited to have found roughly one third of all the bugs discovered by file fuzzing during the development of Microsoft’s Windows 7, saving millions of dollars by avoiding expensive security patches to nearly a billion PCs.

In this talk, I will present the SAGE project, highlight connections with program verification, and discuss open research challenges.

This is joint work with Michael Levin, David Molnar, Ella Bounimova, and other contributors.
Probabilistic model checking – the verification of models incorporating random phenomena – has enjoyed a rapid increase of interest. Thanks to the availability of mature tool support and efficient verification algorithms, probabilistic model checking has been successfully applied to case studies from various areas, such as randomized (distributed) algorithms, planning and AI, security, hardware, stochastic scheduling, reliability analysis, and systems biology [9]. In addition, model-checking techniques have been adopted by mainstream model-based performance and dependability tools as effective analysis means. Probabilistic model checking can thus be viewed as a viable alternative and extension to traditional model-based performance analysis [1].

Typical properties that are checked are quantitative reachability objectives, such as: does the probability to reach a certain set of goal states (by avoiding illegal states) exceed $\frac{1}{2}$? Extra constraints can be incorporated as well that e.g., require the goal to be reached within a certain number of transitions, within a certain budget, or within a real-time deadline. For models exhibiting both transition probabilities and non-determinism, maximal and minimal probabilities are considered. Intricate combinations of numerical (or simulation) techniques for Markov chains, optimization algorithms, and traditional CTL or LTL model-checking algorithms result in simple, yet very efficient verification procedures [2, 10]. Verifying time-bounded reachability properties on continuous-time models of tens of millions of states usually is a matter of seconds. Using symbolic representation techniques such as multi-terminal BDDs, much larger systems can be treated efficiently as well. A gentle introduction can be found in [5].

Like in the traditional setting, probabilistic model checking suffers from the curse of dimensionality: the number of states grows exponentially in the number of system components and cardinality of data domains. This hampers the analysis of real-life systems such as biological models involving thousands of molecules [12], and software models of on-board aerospace systems that incorporate probabilistic error models of various system components on top of the “nominal” system behaviour [3].

This talk considers the theory and practice of aggressive abstraction of discrete-time and continuous-time Markov models. Our abstraction technique is based on a partitioning of the concrete state space that is typically much coarser than e.g., bisimulation minimisation. We exploit three-valued abstraction [4] in which a temporal logic formula evaluates to either true, false, or indefinite. In this setting, abstraction is conservative for both positive and negative verification results; in our setting this means that the analysis yields bounds
on the desired probability measures. If the verification of the abstract model yields an indefinite answer (don't know), no conclusion on the validity in the concrete model can be drawn. States in abstract Markov models are groups of concrete states and transitions are either equipped with intervals or modeled as non-deterministic choices. The resulting abstraction is shown to preserve a simulation relation: concrete states are simulated by their corresponding abstract ones.

We present the theoretical foundations of aggressive abstraction of Markov models [6] and show how this technique can be applied in a compositional way. This enables the component-wise abstraction of large models [7, 11]. We present two case studies, one from systems biology and one from queueing theory, illustrating the power of this technique. This includes strategies of which states to group, verification times of the abstract models, and the resulting accuracies of the quantitative results. We show that this abstraction technique enables the verification of models larger than $10^{250}$ states by abstract models of a few hundred thousands states while obtaining results with an accuracy of $10^{-6}$ [8].

Acknowledgement. This work is funded by the EU FP7-projects SENSATION and MEALS, the STW project ArRangeer, and the Excellence Program of the German Federal Government.

References


# Table of Contents

## Invited Papers

Formal Proofs of Code Generation and Verification Tools (Abstract) ....... XIII  
   *Xavier Leroy*

500 Machine-Years of Software Model Checking and SMT Solving  
(Abstract) ...................................................... XIV  
   *Patrice Godefroid*

Model Checking Gigantic Markov Models (Abstract) ............... XV  
   *Joost-Pieter Katoen*

Formal Proofs of Code Generation and Verification Tools  
(Full Paper) ..................................................... 1  
   *Xavier Leroy*

## Program Verification

Lightweight Program Construction and Verification Tools  
in Isabelle/HOL .................................................. 5  
   *Alasdair Armstrong, Victor B.F. Gomes, and Georg Struth*

Completeness of Separation Logic with Inductive Definitions for  
Program Verification ............................................. 20  
   *Makoto Tatsuta and Wei-Ngan Chin*

A Thread-Safe Library for Binary Decision Diagrams ............... 35  
   *Alberto Lovato, Damiano Macedonio, and Fausto Spoto*

Effect-Polymorphic Behaviour Inference for Deadlock Checking .... 50  
   *Ka I Pun, Martin Steffen, and Volker Stolz*

## Testing

Synthesizing Parameterized Unit Tests to Detect Object Invariant  
Violations.......................................................... 65  
   *Maria Christakis, Peter Müller, and Valentin Wütholz*

Formalizing DSL Semantics for Reasoning and Conformance Testing... 81  
   *Sarmen Keshishzadeh and Arjan J. Mooij*

Test Suite Completeness and Partial Models .......................... 96  
   *Adilson Luiz Bonifacio and Arnaldo Vieira Moura*
<table>
<thead>
<tr>
<th>Component-Based Systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated Error-Detection and Repair for Compositional Software Specifications</td>
<td>Dalal Alrajeh and Robert Craven</td>
</tr>
<tr>
<td></td>
<td>111</td>
</tr>
<tr>
<td>A General Framework for Architecture Composability</td>
<td>Paul Attie, Eduard Baranov, Simon Bliudze, Mohamad Jaber,</td>
</tr>
<tr>
<td></td>
<td>and Joseph Sifakis</td>
</tr>
<tr>
<td></td>
<td>128</td>
</tr>
<tr>
<td>Trace Checking of Metric Temporal Logic with Aggregating Modalities</td>
<td>Domenico Bianculli, Carlo Ghezzi, and Srdan Krstić</td>
</tr>
<tr>
<td>Using MapReduce</td>
<td>144</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Real-Time and Embedded Systems</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Increasing Consistency in Multi-site Data Stores: Megastore-CGC and Its Formal Analysis</td>
<td>Jon Grov and Peter Csaba Ölveczky</td>
</tr>
<tr>
<td></td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>175</td>
</tr>
<tr>
<td>Formal Verification of Discrete-Time MATLAB/Simulink Models Using Boogie</td>
<td>Robert Reicherdt and Sabine Glesner</td>
</tr>
<tr>
<td></td>
<td>190</td>
</tr>
<tr>
<td>A Formal Model for Constraint-Based Deployment Calculation and Analysis for Fault-Tolerant Systems</td>
<td>Klaus Becker, Bernhard Schätz, Michael Armbruster, and Christian Buckl</td>
</tr>
<tr>
<td></td>
<td>205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model Checking and Automata Learning</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimising the ProB Model Checker for B Using Partial Order Reduction</td>
<td>Ivaylo Dobrikov and Michael Leuschel</td>
</tr>
<tr>
<td></td>
<td>220</td>
</tr>
<tr>
<td>Rapid Prototyping of a Semantically Well Founded Circus Model Checker</td>
<td>Alexandre Mota, Adalberto Farias, André Didier, and Jim Woodcock</td>
</tr>
<tr>
<td></td>
<td>235</td>
</tr>
<tr>
<td>Learning Extended Finite State Machines</td>
<td>Sofia Cassel, Falk Howar, Bengt Jonsson, and Bernhard Steffen</td>
</tr>
<tr>
<td></td>
<td>250</td>
</tr>
<tr>
<td>Tool Papers</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Translating Event-B Machines to Database Applications</td>
<td>265</td>
</tr>
<tr>
<td>Qi Wang and Tim Wahls</td>
<td></td>
</tr>
<tr>
<td>IKOS: A Framework for Static Analysis Based on Abstract Interpretation</td>
<td>271</td>
</tr>
<tr>
<td>Guillaume Brat, Jorge A. Navas, Nija Shi, and Arnaud Venet</td>
<td></td>
</tr>
<tr>
<td>A Toolset for Support of Teaching Formal Software Development</td>
<td>278</td>
</tr>
<tr>
<td>Štefan Korečko, Ján Sorád, Zuzana Dudláková, and Branislav Sobota</td>
<td></td>
</tr>
<tr>
<td>Execution and Verification of UML State Machines with Erlang</td>
<td>284</td>
</tr>
<tr>
<td>Ricardo J. Rodríguez, Lars-Åke Fredlund, Ángel Herranz, and Julio Mariño</td>
<td></td>
</tr>
<tr>
<td>A Tool for Verifying Dynamic Properties in B</td>
<td>290</td>
</tr>
<tr>
<td>Fama Diagne, Amel Mammar, and Marc Frappier</td>
<td></td>
</tr>
<tr>
<td>WeVerca: Web Applications Verification for PHP</td>
<td>296</td>
</tr>
<tr>
<td>David Hauzar and Jan Kofroň</td>
<td></td>
</tr>
<tr>
<td>Program Correctness</td>
<td></td>
</tr>
<tr>
<td>More Flexible Object Invariants with Less Specification Overhead</td>
<td>302</td>
</tr>
<tr>
<td>Stefan Huster, Patrick Heckeler, Hanno Eichelberger, Jürgen Ruf, Sebastian Burg, Thomas Kropf, and Wolfgang Rosenstiel</td>
<td></td>
</tr>
<tr>
<td>Verified Functional Iterators Using the FoCaLiZe Environment</td>
<td>317</td>
</tr>
<tr>
<td>Catherine Dubois and Renaud Rioboo</td>
<td></td>
</tr>
<tr>
<td>Tool Support for Teaching Hoare Logic</td>
<td>332</td>
</tr>
<tr>
<td>Tadeusz Sznuk and Alekys Schubert</td>
<td></td>
</tr>
<tr>
<td>Adaptive and Multi-Agent Systems</td>
<td></td>
</tr>
<tr>
<td>A Two-Phase Static Analysis for Reliable Adaptation</td>
<td>347</td>
</tr>
<tr>
<td>Pierpaolo Degano, Gian-Luigi Ferrari, and Letterio Galletta</td>
<td></td>
</tr>
<tr>
<td>Formal Modelling and Verification of Cooperative Ant Behaviour in Event-B</td>
<td>363</td>
</tr>
<tr>
<td>Linas Laibinis, Elena Troubitsyna, Zeineb Graja, Frédéric Migeon, and Ahmed Hadj Kacem</td>
<td></td>
</tr>
<tr>
<td>A Formal Privacy Policy Framework for Social Networks</td>
<td>378</td>
</tr>
<tr>
<td>Raúl Pardo and Gerardo Schneider</td>
<td></td>
</tr>
<tr>
<td>Author Index</td>
<td>393</td>
</tr>
</tbody>
</table>