Lecture Notes in Computer Science 8734

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

Borzoo Bonakdarpour Scott A. Smolka (Eds.)

Runtime Verification

5th International Conference, RV 2014
Toronto, ON, Canada, September 22-25, 2014
Proceedings

Springer

Volume Editors

Borzoo Bonakdarpour
McMaster University
Department of Computing and Software
1280 Main Street West, Hamilton
ON L8S 4L7, Canada
E-mail: borzoo@mcmaster.ca

Scott A. Smolka
State University of New York at Stony Brook
Department of Computer Science
1423 Computer Science, Stony Brook
NY 11794-4400, USA
E-mail: sas@cs.sunysb.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-11163-6 e-ISBN 978-3-319-11164-3
DOI 10.1007/978-3-319-11164-3
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014947359

LNCS Sublibrary: SL 2 – Programming and Software Engineering

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The 14th International Conference on Runtime Verification (RV 2014) was held September 22–25, 2014, at the Fields Institute for Research in Mathematical Sciences on the campus of University of Toronto, Canada. The conference program included invited talks, tutorials, peer-reviewed presentations, and tool demonstrations.

RV started in 2001 as an annual workshop and turned into a conference in 2010. The workshops were organized as satellite events to an established forum, including CAV and ETAPS. The proceedings for RV from 2001 to 2005 were published in the *Electronic Notes in Theoretical Computer Science*. Since 2006, the RV proceedings have been published in Springer's *Lecture Notes in Computer Science*.

RV 2014 was attended by researchers and practitioners from all around the world. The conference program included papers on a wide variety of subjects, such as theoretical aspects of runtime verification, testing, tracing, bug finding, monitoring distributed systems, timed systems, and cyber-physical systems.

We are extremely pleased to have had three excellent invited speakers:

- Jeannette Wing, Vice President and Head of Microsoft Research International, is a leading figure in computer science research, particularly in formal methods, security, and privacy.
- Kevin Driscoll is a fellow at Honeywell Labs with 40 years, experience in safety and security critical systems.
- Assaf Schuster is Professor of Computer Science at the Technion, Israel, and has made significant contributions to monitoring distributed data streams and big data technology.

The conference also included two exciting tutorials:

- Vijay K. Garg (UT-Austin) and Neeraj Mittal (UT-Dallas) gave a tutorial on lattice-theoretic approaches to monitoring distributed systems.
- David Basin (ETH-Zurich) and Felix Klaedtke (NEC Labs, Europe) gave the second tutorial on runtime monitoring and enforcement of security policies.

We would like to extend our deep thanks to the authors of all submitted papers, to the members of the Program Committee, and to the external reviewers for their outstanding job in thoroughly evaluating all submitted papers. RV 2014 received 70 submissions: 57 regular papers, three tool papers, and 10 short papers. Most regular papers were reviewed by five Program Committee members. Tool and short papers were reviewed by three members of the Program Committee, who in the end decided to accept 18 are regular papers, 2 are tool papers, 7 short papers. Most paper discussions were conducted through the EasyChair conference manager. Four papers were discussed over a live conference call.

We would also like to thank the Fields Institute for its generous monetary contribution to the conference, as well as sharing its facility to hold the conference free of charge. We highly appreciate EasyChair for its free service to manage submissions.

Finally, our special thanks go to the incomparable chair of the Steering Committee, Klaus Havelund, for his invaluable help during all stages of organizing RV 2014.

July 2014

Borzoo Bonakdarpour
Scott A. Smolka

Organization

General Chair

Sebastian Fischmeister	University of Waterloo, Canada

Program Chairs

Borzoo Bonakdarpour	McMaster University, Canada
Scott Smolka	Stony Brook University, USA

Tools Track Chair

Ezio Bartocci	TU Wien, Austria

Runtime Monitoring Competition Chairs

Ezio Bartocci	TU Wien, Austria
Borzoo Bonakdarpour	McMaster University, Canada
Ylies Falcone	Université Joseph Fourier, France

Tutorials and Proceedings Chair

Nadia Polikarpova	ETH Zurich, Switzerland

Publicity Chair

Ylies Falcone	Université Joseph Fourier, France

Local Arrangements Chair

Patrick Lam	University of Waterloo, Canada

Program Committee

Gul Agha	University of Illinois at Urbana-Champaign, USA
Thomas Ball	Microsoft Research, USA
Howard Barringer	The University of Manchester, UK
Ezio Bartocci	TU Wien, Austria
David Basin	ETH Zurich, Switzerland

Saddek Bensalem	CEA-Leti, France
Borzoo Bonakdarpour (Co-chair)	McMaster University, Canada
Ivona Brandic	TU Wien, Austria
Marsha Chechik	University of Toronto, Canada
Michael Clarkson	George Washington University, USA
Laura Dillon	Michigan State University, USA
Shlomi Dolev	Ben-Gurion University, Israel
Alastair Donaldson	Imperial College London, UK
Dawson Engler	Stanford University, USA
Ylies Falcone	Université Joseph Fourier, France
Vijay Garg	University of Texas at Austin, USA
Steve Goddard	University of Nebraska-Lincoln, USA
Ganesh Gopalakrishnan	University of Utah, USA
Wolfgang Grieskamp	Google, USA
Radu Grosu	TU Wien, Austria
Klaus Havelund	NASA/JPL, USA
Mats Heimdahl	University of Minnesota, USA
Gerard Holzmann	NASA/JPL, USA
Taylor Johnson	UT-Arlington, USA
Daniel Keren	Haifa University, Israel
Sandeep Kulkarni	Michigan State University, USA
Marta Kwiatkowska	University of Oxford, USA
Insup Lee	University of Pennsylvania, USA
Axel Legay	IRISA/Inria, Rennes, France
Martin Leucker	University of Lübeck, Germany
Leonardo Mariani	University of Milano Bicocca, Italy
Patrick Meredith	University of Illinois at Urbana-Champaign, USA
David Naumann	Stevens Institute of Technology, USA
Samaneh Navabpour	University of Waterloo, Canada
Doron Peled	Bar-Ilan University, Israel
Mauro Pezzè	University of Lugano, Switzerland
Lee Pike	Galois, Inc., USA
Nadia Polikarpova	ETH Zurich, Switzerland
Zvonimir Rakamaric	University of Utah, USA
Grigore Rosu	University of Illinois at Urbana-Champaign, USA
Andrey Rybalchenko	TUM, Germany
Andre Schiper	EPFL, Switzerland
Scott Smolka (Co-chair)	Stony Brook University, USA
Oleg Sokolsky	University of Pennsylvania, USA
Scott Stoller	Stony Brook University, USA
Serdar Tasiran	Koç University, Turkey
Michael W. Whalen	University of Minnesota, USA
Lenore Zuck	University of Illinois at Chicago, USA

Additional Reviewers

Abdellatif, Takoua
Alglave, Jade
Arusoaie, Andrei
Avni, Hillel
Bak, Stanley
Balasubramanian, Bharath
Bardsley, Ethel
Binun, Alexander
Blankenburg, Martin
Chang, Yen-Jung
Charalambides, Minas
Chauhan, Himanshu
Chen, Yu-Fang
Chiang, Wei-Fan Chiang
Chong, Nathan
Chudnov, Andrey
Ciobaca, Stefan
Combaz, Jaques
Creswick, Rogan
Decker, Normann
Estler, Hans-Christian
Farokhi, Soodeh
Feng, Lu
Fernandez, Jean-Claude
Griffith, Dennis
Haran, Arvind
Hicks, Michael
Huang, Jeff
Ivanov, Radoslav
Jaber, Mohamad
Jovanovic, Aleksandra
Kahil, Ramzi Martin
Kandl, Susanne
Kim, Chang Hwan Peter
Klaedtke, Felix
Korthikanti, Vijay Anand
Kuru, Ismail
Kühn, Franziska
Liew, Dan
Lucanin, Drazen
Margaria, Tiziana
Marinovic, Srdjan
Matar, Hassan Salehe
Melnychenko, Oleksandr
Meng, Wenrui
Mutlu, Erdal
Mutluergil, Suha Orhun
Müller, Peter
Nizol, Matthew
Ozkan, Burcu Kulahcioglu
Paoletti, Nicola
Park, Junkil
Pastore, Fabrizio
Porter, Joseph
Ratasich, Denise
Reger, Giles
Rydeheard, David
Santoro, Mauro
Scheffel, Torben
Sedwards, Sean
Selyunin, Konstantin
Sharma, Subodh
Sridhar, Meera
Stümpel, Annette
Thoma, Daniel
Thomson, Paul
Ujma, Mateusz
Wang, Shaohui
Weimer, James
Weiss, Gera
Wickerson, John
Wiltsche, Clemens
Winwood, Simon
Xu, Meng
Zalinescu, Eugen

Invited Talks

Murphy Strikes Again

Kevin Driscoll (Honeywell Labs, USA)

An objective of a conference keynote is to provide some rationale and motivation for the conference: Why are we here? For this conference: Why do Runtime Verification? It must be for applications critical enough to warrant this additional expense to ensure that the application performs adequately in the presence of faults – design faults and hardware faults. There is an interesting link between the latter and the former. In critical applications, there often is a higher density of faults in the fault-tolerance software than there is in the rest of the software! Three reasons for this are: (1) The higher density of complex conditional branches in this type of software. (2) The lack of understanding of all possible failure scenarios leading to vague or incomplete requirements. (3) This software is the last to be tested... when the funding and schedule are exhausted. My boss once said that "All system failures are caused by design faults." This is because, regardless of the requirements, critical systems should be designed to never fail. It is extremely rare for a critical system to fail in a way that was anticipated by the designers (e.g., redundancy exhaustion). NASA's C. Michael Holloway observed: "To a first approximation, we can say that accidents are almost always the result of incorrect estimates of the likelihood of one or more things." This keynote will explore the factors that lead to designers underestimating the possibility/probabilities of certain failures. Examples of rare, but actually occurring, failures will be given. These will include Byzantine faults, component transmogrification, "evaporating" software, and exhaustively tested software that still failed. The well known Murphy's Law states that: "If anything can go wrong, it will go wrong." For critical systems, the following should added: "And, if anything can't go wrong, it will go wrong anyway."

Monitoring Big, Distributed, Streaming Data

Assaf Schuster (Technion, Israel)

More and more tasks require efficient processing of continuous queries over scalable, distributed data streams. Examples include optimizing systems using their operational log history, mining sentiments using sets of crawlers, and data fusion over heterogeneous sensor networks. However, distributed mining and/or monitoring of global behaviors can be prohibitively difficult. The naïve solution which sends all data to a central location mandates extremely high communication volume, thus incurring unbearable overheads in terms of resources and energy. Furthermore, such solutions require expensive powerful central platform,

while data transmission may violate privacy rules. An attempt to enhance the naïve solution by periodically polling aggregates is bound to fail, exposing a vicious tradeoff between communication and latency. Given a continuous global query, the solution proposed in the talk is to generate filters, called safe zones, to be applied locally at each data stream. Essentially, the safe zones represent geometric constraints which, until violated by at least one of the sources, guarantee that a global property holds. In other words, the safe zones allow for constructive quiescence: There is no need for any of the data sources to transmit anything as long as all constraints are held with the local data confined to the local safe zone. The typically-rare violations are handled immediately, thus the latency for discovering global conditions is negligible. The safe zones approach makes the overall system implementation, as well as its operation, much simpler and cheaper. The saving, in terms of communication volume, can reach many orders of magnitude. The talk will describe a general approach for compiling efficient safe zones for many tasks and system configurations.

Formal Methods: An Industrial Perspective

Jeannette Wing (Carnegie Mellon University and Microsoft Research, USA)

Formal methods research has made tremendous progress since the 1980s when a proof using a theorem prover was worthy of a Ph.D. thesis and a bug in a VLSI textbook was found using a model checker. Now, with advances in theorem proving, model checking, satisfiability modulo theories (SMT) solvers, and program analysis, the engines of formal methods are more sophisticated and are applicable and scalable: to a wide range of domains, from biology to mathematics; to a wide range of systems, from asynchronous systems to spreadsheets; and for a wide range of properties, from security to program termination. In this talk, I will present a few Microsoft Research stories of advances in formal methods and their application to Microsoft products and services. Formal methods use, however, is not routine—yet—in industrial practice. So, I will close with outstanding challenges and new directions for research in formal methods.

Invited Tutorials

A Lattice-Theoretic Approach to Monitoring Distributed Computations

Vijay K. Garg (UT Austin, USA)
Neeraj Mittal (UT Dallas, USA)

Reasoning about distributed programs is hard because the non-deterministic interleaving of concurrent activities in the system dramatically increases the number of possible executions of the program. This non-determinism also makes it difficult to test or verify the correctness of a distributed program before deployment. Continuous monitoring of a running system is a complementary approach for increasing the dependability of a distributed program after deployment.

An execution of a distributed system, also referred to as a distributed computation, can be modeled as a partially ordered set (poset) of events ordered by the happened-before relation. The set of all consistent global states of the computation correspond to the lattice of all down-sets of the poset. The problem of runtime monitoring can be viewed as evaluating a predicate on this lattice. In this tutorial, we will give a survey of algorithms and their limitations for evaluating global predicates in distributed systems. The algorithms exploit lattice-theoretic properties of predicates for efficiency. For example, if the given predicate B is meet-closed and join-closed, then we can compute a subcomputation (called slice) which exactly captures all the consistent global states that satisfy B. We will describe centralized and distributed algorithms to compute such a slice. We also show how slices can be used to detect temporal logic predicates in a distributed computation.

Runtime Monitoring and Enforcement of Security Policies

David Basin (ETH Zurich, Switzerland)
Felix Klaedtke (NEC Europe Ltd., Switzerland)

Many kinds of digitally stored data should only be used in restricted ways. The intended usage may be stipulated by government regulations, corporate privacy policies, preferences of the data owner, etc. Such policies cover not only who may access which data, but also how the data may or must not be used after access. An example of such a usage restriction is that "collected data must be deleted after 30 days and not accessed or forwarded to third parties."

In this tutorial, we present different methods and results for monitoring and enforcing such policies along with their underlying foundations. We show how temporal logical can be used not only to formalize such regulations, but to

synthesize efficient monitors from specifications. These monitors can then be used either online or offline to check whether the behavior of system agents, i.e., users and processes, is policy compliant. A particular focus here will be on the use of metric first-order temporal logic as a policy language, its algorithmic realization in the MonPoly tool, and our experience using this tool.

We will also consider the question of when and how can such policies be enforced by execution monitoring. We will review Schneider's seminal work on policy enforcement as well as its limitations. We will show how to overcome the limitations of Schneider's setting by distinguishing between system actions that are controllable by an enforcement mechanism and those actions that are only observable, that is, the enforcement mechanism sees them but cannot prevent their execution. For this refined setting, we give necessary and sufficient conditions on when a security policy is enforceable. Furthermore for different specification languages, we investigate the problem of deciding whether a given policy is enforceable and synthesizing an enforcement mechanism from an enforceable policy.

Table of Contents

Runtime Verification of Real-Time and Embedded Systems

Testing and Bug Finding

Inference and Learning