## Lecture Notes in Computer Science

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

#### Editorial Board

David Hutchison Lancaster University, Lancaster, UK Takeo Kanade Carnegie Mellon University, Pittsburgh, PA, USA Josef Kittler University of Surrey, Guildford, UK Jon M. Kleinberg Cornell University, Ithaca, NY, USA Friedemann Mattern ETH Zurich, Zurich, Switzerland John C. Mitchell Stanford University, Stanford, CA, USA Moni Naor Weizmann Institute of Science, Rehovot, Israel C. Pandu Rangan Indian Institute of Technology, Madras, India Bernhard Steffen TU Dortmund University, Dortmund, 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, Saarbrücken, Germany More information about this series at http://www.springer.com/series/7408

# Verified Software

Theories, Tools, and Experiments

8th International Conference, VSTTE 2016 Toronto, ON, Canada, July 17–18, 2016 Revised Selected Papers



*Editors* Sandrine Blazy IRISA, University of Rennes 1 Rennes France

Marsha Chechik Department of Computer Science University of Toronto Toronto, ON Canada

ISSN 0302-9743 ISSN 1611-3349 (electronic) Lecture Notes in Computer Science ISBN 978-3-319-48868-4 ISBN 978-3-319-48869-1 (eBook) DOI 10.1007/978-3-319-48869-1

Library of Congress Control Number: 2016956493

LNCS Sublibrary: SL2 - Programming and Software Engineering

#### © Springer International Publishing AG 2016

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature The registered company is Springer International Publishing AG The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

### Preface

This volume contains the papers presented at the 8th International Conference on Verified Software: Theories, Tool and Experiments (VSTTE), which was held in Toronto, Canada, during July 17–18, 2016, co-located with the 28th International Conference on Computer-Aided Verification. The final version of the papers was prepared by the authors after the event took place, which permitted them to take feedback received at the meeting into account. VSTTE originated from the Verified Software Initiative (VSI), which is an international initiative directed at the scientific challenges of large-scale software verification. The inaugural VSTTE conference was held at ETH Zurich in October 2005, and was followed by VSTTE 2008 in Toronto, VSTTE 2010 in Edinburgh, VSTTE 2012 in Philadelphia, VSTTE 2013 in Menlo Park, VSTTE 2014 in Vienna, and VSTTE 2015 in San Francisco. The goal of the VSTTE conference is to advance the state of the art through the interaction of theory development, tool evolution, and experimental validation.

The call for papers for VSTTE 2016 solicited submissions describing large-scale verification efforts that involve collaboration, theory unification, tool integration, and formalized domain knowledge. We were especially interested in papers describing novel experiments and case studies evaluating verification techniques and technologies. We welcomed papers describing education, requirements modeling, specification languages, specification/verification, formal calculi, software design methods, automatic code generation, refinement methodologies, compositional analysis, verification tools (e.g., static analysis, dynamic analysis, model checking, theorem proving), tool integration, benchmarks, challenge problems, and integrated verification environments. We received 21 submissions. Each submission was reviewed by at least three members of the Program Committee. The committee decided to accept 12 papers for presentation at the conference. The program also included six invited talks, given by Zachary Tatlock (Washington), Mark Lawford (McMaster), Kristin Yvonne Rozier (Iowa State), Michael Tautschnig (Amazon), and Oksana Tkachuk (NASA Ames). The volume includes abstracts or full-paper versions of some of these talks.

We would like to thank the invited speakers and all submitting authors for their contribution to the program. We are very grateful to our general chair, Temesghen Kahsai, for his tremendous help with organizing this event. We also thank Azadeh Farzan (CAV PC co-chair) and Zak Kinsaid (CAV Workshops chair) for logistical support, and to Natarajan Shankar for his vision for this year's VSTTE and other events in this series. Last but definitely not least, we thank the external reviewers and the Program Committee for their reviews and their help in selecting the papers that appear in this volume. This volume was generated with the help of EasyChair.

September 2016

Marsha Chechik Sandrine Blazy

## Organization

#### **Program Committee**

June Andronick Frédéric Besson Nikolaj Bjorner Sandrine Blazy Marsha Chechik Ernie Cohen Deepak D'Souza Jean-Christophe Filliatre Vijay Ganesh Arie Gurfinkel William Harris Temesghen Kahsai Vladimir Klebanov Rustan Leino Tiziana Margaria David Naumann Nadia Polikarpova Kristin Yvonne Rozier Natarajan Shankar Natasha Sharygina Richard Trefler Michael Whalen Naijun Zhan

NICTA and UNSW, Australia Inria, France Microsoft Research, USA **IRISA**, France University of Toronto, Canada Amazon, USA Indian Institute of Science, Bangalore, India **CNRS**, France University of Waterloo, Canada Software Engineering Institute, Carnegie Mellon University, USA Georgia Institute of Technology, USA NASA Ames/CMU, USA Karlsruhe Institute of Technology, Germany Microsoft Research, USA Lero. Ireland Stevens Institute of Technology, USA MIT CSAIL, USA University of Cincinnati, USA SRI International, USA University of Lugano, Switzerland University of Waterloo, Canada University of Minnesota, USA Institute of Software, Chinese Academy of Sciences, China

#### **Additional Reviewers**

Alt, Leonardo Berzish, Murphy Bormer, Thorsten Chen, Mingshuai Fedyukovich, Grigory Graham-Lengrand, Stephane Guelev, Dimitar Hyvärinen, Antti Kuraj, Ivan Marescotti, Matteo Tiwari, Ashish Zhang, Wenhui Zheng, Yunhui Zulkoski, Ed

**Abstracts Short Papers** 

## Advanced Development of Certified OS Kernels

Zhong Shao

Yale University, New Haven, USA

Abstract. Operating System (OS) kernels form the backbone of all system software. They can have a significant impact on the resilience, extensibility, and security of today's computing hosts. We present a new compositional approach [3] for building certifiably secure and reliable OS kernels. Because the very purpose of an OS kernel is to build layers of abstraction over hardware resources, we insist on uncovering and specifying these layers formally, and then verifying each kernel module at its proper abstraction level. To support reasoning about user-level programs and linking with other certified kernel extensions, we prove a strong contextual refinement property for every kernel function, which states that the implementation of each such function will behave like its specification under any kernel/user (or host/guest) context. To demonstrate the effectiveness of our new approach, we have successfully implemented and specified a practical OS kernel and verified its (contextual) functional correctness property in the Coq proof assistant. We show how to extend our base kernel with new features such as virtualization [3], interrupts and device drivers [1], and end-to-end information flow security [2], and how to quickly adapt existing verified layers to build new certified kernels for different domains.

This research is based on work supported in part by NSF grants 1065451, 1319671, and 1521523 and DARPA grants FA8750-12-2-0293 and FA8750-15-C-0082. Any opinions, findings, and conclusions contained in this document are those of the authors and do not reflect the views of these agencies.

#### References

- Chen, H., Wu, X., Shao, Z., Lockerman, J., Gu, R.: Toward compositional verification of interruptible OS kernels and device drivers. In: PLDI 2016: 2016 ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 431–447(2016)
- Costanzo, D., Shao, Z., Gu, R.: End-to-end verification of information-flow security for C and assembly programs. In: PLDI 2016: 2016 ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 648–664 (2016)
- Gu, R., Koenig, J., Ramananandro, T., Shao, Z., Wu, X., Weng, S-C., Zhang, H., Guo. Y.: Deep specifications and certified abstraction layers. In: POPL 2015: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming languages, pp. 595–608 (2015)

## Automating Software Analysis at Large Scale

Michael Tautschnig

Queen Mary University of London, London, UK Amazon Web Services, Ashburn, USA

**Abstract.** Software model checking tools promise to deliver genuine traces to errors, and sometimes even proofs of their absence. As static analysers, they do not require concrete execution of programs, which may be even more beneficial when targeting new platforms. Academic research focusses on improving scalability, yet largely disregards practical technical challenges to make tools cope with real-world code.

At Amazon, both scalability requirements as well as real-world constraints apply. Our prior work analysing more than 25,000 software packages in the Debian/GNU Linux distribution containing more than 400 million lines of C code not only led to more than 700 public bug reports, but also provided a solid preparation for the challenges at Amazon.

## RACE to Build Highly Concurrent and Distributed Systems

Oksana Tkachuk

NASA Ames Research Center, Moffett, USA oksana.tkachuk@nasa.gov

**Abstract.** Instantiating, running, and monitoring highly concurrent and distributed systems presents many challenges. Such systems are prone to: concurrency-related issues (races, deadlocks), communication problems (dropped connections), functional issues (unhandled messages), and scalability (the size of the system grows with the number of communicating components).

This talk will present solutions to the above problems implemented in RACE: Runtime for Airspace Concept Evaluation, designed and developed at NASA Ames Research Center. RACE is a framework for instantiating and running highly concurrent and distributed systems. RACE employs actor programming model, as implemented in the Akka framework. Akka actors communicate through asynchronous messages, do not share state, and process their own messages sequentially. RACE is implemented in the Scala programming language, which improves type safety compared to other JVM languages. RACE includes many building blocks needed to create distributed systems, including actors for exporting, importing, translating, archiving, replaying, and visualizing data.

RACE is being evaluated in the context of building and running simulations for National Airspace System (NAS) at NASA. For example, RACE can be used to get flight and weather data from various FAA servers, process, and visualize it in the NASA's World Wind viewer. However, RACE is an open source, highly-configurable and extensible platform, which makes it suitable for a wide range of applications. RACE source code is available at https://github.com/ NASARace/race. More information can be found on the RACE web site at http://nasarace.github.io/race.

## Contents

Stupid Tool Tricks for Smart Model Based Design	1
Specification: The Biggest Bottleneck in Formal Methods and Autonomy <i>Kristin Yvonne Rozier</i>	8
Order Reduction for Multi-core Interruptible Operating Systems Jonas Oberhauser	27
Producing All Ideals of a Forest, Formally (Verification Pearl) Jean-Christophe Filliâtre and Mário Pereira	46
Constructing Semantic Models of Programs with the Software Analysis Workbench	56
Bidirectional Grammars for Machine-Code Decoding and Encoding Gang Tan and Greg Morrisett	73
Automated Verification of Functional Correctness of Race-Free GPU Programs	90
The Matrix Reproved (Verification Pearl)	107
Enabling Modular Verification with Abstract Interference Specifications for a Concurrent Queue	119
Accelerating the General Simplex Procedure for Linear Real Arithmetic via GPUs	129
JavaSMT: A Unified Interface for SMT Solvers in Java	139
Relational Program Reasoning Using Compiler IR Moritz Kiefer, Vladimir Klebanov, and Mattias Ulbrich	149

Resolution in Solving Graph Problems	166
SMT-based Software Model Checking: An Experimental Comparison of Four Algorithms	181
Author Index	199