Laure Petrucci · Cristina Seceleanu
Ana Cavalcanti (Eds.)

Critical Systems: Formal Methods and Automated Verification

Joint 22nd International Workshop
on Formal Methods for Industrial Critical Systems
and 17th International Workshop
on Automated Verification of Critical Systems, FMICS-AVoCS 2017
Turin, Italy, September 18–20, 2017
Proceedings

Springer

Editors
Laure Petrucci
Paris 13 University
Villetaneuse
France

Ana Cavalcanti
University of York
York
UK

Cristina Seceleanu
Mälardalen University
Västerås
Sweden

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-67112-3 ISBN 978-3-319-67113-0 (eBook)
DOI 10.1007/978-3-319-67113-0

Library of Congress Control Number: 2017952389

LNCS Sublibrary: SL2 – Programming and Software Engineering

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 International Workshop on Formal Methods for Industrial Critical Systems and Automated Verification of Critical Systems (FMICS-AVoCS), held in Turin, Italy, September 18–20, 2017. FMICS-AVoCS 2017 combines the 22nd International Workshop on Formal Methods for Industrial Critical Systems and the 17th International Workshop on Automated Verification of Critical Systems.

The aim of the FMICS workshop series is to provide a forum for researchers who are interested in the development and application of formal methods in industry. In particular, FMICS brings together scientists and practitioners who are active in the area of formal methods and interested in exchanging their experiences in the industrial usage of these methods. The FMICS workshop series also strives to promote research and development that targets the improvement of formal methods and tools for industrial applications.

The aim of the AVoCS workshop series is to contribute to the interaction and exchange of ideas among members of the international research community on tools and techniques for the verification of critical systems. The subject is to be interpreted broadly and inclusively. It covers all aspects of automated verification, including model checking, theorem proving, SAT/SMT constraint solving, abstract interpretation, and refinement pertaining to various types of critical systems (safety-critical, business-critical, performance-critical, etc.) that need to meet stringent dependability requirements.

This year we received 30 submissions, out of which 8 were submitted to the new special track on "Formal methods for mobile and autonomous robots", focusing on the design, verification, and implementation of mobile and autonomous robots based on formal methods.

Each of these submissions went through a rigorous review process in which each paper was reviewed by at least three researchers from a strong Program Committee of international reputation. We selected 14 papers, 4 of them for the special track, for presentation during the workshop and inclusion in the workshop's proceedings, which resulted in an acceptance rate of 47%.

The regular track papers span various topics on system modeling and verification, such as deductive verification of code, automata learning techniques, event-based timing constraints verification, and model checking software components, as well as topics related to testing and scheduling, such as automatic conformance testing of industrial systems, model-based testing of asynchronous systems, and formal-methods-backed schedulability analysis.

The papers accepted for the special track cover recent results and open problems related to verifying mobile and autonomous robots.

The workshop also featured keynotes by Prof. Parosh Abdullah (Uppsala University, Sweden) and Prof. Kerstin Eder (University of Bristol, UK), and a tutorial offered by Prof. Tiziana Margaria (University of Limerick and Lero - The Irish Software Research Centre, Ireland) and Prof. Bernhard Steffen (TU Dortmund, Germany). We hereby thank the invited speakers for having accepted our invitation, and the tutors for organizing the tutorial.

We are grateful to the editorial staff of Springer for publishing the workshop's proceedings, EasyChair for assisting us in managing the complete process from submission to proceedings, as well as ERCIM and EASST for their support. Finally, we would like to thank the Program Committee members and the external reviewers, for their accurate and timely reviews, all authors for their submissions, and all attendees of the workshop for their participation.

July 2017

Laure Petrucci
Cristina Seceleanu
Ana Cavalcanti

Organization

Program Committee

María Alpuente	Universitat Politècnica de València, Spain
Jiří Barnat	Masaryk University, Czech Republic
Ana Cavalcanti	University of York, UK
Michael Dierkes	Rockwell Collins, France
Kerstin Eder	University of Bristol, UK
Alessandro Fantechi	Università degli Studi di Firenze, Italy
Michael Fisher	University of Liverpool, UK
Francesco Flammini	Ansaldo STS, Naples, Italy
María Del Mar Gallardo	University of Málaga, Spain
Michael Goldsmith	University of Oxford, UK
Gudmund Grov	Heriot-Watt University, UK
Matthias Güdemann	Diffblue Ltd., Oxford, UK
Marieke Huisman	University of Twente, The Netherlands
Gerwin Klein	NICTA and University of New South Wales, Australia
Lars Kristensen	Bergen University College, Norway
Anna-Lena Lamprecht	University of Limerick, Ireland
Peter Gorm Larsen	Aarhus University, Denmark
Thierry Lecomte	ClearSy, Aix-en-Provence, France
Radu Mateescu	Inria Grenoble - Rhône-Alpes, France
David Mentré	Mitsubishi Electric R&D Centre Europe, Rennes, France
Stephan Merz	Inria Nancy, France
Manuel Núñez	Universidad Complutense de Madrid, Spain
Charles Pecheur	Université catholique de Louvain, Belgium
Marielle Petit-Doche	Systerel, Aix-en-Provence, France
Laure Petrucci	Université Paris 13 and CNRS, France
Markus Roggenbach	Swansea University, UK
Matteo Rossi	Politecnico di Milano, Italy
Marco Roveri	FBK-irst, Italy
Thomas Santen	Microsoft Research Advanced Technology Labs Europe, Germany
Cristina Seceleanu	Mälardalen University, Sweden
Bernhard Steffen	University of Dortmund, Germany
Jun Sun	Singapore University of Technology and Design, Singapore
Maurice Ter Beek	ISTI-CNR, Pisa, Italy
Helen Treharne	University of Surrey, UK
Xavier Urbain	Université Claude Bernard Lyon 1, France

Jaco van de Pol	University of Twente, The Netherlands
Peter Ölveczky	University of Oslo, Norway

Additional Reviewers

Armand, Michaël
Basile, Davide
Bendík, Jaroslav
Boudjadar, Jalil
Boyer, Benoît
Bozzano, Marco
Brecknell, Matthew
Carnevali, Laura
Cousineau, Denis
Cruanes, Simon
Dennis, Louise
Dixon, Clare
Griggio, Alberto
Guérin Lassous, Isabelle
Happa, Jassim
Insa, David
Kamali, Maryam
Lang, Frédéric
Linker, Sven
Longuet, Delphine
Macedo, Hugo Daniel
Marsso, Lina
Merino, Pedro
Micheli, Andrea
Murray, Toby
Panizo, Laura
Pardo, Daniel
Poskitt, Christopher M.
Potop-Butucaru, Maria
Salmerón, Alberto
Tixeuil, Sébastien
Wang, Jingyi

Replacing Store Buffers by Load Buffers in Total Store Ordering (Invited Lecture)

Parosh Aziz Abdulla[1], Mohamed Faouzi Atig[1], Ahmed Bouajjani[2], and Tuan Phong Ngo[1]

[1] Uppsala University, Uppsala, Sweden
{parosh,mohamed_faouzi.atig,tuan-phong.ngo}@it.uu.se
[2] IRIF, Université Paris Diderot, Paris, France
abou@irif.fr

To gain more efficiency and save energy, almost all modern multi-processor architectures execute instructions in an out-of-order fashion. This means that processors execute instructions in an order governed by the availability of input data rather than by their original order in the program. The out-of-order execution does not affect the behavior of *sequential* programs. However, in the concurrent setting, many new (and unexpected) behaviors may be observed in program executions. We can no longer assume the classical Sequential Consistency (SC) semantics that has for decades been the standard semantics for concurrent programs. Sequential consistency means that "the result of any execution of the program is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program" [8]. In fact, even well-known concurrent algorithms such as mutual exclusion and producer-consumer protocols that are correct under the SC semantics, may not satisfy their specifications any more when run on modern architectures. This means that it is relevant to carry out program verification in order to to ensure correctness under these new premises.

To carry out formal verification, we need to have a well-defined semantics for the program under consideration. The inadequacy of the SC semantics has led to the invention of new program semantics, so called *Weak, (or relaxed) Memory Models*, by allowing permutations between certain types of memory operations [4–6]. One of the most popular memory models is Total Store Ordering (TSO) that corresponds, among others, to the relaxation adopted by Sun's SPARC multiprocessors [11] and formalizations of the Intel x86-tso memory model [9, 10]. The TSO model inserts an unbounded non-lossy (perfect) FIFO buffer (queue), called a *store buffer*, between each processor and the main memory. When a processor performs a write operation, the memory will not be immediately updated as is the case in the SC semantics. Instead, the write operation will be appended to the tail of the store buffer of the processor. In such a case, we say that the write operation is *pending*. A pending write operation is only visible to the processor that has issued it, but not to the rest of the processors. At any point during the execution of the program, the memory may be *updated*, i.e., the write operation at the head of the store buffer of one of the processors may non-deterministically be fetched and used to update the memory. The update operation overwrites the memory position corresponding to the variable on which the write operation is performed.

After the update operation, the write operation will be visible to all the processors. If a processor performs a read operation, then it searches first its own store buffer for the latest pending write operation on the same variable. If no pending write operation exists on that variable in the buffer, the processor fetches the value from the memory.

In this lecture, we describe an alternative semantics called the *dual TSO* semantics [3]. The new semantics is equivalent to the classical TSO semantics but more amenable for efficient algorithmic verification. The main idea is to replace the store buffers of the processors by *load buffers*. The load buffer of a processor contains pending read operations instead of write operations. Intuitively, the read operation at the end of a buffer can be consumed and used to perform a local read operation by the processor. The flow of information will now be in the reverse direction, i.e., write operations by processors will immediately update the memory, while the values of the variables are propagated non-deterministically from the memory to the load buffers of the processors. When a processor performs a read operation, it fetches its value from the tail of its buffer.

One interesting aspect of the dual semantics is that it presents a new (yet equivalent) view of the classical memory model of TSO. Furthermore, the model allows to incorporate *lossiness* into the semantics. More precisely, if we extend the semantics by allowing the load buffers of the processors to lose messages non-deterministically, then the set of reachable processor states will remain the same. The equivalent lossy semantics allows the application the framework of well-structured systems [1, 2, 7] in a straightforward manner leading to a simple proof of decidability of safety properties for finite-state programs operating on Dual-TSO.

References

1. Abdulla, P., Cerans, K., Jonsson, B., Tsay, Y.: General decidability theorems for infinite-state systems. In: LICS 1996, pp. 313–321. IEEE Computer Society (1996)
2. Abdulla. P.A.: Well (and better) quasi-ordered transition systems. Bull. Symb. Log. **16**(4), 457–515, (2010)
3. Abdulla, P.A., Atig, M.F., Bouajjani, A., Ngo, T.P.: The benefits of duality in verifying concurrent programs under TSO. In: Desharnais, J., Jagadeesan, R. (eds.) 27th International Conference on Concurrency Theory, CONCUR 2016, 23–26 August 2016, Québec City, Canada, vol. 59. LIPIcs, pp. 5:1–5:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
4. Adve, S., Gharachorloo, K.: Shared memory consistency models: a tutorial. Computer **29** (12) 1996
5. Adve, S., Hill, M.D.: Weak ordering - a new definition. In: ISCA (1990)
6. Dubois, M., Scheurich, C., Briggs, F.A.: Memory access buffering in multiprocessors. In: ISCA (1986)
7. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. **256**(1–2), 63–92 (2001)

8. Lamport, L.: How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Trans. Comp. **C-28**(9) (1979)
9. Owens, S., Sarkar, S., Sewell, P: A better x86 memory model: x86-tso. In: TPHOL (2009)
10. Sewell, P., Sarkar, S., Owens, S., Nardelli, F.Z., Myreen, M.O.: x86-tso: a rigorous and usable programmer's model for x86 multiprocessors. CACM **53** (2010)
11. Weaver, D., Germond, T. (eds.): The SPARC Architecture Manual Version 9. PTR Prentice Hall (1994)

Contents

Automated Verification Techniques

Deductive Functional Verification of Safety-Critical Embedded C-Code: An Experience Report 3
Dilian Gurov, Christian Lidström, Mattias Nyberg, and Jonas Westman

Verifying Event-Based Timing Constraints by Translation into Presburger Formulae 19
Björn Lisper

Query Checking for Linear Temporal Logic 34
Samuel Huang and Rance Cleaveland

Testing and Scheduling

Automatic Conformance Testing of Safety Instrumented Systems for Offshore Oil Platforms 51
Hallan William Veiga, Max Hering de Queiroz, Jean-Marie Farines, and Marcelo Lopes de Lima

Model-Based Testing for Asynchronous Systems 66
Alexander Graf-Brill and Holger Hermanns

Information Leakage as a Scheduling Resource 83
Fabrizio Biondi, Mounir Chadli, Thomas Given-Wilson, and Axel Legay

A Unified Formalism for Monoprocessor Schedulability Analysis Under Uncertainty 100
Étienne André

Special Track: Formal Methods for Mobile and Autonomous Robots

CRutoN: Automatic Verification of a Robotic Assistant's Behaviours 119
Paul Gainer, Clare Dixon, Kerstin Dautenhahn, Michael Fisher, Ullrich Hustadt, Joe Saunders, and Matt Webster

Sampling-Based Reactive Motion Planning with Temporal Logic Constraints and Imperfect State Information 134
Felipe J. Montana, Jun Liu, and Tony J. Dodd

Sampling-Based Path Planning for Multi-robot Systems with Co-Safe Linear Temporal Logic Specifications 150
Felipe J. Montana, Jun Liu, and Tony J. Dodd

Certified Gathering of Oblivious Mobile Robots: Survey of Recent Results and Open Problems 165
Thibaut Balabonski, Pierre Courtieu, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain

Modeling and Analysis Techniques

Learning-Based Testing the Sliding Window Behavior of TCP Implementations 185
Paul Fiterău-Broştean and Falk Howar

Optimizing Feature Interaction Detection 201
Alessandro Fantechi, Stefania Gnesi, and Laura Semini

Formalising the Dezyne Modelling Language in mCRL2 217
Rutger van Beusekom, Jan Friso Groote, Paul Hoogendijk, Robert Howe, Wieger Wesselink, Rob Wieringa, and Tim A.C. Willemse

Erratum to: Certified Gathering of Oblivious Mobile Robots: Survey of Recent Results and Open Problems. E1
Thibaut Balabonski, Pierre Courtieu, Lionel Rieg, Sébastien Tixeuil, and Xavier Urbain

Author Index 235