Lecture Notes in Computer Science

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

Editorial Board

David Hutchison
   Lancaster University, UK
Takeo Kanade
   Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
   University of Surrey, Guildford, UK
Jon M. Kleinberg
   Cornell University, Ithaca, NY, USA
Friedemann Mattern
   ETH Zurich, Switzerland
John C. Mitchell
   Stanford University, CA, USA
Moni Naor
   Weizmann Institute of Science, Rehovot, Israel
C. Pandu Rangan
   Indian Institute of Technology, Madras, India
Bernhard Steffen
   TU Dortmund University, Germany
Demetri Terzopoulos
   University of California, Los Angeles, CA, USA
Doug Tygar
   University of California, Berkeley, CA, USA
Gerhard Weikum
   Max Planck Institute for Informatics, Saarbruecken, Germany
Preface

This volume contains the proceedings of the 12th Asian Symposium on Programming Languages and Systems (APLAS 2014), held in Singapore, during November 17–19, 2014. APLAS aims at stimulating programming language research by providing a forum for the presentation of the latest results and the exchange of ideas in topics concerned with programming languages and systems. APLAS is based in Asia, but is an international forum that serves the worldwide programming language community. Past APLAS symposia were successfully held in Melbourne (2013), Kyoto (2012), Kenting (2011), Shanghai (2010), Seoul (2009), Bangalore (2008), Singapore (2007), Sydney (2006), Tsukuba (2005), Taipei (2004), and Beijing (2003) after three informal workshops.

The topics covered in the conference include, but are not limited to, 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; software security; concurrency and parallelism; and tools and environments for programming, verification, and implementation.

This year, 57 papers were submitted to APLAS. Each submission was reviewed by three or more Program Committee members. After thoroughly evaluating the relevance and quality of each paper, the committee chose to accept 24 papers for presentation at the conference.

This year’s program also continued the APLAS tradition of invited talks by distinguished researchers:

- Zhenjiang Hu (NII) on “What Is the Essence of Bidirectional Programming?”
- Julien Verlaguet (Facebook) on “Incremental Adoption of Static-Typing”
- Dexter Kozen (Cornell University) on “NetKAT: A Formal System for the Verification of Networks”

This program would not have been possible without the unrelenting efforts of several people, whom we would like to thank. First, the Program Committee and additional reviewers for the hard work put in toward ensuring the high quality of the proceedings. Our thanks also go to the Asian Association for Foundation of Software (AAFS), founded by Asian researchers in cooperation with many researchers from Europe and the USA, for sponsoring and supporting APLAS. We would like to warmly thank the Steering Committee in general and Wei-Ngan Chin and Cristian Gherghina for their support in organizing the conference and the poster session. Finally, we are grateful to Andrei Voronkov whose EasyChair system eased the processes of submission, paper selection, and proceedings compilation.

September 2014
Jacques Garrigue
Organization

Program Committee

Xiaojuan Cai  
Shanghai Jiao Tong University, China

James Chapman  
IoC, Tallinn University of Technology, Estonia

Jacques Garrigue  
Nagoya University, Japan

Cristian Gherghina  
Singapore University of Technology and Design

Eric Goubault  
CEA, France

Fei He  
Tsinghua University, China

Gerwin Klein  
NICTA and UNSW, Australia

Raghavan Komondoor  
Indian Institute of Science, Bangalore, India

Paddy Krishnan  
Oracle, Australia

Daan Leijen  
Microsoft Research, USA

Yasuhiko Minamide  
University of Tsukuba, Japan

Shin-Cheng Mu  
Academia Sinica, Taiwan

Sungwoo Park  
Pohang University of Science and Technology, South Korea

Julian Rathke  
University of Southampton, UK

Sukyoung Ryu  
KAIST, South Korea

Alexandra Silva  
Radboud University Nijmegen, The Netherlands

Martin Sulzmann  
Hochschule Karlsruhe, Germany

Munehiro Takimoto  
Tokyo University of Science, Japan

Jan Vitek  
Northeastern University, USA

Hongwei Xi  
Boston University, USA

Additional Reviewers

Andronick, June  
Geuvers, Herman

Athaiya, Snigdha  
Hage, Jurriaan

Basold, Henning  
He, Chaodong

Boyland, John  
Im, Hyeonseung

Chuang, Tyng-Ruey  
Jeannin, Jean-Baptiste

Clairambault, Pierre  
Jongmans, Sung-Shik T.Q.

Costea, Andreea  
K.R., Ravangendra

Cristescu, Ioana  
Krebers, Robbert

Danish, Matthew  
Laird, James

Fahrenberg, Uli  
Lewis, Corey

Fernandez, Matthew  
Liang, Hongjin

Ferrara, Pietro  
Liangze, Yin
<table>
<thead>
<tr>
<th>Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liu, Jiaxiang</td>
<td>Roux, Pierre</td>
</tr>
<tr>
<td>Lopes, Antónia</td>
<td>Sammartino, Matteo</td>
</tr>
<tr>
<td>Mackie, Ian</td>
<td>Santosa, Andrew</td>
</tr>
<tr>
<td>McBride, Conor</td>
<td>Sato, Haruhiko</td>
</tr>
<tr>
<td>Mimram, Samuel</td>
<td>Schöpp, Ulrich</td>
</tr>
<tr>
<td>Miné, Antoine</td>
<td>Sewell, Thomas</td>
</tr>
<tr>
<td>Moy, Yannick</td>
<td>Sharma, Asankhaya</td>
</tr>
<tr>
<td>Murray, Toby</td>
<td>Sneyers, Jon</td>
</tr>
<tr>
<td>Oh, Hakjoo</td>
<td>Ta, Quang Trung</td>
</tr>
<tr>
<td>Petri, Gustavo</td>
<td>Taghdiri, Mana</td>
</tr>
<tr>
<td>Pouzet, Marc</td>
<td>Tzevelekos, Nikos</td>
</tr>
<tr>
<td>Pérez, Jorge A.</td>
<td>Vazou, Niki</td>
</tr>
<tr>
<td>Rama, Girish Maskeri</td>
<td>Voigtländer, Janis</td>
</tr>
<tr>
<td>Ren, Zhiqiang</td>
<td>Wang, Bow-Yaw</td>
</tr>
<tr>
<td>Robert, Thomas</td>
<td></td>
</tr>
</tbody>
</table>
Invited Presentations
What Is the Essence of Bidirectional Programming?

Zhenjiang Hu
National Institute of Informatics, Japan
hu@nii.ac.jp

Bidirectional transformations [8, 4, 13] provide a novel mechanism for synchronizing and maintaining the consistency of information between input and output. The idea of bidirectional transformations is originated from the view updating mechanism in the database community [1, 5, 9], and has been attracting a lot of attention from a wide range of communities, including programming languages, software engineering and databases, which has motivated the proposal of a vast number of bidirectional approaches aiming to solve the problems of different bidirectional applications.

A bidirectional transformation basically consists of a pair of transformations: the forward transformation $get\ s$ is used to produce a target view $v$ from a source $s$, while the putback transformation $put\ s\ v$ is used to reflect modifications on the view $v$ to the source $s$. These two transformations should be well-behaved in the sense that they satisfy the following round-tripping laws.

$$\text{GetPut} \quad put\ s\ (get\ s) = s$$
$$\text{PutGet} \quad get\ (put\ s\ v) = v$$

The GetPut property requires that no change in the view shall be reflected as no change in the source, while the PutGet property requires all changes in the view to be completely reflected to the source so that the changed view can be computed again by applying the forward transformation to the changed source.

Bidirectional programming is to develop well-behaved bidirectional transformations in order to solve various synchronization problems. A straightforward approach to bidirectional programming is to write two unidirectional transformations. Although this solution provides full control over both get and putback transformations and can be realized using standard programming languages, the programmer needs to show that the two transformations satisfy the well-behavedness laws, and a modification to one of the transformations requires an adaptation of the other transformation as well as a new well-behavedness proof.

It should be preferable to write just a single program that can denote both transformations, in order to ease and enable maintainable bidirectional programming. Then what should this single program be? Most existing bidirectional programming languages are to aid programmers in writing a forward transformation $get$ and deriving a backward transformation $put$ for free [8, 3, 2, 11, 12, 16, 20, 15, 19, 10]. However, the maintainability offered by such languages comes at the cost of expressiveness and (more importantly) predictability because the ambiguity
of synchronization handled by the putback transformation is solved by default strategies over which programmers have little control.

One interesting but less known fact is that while get usually loses information when mapping from a source to a view, put must preserve information when putting back from the view to the source, according to the PutGet property. Furthermore, it has been shown in [7, 6] that, for a putback transformation put, if there exists a forward transformation get then such get is uniquely determined by put. In other words, the essence of bidirectional programming is nothing but to write putback transformation.

In this talk, I will report our recent progress on putback-based bidirectional programming, explaining how to design user-friendly languages for supporting putback-based bidirectional programming [17, 18], showing how to systematically check whether the definition of a put is in a valid form that guarantees that the corresponding unique get exists [14], demonstrating how to apply putback-based bidirectional programming to solve practical problems [6, 21], and highlighting important issues and challenges for future work.

References

Over the last year, Facebook migrated nearly its entire PHP codebase to Hack: a gradually typed language that interoperates seamlessly with PHP. At Facebook’s scale, it would have been difficult to completely transition to Hack right away. To make this transition successful, the language and its type-system had to be designed with interoperability in mind. In this talk, we will review the design decisions that were made during the conception of the language. We will discuss the tradeoffs that had to be considered to find a balance between ease of use and safety.

Hack is an interesting case study of retrofitted static-typing into a dynamic language. Some parts are specific to PHP, but we are hopeful that the lessons learnt will be valuable to anyone interested in dynamic languages and type-systems in general.
NetKAT — A Formal System for the Verification of Networks

Dexter Kozen
Cornell University, Ithaca, NY 14853-7501, USA
kozen@cs.cornell.edu
http://www.cs.cornell.edu/~kozen

Abstract. This paper presents an survey of recent work in the development of NetKAT, a formal system for reasoning about packet switching networks, and its role in the emerging area of software-defined networking.

Keywords: Kleene algebra, Kleene algebra with tests, NetKAT, software defined networking, packet switching, OpenFlow, Frenetic.
Table of Contents

Invited Presentation

NetKAT — A Formal System for the Verification of Networks ........ 1
Dexter Kozen

Regular Papers

Optimized Compilation of Multiset Rewriting with Comprehensions .... 19
Edmund Soon Lee Lam and Iliano Cervesato

Logic Programming and Logarithmic Space ......................... 39
Clément Aubert, Marc Bagnol, Paolo Pistone, and Thomas Seiller

Automatic Memory Management Based on Program Transformation
Using Ownership .................................................. 58
Tatsuya Sonobe, Kohei Suenaga, and Atsushi Igarashi

The Essence of Ruby .............................................. 78
Katsuhiro Ueno, Yutaka Fukasawa, Akimasa Morihata,
and Atsushi Ohori

Types for Flexible Objects ........................................ 99
Zachary Palmer, Pottayil Harisanker Menon,
Alexander Rozenshtein, and Scott Smith

A Translation of Intersection and Union Types for the λμ-Calculus .... 120
Kentaro Kikuchi and Takafumi Sakurai

A Formalized Proof of Strong Normalization for Guarded Recursive
Types ................................................................. 140
Andreas Abel and Andrea Vezzosi

Functional Pearl: Nearest Shelters in Manhattan ...................... 159
Shin-Cheng Mu and Ting-Wei Chen

SUPPL: A Flexible Language for Policies ............................ 176
Robert Dockins and Andrew Tolmach

A Method for Scalable and Precise Bug Finding Using Program
Analysis and Model Checking ....................................... 196
Manuel Valdiviezo, Cristina Cifuentes, and Padmanabhan Krishnan

Model-Checking for Android Malware Detection ...................... 216
Fu Song and Tayssir Touili
Necessary and Sufficient Preconditions via Eager Abstraction .......................... 236
Mohamed Nassim Seghir and Peter Schrammel

Resource Protection Using Atomics: Patterns and Verification ...................... 255
Afshin Amighi, Stefan Blom, and Marieke Huisman

Resource Analysis of Complex Programs with Cost Equations ................. 275
Antonio Flores-Montoya and Reiner Hähnle

Simple and Efficient Algorithms for Octagons ................................. 296
Aziem Chawdhary, Ed Robbins, and Andy King

Compositional Entailment Checking for a Fragment of Separation Logic .................. 314
Constantin Enea, Ondřej Lengál, Mihaela Sighireanu, and Tomáš Vojnar

Automatic Constrained Rewriting Induction towards Verifying Procedural Programs .......................... 334
Cynthia Kop and Naoki Nishida

A ZDD-Based Efficient Higher-Order Model Checking Algorithm ............ 354
Taku Terao and Naoki Kobayashi

Inferring Grammatical Summaries of String Values .............................. 372
Se-Won Kim, Wooyoung Chin, Jimin Park, Jeongmin Kim, and Sukyoung Ryu

Syntax-Directed Divide-and-Conquer Data-Flow Analysis ......................... 392
Shigeyuki Sato and Akimasa Morihata

Address Chain: Profiling Java Objects without Overhead in Java Heaps ...................... 408
Xiaohua Shi, Junru Xie, and Hengyang Yu

Call-by-Value in a Basic Logic for Interaction .................................. 428
Ulrich Schöpp

A Precise and Abstract Memory Model for C Using Symbolic Values .......... 449
Frédéric Besson, Sandrine Blazy, and Pierre Wilke

Hereditary History-Preserving Bisimilarity: Logics and Automata ........... 469
Paolo Baldan and Silvia Crafa

Author Index ................................................................................. 489