

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison, UK

Josef Kittler, UK

Friedemann Mattern, Switzerland

Moni Naor, Israel

Bernhard Steffen, Germany

Doug Tygar, USA

Takeo Kanade, USA

Jon M. Kleinberg, USA

John C. Mitchell, USA

C. Pandu Rangan, India

Demetri Terzopoulos, USA

Gerhard Weikum, Germany

## Advanced Research in Computing and Software Science

Subline of Lecture Notes in Computer Science

## Subline Series Editors

Giorgio Ausiello, *University of Rome 'La Sapienza', Italy*

Vladimiro Sassone, *University of Southampton, UK*

## Subline Advisory Board

Susanne Albers, *TU Munich, Germany*

Benjamin C. Pierce, *University of Pennsylvania, USA*

Bernhard Steffen, *University of Dortmund, Germany*

Deng Xiaotie, *Peking University, Beijing, China*

Jeannette M. Wing, *Microsoft Research, Redmond, WA, USA*

More information about this series at <http://www.springer.com/series/7407>

Constantin Enea · Ruzica Piskac (Eds.)

# Verification, Model Checking, and Abstract Interpretation

20th International Conference, VMCAI 2019  
Cascais, Portugal, January 13–15, 2019  
Proceedings

*Editors*

Constantin Enea  
IRIF  
University Paris Diderot and CNRS  
Paris, France

Ruzica Piskac  
Yale University  
New Haven, CT, USA

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-030-11244-8              ISBN 978-3-030-11245-5 (eBook)  
<https://doi.org/10.1007/978-3-030-11245-5>

Library of Congress Control Number: 2018966547

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer Nature Switzerland AG 2019

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. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

# Preface

This volume contains the papers presented at VMCAI 2019: the International Conference on Verification, Model Checking, and Abstract Interpretation held during January 13–15, 2019, in Cascais, Portugal, co-located with POPL 2019 (the annual ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages). Previous meetings were held in Port Jefferson (1997), Pisa (1998), Venice (2002), New York (2003), Venice (2004), Paris (2005), Charleston (2006), Nice (2007), San Francisco (2008), Savannah (2009), Madrid (2010), Austin (2011), Philadelphia (2012), Rome (2013), San Diego (2014), Mumbai (2015), St. Petersburg, Florida (2016), Paris (2017), and Los Angeles (2018).

VMCAI provides a forum for researchers from the communities of verification, model checking, and abstract interpretation to present their research and aims to facilitate interaction, cross-fertilization, and advancement of hybrid methods that combine these and related areas. VMCAI topics include: program verification, model checking, abstract interpretation, program synthesis, static analysis, type systems, deductive methods, decision procedures, theorem proving, program certification, debugging techniques, program transformation, optimization, hybrid and cyber-physical systems.

This year the conference received 62 submissions, of which 27 were selected for publication in the proceedings. Each submission was reviewed by at least three Program Committee members, and the main selection criteria were quality, relevance, and originality. In addition to the presentations of the 27 selected papers, the conference also featured three invited keynote talks by Nuno P. Lopes (Microsoft Research), Kedar Namjoshi (Nokia Bell Labs), Sylvie Putot (Ecole Polytechnique). We warmly thank them for their participation and contributions.

We would like to thank the members of the Program Committee and the external reviewers for their excellent work. We also thank the members of the Steering Committee, and in particular Lenore Zuck and Andreas Podelski, for their helpful advice, assistance, and support. We thank the POPL 2019 Organizing Committee for providing all the logistics for organizing VMCAI. We are also indebted to EasyChair for providing an excellent conference management system.

November 2018

Constantin Enea  
Ruzica Piskac

# Organization

## Program Co-chairs

Constantin Enea  
Ruzica Piskac

IRIF, University of Paris Diderot and CNRS, France  
Yale University, USA

## Program Committee

Miltiadis Allamanis

Timos Antonopoulos

Domagoj Babic

Josh Berdine

Ahmed Bouajjani

Patrick Cousot

Cezara Dragoi

Constantin Enea

Javier Esparza

Jerome Feret

Khalil Ghorbal

Roberto Giacobazzi

Alberto Griggio

Jan Kretinsky

Ori Lahav

Anthony Widjaja Lin

Ruben Martins

Kedar Namjoshi

K. Narayan Kumar

Dejan Nickovic

Jens Palsberg

Ruzica Piskac

Sylvie Putot

Daniel Schwartz-Narbonne

Martina Seidl

Sharon Shoham

Caterina Urban

Lenore Zuck

Damien Zufferey

Microsoft Research

Yale University, USA

Google, Inc.

Facebook

IRIF, University of Paris Diderot and CNRS, France

New York University, USA

Inria Paris, ENS, France

IRIF, University of Paris Diderot and CNRS, France

Technical University of Munich, Germany

Inria Paris, ENS, France

Inria Rennes, France

University of Verona, Italy

Fondazione Bruno Kessler, Italy

Technical University of Munich, Germany

Tel Aviv University, Israel

University of Oxford, UK

Carnegie Mellon University, USA

Nokia Bell Labs

Chennai Mathematical Institute, India

Austrian Institute of Technology AIT, Austria

University of California, Los Angeles, USA

Yale University, USA

LIX, Ecole Polytechnique

Amazon Web Services

Johannes Kepler University Linz, Austria

Tel Aviv University, Israel

ETH Zurich, Switzerland

University of Illinois at Chicago, USA

MPI-SWS

## **Additional Reviewers**

Alpernas, Kalev  
Ashok, Pranav  
Athaiya, Snigdha  
Balatsouras, George  
Bardin, Sebastien  
Bernardi, Giovanni  
Bouaziz, Mehdi  
Bozga, Marius  
Choi, Wontae  
Cox, Arlen  
Eilers, Marco  
Enea, Constantin  
Goubault, Eric  
Gurfinkel, Arie  
Habermehl, Peter  
Hanam, Quinn  
Herbreteau, Frédéric  
Hollingum, Nicholas  
Irfan, Ahmed  
Itzhaky, Shachar  
Jaax, Stefan  
Kaminski, Benjamin Lucien  
Krämer, Julia  
Laroussinie, François

Le, Xuan Bach  
Lefaucheux, Engel  
Meggendorfer, Tobias  
Meyer, Philipp  
Meyer, Roland  
Niskanen, Reino  
Padon, Oded  
Praveen, M.  
Rajani, Vineet  
Rotar, Alexej  
Roveri, Marco  
Sankur, Ocan  
Sighireanu, Mihaela  
Simon, Axel  
Sogokon, Andrew  
Srinivasan, Venkatesh  
Srivathsan, B.  
Thibault, Joan  
Titolo, Laura  
Trabish, David  
Vizel, Yakir  
Weininger, Maximilian  
Welzel, Christoph  
Wolf, Karsten

## **Abstract of Invited Keynote Talks**



# Semantics for Compiler IRs: Undefined Behavior is not Evil!

Nuno P. Lopes

Microsoft Research  
nlopes@microsoft.com

## Summary

Building a compiler IR is tricky. First, it should be efficient to compile the desired source language(s) (C, C++, Rust, etc) to this IR. Second, the IR should support all the desired optimizations and analyses, and these should run efficiently. Finally, it should be possible to lower this IR into the desired target(s) assembly efficiently. Striking a good tradeoff in this design space is not easy.

Undefined behavior (UB) has been used in production compilers' IRs for many years, including all of GCC, ICC, LLVM, MSVC. Perhaps surprisingly, even formally verified compilers which target safety-critical systems, such as CompCert [3], have UB in their IR.

In this talk, we will explore what UB is, what it achieves, why it may be a good idea, and why it is not as evil as most people think it is. This is based on work on formalizing LLVM IR's UB semantics [2], a memory model for LLVM supporting UB [1], and work on formal verification of LLVM optimizations that exploit UB [4].

**Short Bio:** Nuno Lopes is a researcher at MSR Cambridge. He holds a PhD from the University of Lisbon, and has previously interned at MSR Redmond, Apple, Max Planck Institute (MPI-SWS), and the Institute for Systems and Robotics (ISR) Lisbon. Nuno's interests include software verification, compilers, and mixing the two.

## References

1. Lee, J., Hur, C.-K., Jung, R., Liu, Z., Regehr, J., Lopes, N.P.: Reconciling high-level optimizations and low-level code in LLVM. In: Proceedings of the ACM on Programming Languages, vol. 2(OOPSLA), November 2018
2. Lee, J., Kim, Y., Song, Y., Hur, C.-K., Das, S., Majnemer, D., Regehr, J., Lopes, N.P.: Taming undefined behavior in LLVM. In: PLDI (2017)
3. Leroy, X.: Formal verification of a realistic compiler. Commun. ACM **52**(7), 107–115 (2009)
4. Lopes, N.P., Menendez, D., Nagarakatte, S., Regehr, J.: Provably correct peephole optimizations with Alive. In: PLDI (2015)

# Designing Self-certifying Software Systems

Kedar S. Namjoshi

Bell Labs, Nokia

`kedar.namjoshi@nokia-bell-labs.com`

**Abstract.** Large software systems are hard to understand. The size and complexity of the implementation, possibly written in a mix of programming languages, the number of potential configurations, concurrency, distribution, and several other factors contribute to the difficulty of precisely analyzing system behavior. How can one have confidence in the correct working of such a complex system? In this talk, I explore an unusual approach to this challenge. Suppose that a software system is designed so that it produces a mathematical justification (a *certificate*) for the correctness of its result. The behavior of such a *self-certifying* system can then be formally verified at run time, merely by checking the validity of each certificate as it is generated, without having to examine or reason directly about the system implementation. Self-certification thus shrinks the size of the trusted computing base, often by orders of magnitude, as only the certificate checker must be trusted. The central research question is the design of a certificate format that is comprehensive, easy to generate, and straightforward to check. I will sketch how this may be done for a variety of software system types: model checkers and static analyzers, network operating systems, and optimizing compilers. I will also discuss several intriguing open questions and describe some of the unexpected benefits of certification.

**Short Bio:** Kedar Namjoshi is a member of technical staff at Nokia Bell Labs in Murray Hill, NJ. He received his Ph.D. from the University of Texas at Austin with E. Allen Emerson, and the B.Tech. degree from the Indian Institute of Technology, Madras, both in the Computing Sciences. His research interests include program semantics, specification logics and verification, model checking, static program analysis, distributed computing, and programming methodology.

# Under and Over Approximated Reachability Analysis for the Verification of Control Systems

Sylvie Putot

LIX, CNRS and Ecole Polytechnique, Palaiseau, France  
`putot@lix.polytechnique.fr`

**Abstract.** This talk will present a class of methods to compute under and over approximating flowpipes [1, 2] for differential systems, possibly with delays, systems that are pervasive in the modeling of networked control systems. Computing over-approximations of the reachable states has become a classical tool for the safety verification of control systems. Under-approximations are notoriously more difficult to compute, and their use for verification much less studied. We will discuss the guarantees and properties that can be obtained from the joint use of these under and over-approximations for control systems with inputs and disturbances.

**Short Bio:** Sylvie Putot is Professor in the Department of Computer Science of Ecole Polytechnique. Her research focuses on set-based methods and abstractions for the verification of numerical programs and more generally cyber-physical systems. She is also one of the main authors of the Fluctuat static analyzer, dedicated to the analysis of floating-point programs.

## References

1. Goubault, E., Putot, S.: Forward inner-approximated reachability of non-linear continuous systems. In: Frehse, G., Mitra, S. (eds.) Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC 2017, Pittsburgh, PA, USA, 18–20 April 2017. ACM (2017)
2. Goubault, E., Putot, S., Sahlmann, L.: Inner and outer approximating flowpipes for delay differential equations. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10982, pp. 523–541. Springer, Cham (2018)

# Contents

On the Semantics of Snapshot Isolation . . . . .	1
<i>Azalea Raad, Ori Lahav, and Viktor Vafeiadis</i>	
Program Synthesis with Equivalence Reduction . . . . .	24
<i>Calvin Smith and Aws Albarghouthi</i>	
Minimal Synthesis of String to String Functions from Examples . . . . .	48
<i>Jad Hamza and Viktor Kunčák</i>	
Automatic Program Repair Using Formal Verification and Expression Templates . . . . .	70
<i>Thanh-Toan Nguyen, Quang-Trung Ta, and Wei-Ngan Chin</i>	
Lazy but Effective Functional Synthesis . . . . .	92
<i>Grigory Fedyukovich, Arie Gurfinkel, and Aarti Gupta</i>	
Static Analysis of Binary Code with Memory Indirections Using Polyhedra . . . . .	114
<i>Clément Ballabriga, Julien Forget, Laure Gonnord, Giuseppe Lipari, and Jordy Ruiz</i>	
Disjunctive Relational Abstract Interpretation for Interprocedural Program Analysis . . . . .	136
<i>Rémy Boutonnet and Nicolas Halbwachs</i>	
Exploiting Pointer Analysis in Memory Models for Deductive Verification . . . . .	160
<i>Quentin Bouillaguet, François Bobot, Mihaela Sighireanu, and Boris Yakobowski</i>	
Small Faults Grow Up - Verification of Error Masking Robustness in Arithmetically Encoded Programs . . . . .	183
<i>Anja F. Karl, Robert Schilling, Roderick Bloem, and Stefan Mangard</i>	
Relatively Complete Pushdown Analysis of Escape Continuations. . . . .	205
<i>Kimball Germane and Matthew Might</i>	
Demand Control-Flow Analysis . . . . .	226
<i>Kimball Germane, Jay McCarthy, Michael D. Adams, and Matthew Might</i>	

Effect-Driven Flow Analysis. . . . .	247
<i>Jens Nicolay, Quentin Stiévenart, Wolfgang De Meuter, and Coen De Roover</i>	
Type-Directed Bounding of Collections in Reactive Programs. . . . .	275
<i>Tianhan Lu, Pavol Černý, Bor-Yuh Evan Chang, and Ashutosh Trivedi</i>	
Solving and Interpolating Constant Arrays Based on Weak Equivalences. . . . .	297
<i>Jochen Hoenicke and Tanja Schindler</i>	
A Decidable Logic for Tree Data-Structures with Measurements. . . . .	318
<i>Xiaokang Qiu and Yanjun Wang</i>	
A Practical Algorithm for Structure Embedding . . . . .	342
<i>Charlie Murphy and Zachary Kincaid</i>	
EUFORIA: Complete Software Model Checking with Uninterpreted Functions. . . . .	363
<i>Denis Bueno and Karem A. Sakallah</i>	
Fast BGP Simulation of Large Datacenters. . . . .	386
<i>Nuno P. Lopes and Andrey Rybalchenko</i>	
Verification of an Industrial Asynchronous Leader Election Algorithm Using Abstractions and Parametric Model Checking . . . . .	409
<i>Étienne André, Laurent Fribourg, Jean-Marc Mota, and Romain Soulat</i>	
Application of Abstract Interpretation to the Automotive Electronic Control System. . . . .	425
<i>Tomoya Yamaguchi, Martin Brain, Chirs Ryder, Yosikazu Imai, and Yoshiumi Kawamura</i>	
Syntactic Partial Order Compression for Probabilistic Reachability . . . . .	446
<i>Gereon Fox, Daniel Stan, and Holger Hermanns</i>	
Termination of Nondeterministic Probabilistic Programs. . . . .	468
<i>Hongfei Fu and Krishnendu Chatterjee</i>	
Parametric Timed Broadcast Protocols. . . . .	491
<i>Étienne André, Benoit Delahaye, Paulin Fournier, and Didier Lime</i>	
Flat Model Checking for Counting LTL Using Quantifier-Free Presburger Arithmetic . . . . .	513
<i>Normann Decker and Anton Pirogov</i>	
A Parallel Relation-Based Algorithm for Symbolic Bisimulation Minimization . . . . .	535
<i>Richard Huybers and Alfons Laarman</i>	

Combining Refinement of Parametric Models with Goal-Oriented Reduction of Dynamics . . . . .	555
<i>Stefan Haar, Juraj Kolčák, and Loïc Paulevé</i>	
Mechanically Proving Determinacy of Hierarchical Block Diagram Translations. . . . .	577
<i>Viorel Preoteasa, Iulia Dragomir, and Stavros Tripakis</i>	
<b>Author Index</b> . . . . .	601