Lecture Notes in Computer Science **11408**

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

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

Fred Mesnard · Peter J. Stuckey (Eds.)

Logic-Based Program Synthesis and Transformation

28th International Symposium, LOPSTR 2018
Frankfurt/Main, Germany, September 4–6, 2018
Revised Selected Papers

Springer

Editors
Fred Mesnard
University of Reunion Island
Sainte-Clotilde, France

Peter J. Stuckey
Monash University
Melbourne, VIC, Australia

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-13837-0 ISBN 978-3-030-13838-7 (eBook)
https://doi.org/10.1007/978-3-030-13838-7

Library of Congress Control Number: 2019932012

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

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 a selection of the papers presented at LOPSTR 2018, the 28th International Symposium on Logic-Based Program Synthesis and Transformation held during September 4–6, 2018 at the the Goethe University Frankfurt am Main, Germany. It was co-located with PPDP 2018, the 20th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, and WFLP 2018, the 26th International Workshop on Functional and Logic Programming. The co-location of these related conferences has occurred several times and has been stimulating and cross-fertilizing.

Previous LOPSTR symposia were held in Namur (2017), Edinburgh (2016), Siena (2015), Canterbury (2014), Madrid (2013 and 2002), Leuven (2012 and 1997), Odense (2011), Hagenberg (2010), Coimbra (2009), Valencia (2008), Lyngby (2007), Venice (2006 and 1999), London (2005 and 2000), Verona (2004), Uppsala (2003), Paphos (2001), Manchester (1998, 1992, and 1991), Stockholm (1996), Arnhem (1995), Pisa (1994), and Louvain-la-Neuve (1993). More information about the symposium can be found at: http://ppdp-lopstr-18.cs.uni-frankfurt.de/lopstr18.html.

The aim of the LOPSTR series is to stimulate and promote international research and collaboration on logic-based program development. LOPSTR is open to contributions on all aspects of logic-based program development, all stages of the software life cycle, and issues of both programming-in-the-small and programming-in-the-large. LOPSTR traditionally solicits contributions, in any language paradigm, in the areas of synthesis, specification, transformation, analysis and verification, specialization, testing and certification, composition, program/model manipulation, optimization, transformational techniques in software engineering, inversion, applications, and tools. LOPSTR has a reputation for being a lively, friendly forum that allows for the presentation and discussion of both finished work and work in progress. Formal proceedings are produced only after the symposium so that authors can incorporate the feedback from the conference presentation and discussion.

In response to the calls for papers, 29 contributions were submitted from ten countries. The Program Committee accepted seven full papers for immediate inclusion in the formal proceedings, and four more papers presented at the symposium were accepted after a revision and another round of reviewing. Each submission was reviewed by at least three Program Committee members or external referees. The paper "Proving Program Properties as First-Order Satisfiability" by Salvador Lucas won the best paper award, sponsored by Springer. In addition to the 11 contributed papers, this volume includes the abstracts of the invited talks by three outstanding speakers: Philippa Gardner (Imperial College London, UK) and Jorge A. Navas (SRI International, USA), whose talks were shared with PPDP, and Laure Gonnord (University of Lyon 1, France), whose talk was shared with WFLP. We also had two invited tutorials: Fabio Fioravanti (University of Chieti-Pescara, Italy) presented "The VeriMAP System

for Program Transformation and Verification" and Manuel Hermenegildo (IMDEA Software Institute and Technical University of Madrid, Spain) summarized "25 Years of Ciao."

We want to thank the Program Committee members, who worked diligently to produce high-quality reviews for the submitted papers, as well as all the external reviewers involved in the paper selection. We are very grateful to the local organizer, David Sabel, and his team for the great job they did in managing the symposium. Many thanks also to Peter Thiemann, the Program Committee chair of PPDP, and Josep Silva, the Program Committee chair of WFLP, with whom we interacted for coordinating the events. We would also like to thank Andrei Voronkov for his excellent EasyChair system that automates many of the tasks involved in chairing a conference.

Special thanks go to the invited speakers and to all the authors who submitted and presented their papers at LOPSTR 2018. We also thank our sponsors, the Goethe University Frankfurt am Main, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), and Springer for their cooperation and support in the organization of the symposium.

January 2019

Fred Mesnard
Peter J. Stuckey

Organization

Program Committee

Elvira Albert	Universidad Complutense de Madrid, Spain
Sandrine Blazy	University of Rennes 1 - IRISA, France
Mats Carlsson	SICS, Sweden
Agostino Dovier	Università degli Studi di Udine, Italy
Wlodek Drabent	IPI PAN Warszawa, Poland
Gregory Duck	National University of Singapore, Singapore
Maurizio Gabbrielli	University of Bologna, Italy
Juergen Giesl	RWTH Aachen University, Germany
Michael Hanus	CAU Kiel, Germany
Salvador Lucas	Universitat Politècnica de València, Spain
Fred Mesnard	Université de La Réunion, France
Etienne Payet	Université de La Réunion, France
Alberto Pettorossi	Università di Roma Tor Vergata, Italy
Vitor Santos Costa	University of Porto, Portugal
Tom Schrijvers	Katholieke Universiteit Leuven, Belgium
Julien Signoles	CEA LIST, France
Harald Sondergaard	The University of Melbourne, Australia
Fausto Spoto	University of Verona, Italy
Peter Stuckey	The University of Melbourne, Australia
Markus Triska	Vienna University of Technology, Austria
Wim Vanhoof	University of Namur, Belgium
German Vidal	Universitat Politècnica de València, Spain

Additional Reviewers

Gomez-Zamalloa, Miguel
Gordillo, Pablo
Maurica, Fonenantsoa
Schubert, Aleksy
Villanueva, Alicia
Yamada, Akihisa
Yoshimizu, Akira

Abstracts of Invited Talks

Formal Methods for JavaScript

Philippa Gardner

Imperial College London, UK
pg@doc.ic.ac.uk

Abstract. We present a novel, unified approach to the development of compositional symbolic execution tools, which bridges the gap between traditional symbolic execution and compositional program reasoning based on separation logic. We apply our approach to JavaScript, providing support for full verification, whole-program symbolic testing, and automatic compositional testing based on bi-abduction.

Constrained Horn Clauses for Verification

Jorge Navas

SRI International, USA
Jorge.Navas@sri.com

Abstract. Developing scalable software verification tools is a very difficult task. First, due to the undecidability of the verification problem, these tools, must be highly tuned and engineered to provide reasonable efficiency and precision trade-offs. Second, different programming languages come with very diverse assortments of syntactic and semantic features. Third, the diverse encoding of the verification problem makes the integration with other powerful solvers and verifiers difficult. This talk presents SeaHorn – an open source automated Constrained Horn clause-based reasoning framework. SeaHorn combines advanced automated solving techniques based on Satisfiability Modulo Theory (SMT) and Abstract Interpretation. SeaHorn is built on top of LLVM using its front-end(s) to deal with the idiosyncrasies of the syntax and it highly benefits from LLVM optimizations to reduce the verification effort. SeaHorn uses Constrained Horn clauses (CHC) which are a uniform way to formally represent a broad variety of transition systems while allowing many encoding styles of verification conditions. Moreover, the recent popularity of CHC as an intermediate language for verification engines makes it possible to interface SeaHorn with a variety of new and emerging tools. All of these features make SeaHorn a versatile and highly customizable tool which allows researchers to easily build or experiment with new verification techniques.

Experiences in Designing Scalable Static Analyses

Laure Gonnord

University of Lyon 1, France
Laure.Gonnord@univ-lyon1.fr

Abstract. Proving the absence of bugs in a given software (problem which has been known to be intrinsically hard since Turing and Cook) is not the only challenge in software development. Indeed, the ever growing complexity of software increases the need for more trustable optimisations. Solving these two problems (reliability, optimisation) implies the development of safe (without false negative answers) and efficient (wrt memory and time) analyses, yet precise enough (with few false positive answers). In this talk I will present some experiences in the design of scalable static analyses inside compilers, and try to make a synthesis about the general framework we, together with my coauthors, used to develop them. I will also show some experimental evidence of the impact of this work on real-world compilers, as well as future perspective for this area of research.

Abstracts of Invited Tutorials

The VeriMAP System for Program Transformation and Verification

Fabio Fioravanti

University of Chieti-Pescara, Italy
fioravanti@unich.it

Abstract. Constrained Horn Clauses (CHC) are becoming very popular for representing programs and verification problems, and several tools have been developed for checking their satisfiability. In this tutorial we will survey recent work on satisfiability-preserving transformation techniques for CHC and we will show how the VeriMAP system can be used effectively to (i) generate CHC verification conditions from the programming language semantics, (ii) prove safety properties of imperative programs manipulating integers and arrays, (iii) prove relational program properties, such as program equivalence and non-interference, (iv) check the satisfiability of CHC with inductively-defined data structures (e.g. lists and trees), (v) prove safety and controllability properties of time-aware business processes.

25 Years of Ciao

Manuel Hermenegildo

IMDEA Software Institute and Technical University of Madrid, Spain
herme@fi.upm.es

Abstract. Ciao is a logic-based, multi-paradigm programming language which has pioneered over the years many interesting language- and programming environment-related concepts. An example is the notion of programming languages as modular language-building tools rather than closed designs. Another is the idea of dynamic languages that can optionally and gradually offer formal guarantees, which is also a solution for the classic dichotomy between dynamic and static typing: Ciao has many dynamic features (e.g., dynamically typed, dynamic program modification) but includes an assertion language for (optionally) declaring program properties and powerful tools for static inference and static/dynamic checking of such assertions, testing, documentation, etc. We will provide a hands-on overview of these features, concentrating on the novel aspects, the motivations behind their design and implementation, their evolution over time, and, specially, their use. In particular, we will show how the system can be used not only as a programming tool and as a language design tool, but also as a general-purpose program analysis and verification tool, based on the technique of translating program semantics (ranging from source to bytecode, LLVM, or assembly) into Horn-clause representation, and idea which Ciao also introduced early on. Finally, we will present some recent work in areas such as scalability, incrementality, or static vs. dynamic costs, as well as some future plans and ideas.

Contents