Bor-Yuh Evan Chang (Ed.)

Programming Languages and Systems

15th Asian Symposium, APLAS 2017
Suzhou, China, November 27–29, 2017
Proceedings

Springer

Editor
Bor-Yuh Evan Chang
University of Colorado
Boulder, CO
USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-71236-9 ISBN 978-3-319-71237-6 (eBook)
https://doi.org/10.1007/978-3-319-71237-6

Library of Congress Control Number: 2017959598

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 proceedings of the 15th Asian Symposium on Programming Languages and Systems (APLAS 2017), held in Suzhou, China during November 27–29, 2017. APLAS aims to stimulate programming language research by providing a forum for the presentation of latest results and the exchange of ideas in programming languages and systems. APLAS is based in Asia but is an international forum that serves the worldwide programming languages community.

APLAS 2017 solicited submissions in two categories: regular research papers and tool demonstrations. The conference solicits contributions in, but is not limited to, the following topics: semantics, logics, and foundational theory; design of languages, type systems, and foundational calculi; domain-specific languages; compilers, interpreters, and abstract machines; program derivation, synthesis, and transformation; program analysis, verification, and model-checking; logic, constraint, probabilistic and quantum programming; software security; concurrency and parallelism; and tools for programming and implementation.

New to APLAS in 2017, the conference employed a double-blind reviewing process with an author-response period. Within the review period, APLAS 2017 used an internal two-round review process where each submission received three first-round reviews to drive the possible selection of additional expert reviews as needed before the author response period. All submissions received at least three reviews with nearly half of the submissions receiving four or five reviews. The author response period was followed by a two-week Program Committee discussion period with over 425 comments generated and culminating in a synchronous, virtual Program Committee meeting on August 11, 2017, to finalize the selection of papers.

This year APLAS received 56 submissions. After thoroughly evaluating the relevance and quality of each paper, the Program Committee decided to accept 24 contributions. We were also honored to include four invited talks by distinguished researchers:

- Gilles Barthe (IMDEA, Spain) on “Relational Verification of Higher-Order Probabilistic Programs”
- Ron Garcia (University of British Columbia, Canada) on “Gradual Enforcement of Program Invariants”
- Sumit Gulwani (Microsoft Research, USA) on “Programming by Examples: PL Meets ML”
- Naijun Zhan (Chinese Academy of Sciences, China) on “Synthesizing SystemC Code from Delay Hybrid CSP”

This program would not have been possible without the substantial efforts of many people, whom I sincerely thank. The Program Committee, sub-reviewers, and external expert reviewers worked tirelessly to select the strongest possible program while simultaneously offering constructive and supportive comments in their reviews.

Xinyu Feng (University of Science and Technology of China) serving as general chair of APLAS 2017 ensured that all aspects of the conference planning were addressed. I also graciously thank the APLAS Steering Committee for their leadership, as well as APLAS 2016 PC chair Atsushi Igarashi (Kyoto University, Japan) for timely advice.

Lastly, I would like to acknowledge the organizers of the associated events that makes APLAS a truly exciting event: the Poster Session and Student Research Competition (Yu Zhang, University of Science and Technology of China) and the APLAS Workshop on New Ideas and Emerging Results (Wei-Ngan Chin, National University of Singapore and Zhenjiang Hu, National Institute of Informatics, Japan).

September 2017 Bor-Yuh Evan Chang

Organization

General Chair

Xinyu Feng	University of Science and Technology of China

Program Chair

Bor-Yuh Evan Chang	University of Colorado Boulder

Program Committee

Andreas Abel	Gothenburg University, Sweden
Aws Albarghouthi	University of Wisconsin-Madison, USA
Sam Blackshear	Facebook, USA
Yu-Fang Chen	Academia Sinica, Taiwan
Yuting Chen	Shanghai Jiao Tong University, China
Stephen Chong	Harvard University, USA
Vijay D'Silva	Google, USA
Benjamin Delaware	Purdue University, USA
Rayna Dimitrova	The University of Texas at Austin, USA
Cezara Dragoi	Inria, ENS, CNRS, France
William Harris	Georgia Institute of Technology, USA
Guoliang Jin	North Carolina State University, USA
Akash Lal	Microsoft Research, India
Vu Le	Microsoft, USA
Akimasa Morihata	The University of Tokyo, Japan
Sergio Mover	University of Colorado Boulder, USA
Santosh Nagarakatte	Rutgers University, USA
Hakjoo Oh	Korea University, South Korea
Bruno C.D.S. Oliveira	The University of Hong Kong, SAR China
Xiaokang Qiu	Purdue University, USA
Arjun Radhakrishna	University of Pennsylvania, USA
Aseem Rastogi	Microsoft Research, India
Sukyoung Ryu	KAIST, South Korea
Ilya Sergey	University College London, UK
Makoto Tatsuta	National Institute of Informatics, Japan
Tachio Terauchi	Waseda University, Japan
Bow-Yaw Wang	Academia Sinica, Taiwan
Yingfei Xiong	Peking University, China
Kwangkeun Yi	Seoul National University, South Korea

Danfeng Zhang	Pennsylvania State University, USA
Xin Zhang	Georgia Institute of Technology, USA
Kenny Zhu	Shanghai Jiao Tong University, China

Poster Chair

Yu Zhang	University of Science and Technology of China

Workshop on New Ideas and Emerging Results Organizers

Wei-Ngan Chin	National University of Singapore
Zhenjiang Hu	National Institute of Informatics, Japan

Asian Association for Foundation of Software Executive Committee

Co-chairs

Wei-Ngan Chin	National University of Singapore
Zhenjiang Hu	National Institute of Informatics, Japan

Members

Xinyu Feng	University of Science and Technology of China
Yuxi Fu	Shanghai Jiao Tong University, China
Jacques Garrigue	Nagoya University, Japan
Atsushi Igarashi	Kyoto University, Japan
Ranjit Jhala	University of California, San Diego, USA
Yukiyoshi Kameyama	University of Tsukuba, Japan
Naoki Kobayashi	The University of Tokyo, Japan
Shin-Cheng Mu	Academia Sinica, Taiwan
Sungwoo Park	Pohang University of Science and Technology, South Korea
Chung-chieh Shan	Indiana University, USA
Zhong Shao	Yale University, USA
Harald Sondergaard	The University of Melbourne, Australia
Kazunori Ueda	Waseda University, Japan
Hongseok Yang	KAIST, South Korea
Kwangkeun Yi	Seoul National University, South Korea

Additional Reviewers

Abstracts of Invited Talks

Relational Verification of Higher-Order Probabilistic Programs

Gilles Barthe

IMDEA Software Institute, Madrid, Spain

Hyperproperties go beyond the traditional formulation of program verification by considering sets of sets of traces—in contrast to program properties which consider sets of traces. Common instances of hyperproperties include robustness, information flow security, and for probabilistic programs differential privacy. These latter properties are instances of the more restricted class of 2-properties, which contemplate related executions of the same program, or executions of two different programs. These properties can be formally established using lightweight type systems, which are tailored to enfore specific classes of properties, relational program logics, which are tailored to reason about relations between two programs, or product programs which construct from each pair of programs a single product program that emulates their behavior. One challenge, independently of the approach chosen, is to develop methods that support syntax-directed reasoning that is traditionally favoured in standard verification and yet provides sufficient flexibility to accommodate programs that are structurally different or have diverging control flow on different but related inputs.

The talk shall present and compare the different approaches, including Relational Higher-Order Logic [1]. Moreover, it will present several applications, including relational cost and security.

Reference

1. Aguirre, A., Barthe, G., Gaboardi, M., Garg, D., Strub, P.-Y.: A relational logic for higher-order programs. PACMPL 1(ICFP), 21:1–21:29 (2017)

Programming by Examples: PL Meets ML

Sumit Gulwani[1] and Prateek Jain[2]

[1] Microsoft Corporation, Redmond, USA
sumitg@microsoft.com
[2] Microsoft Research, Bangalore, India
prajain@microsoft.com

Abstract. Programming by Examples (PBE) involves synthesizing intended programs in an underlying domain-specific language from example-based specifications. PBE systems are already revolutionizing the application domain of data wrangling and are set to significantly impact several other domains including code refactoring.

There are three key components in a PBE system. (i) A search algorithm that can efficiently search for programs that are consistent with the examples provided by the user. We leverage a divide-and-conquer-based deductive search paradigm that inductively reduces the problem of synthesizing a program expression of a certain kind that satisfies a given specification into sub-problems that refer to sub-expressions or sub-specifications. (ii) Program ranking techniques to pick an intended program from among the many that satisfy the examples provided by the user. We leverage features of the program structure as well of the outputs generated by the program on test inputs. (iii) User interaction models to facilitate usability and debuggability. We leverage active-learning techniques based on clustering inputs and synthesizing multiple programs.

Each of these PBE components leverage both symbolic reasoning and heuristics. We make the case for synthesizing these heuristics from training data using appropriate machine learning methods. This can not only lead to better heuristics, but can also enable easier development, maintenance, and even personalization of a PBE system.

Gradual Enforcement of Program Invariants

Ronald Garcia

University of British Columbia, Vancouver, British Columbia, Canada
rxg@cs.ubc.ca

Abstract. Static and dynamic techniques have long been used to check and enforce properties of program executions. They are often seen as diametrically opposed, as exemplified by the long-running kerfuffle over the merits and deficits of static versus dynamic type checking.

Recently, PL researchers and designers have sought to bridge the divide between these approaches to program checking and analysis. In particular, *gradual typing* sets out to seamlessly combine static and dynamic checking of how closely programs adhere to standard typing disciplines from the literature. In this context, static and dynamic checking and enforcement are treated as complementary rather than conflicting.

In this talk I will discuss the theory and practice of gradual typing. Both have undergone significant development in the last few years. These advances in language design change not only how dynamic and static checking can work together, but also change how we think about each individually.

Synthesizing SystemC Code from Delay Hybrid CSP

Gaogao Yan[1,2], Li Jiao[1], Shuling Wang[1], and Naijun Zhan[1,2]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
{yangg,ljiao,wangsl,znj}@ios.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Delay is omnipresent in modern control systems, which can prompt oscillations and may cause deterioration of control performance, invalidate both stability and safety properties. This implies that safety or stability certificates obtained on idealized, delay-free models of systems prone to delayed coupling may be erratic, and further the incorrectness of the executable code generated from these models. However, automated methods for system verification and code generation that ought to address models of system dynamics reflecting delays have not been paid enough attention yet in the computer science community. In our previous work, on one hand, we investigated the verification of delay dynamical and hybrid systems; on the other hand, we also addressed how to synthesize SystemC code from a verified hybrid system modelled by Hybrid CSP (HCSP) without delay. In this paper, we give a first attempt to synthesize SystemC code from a verified delay hybrid system modelled by Delay HCSP (*d*HCSP), which is an extension of HCSP by replacing ordinary differential equations (ODEs) with delay differential equations (DDEs). We implement a tool to support the automatic translation from *d*HCSP to SystemC.

Contents