Special issue: Advanced Programming Techniques for Construction of Robust, General and Evolutionary Programs

Guest Editorial

Advanced programming techniques for construction of robust, generic and evolutionary programs

Zhenjiang HU[1], Shin-Cheng MU[2] and Stephanie WEIRICH[3]
[1]*National Institute of Informatics*
[2]*Academia Sinica*
[3]*University of Pennsylvania*

In computer science, the way that we investigate computation is with programming. Programming is more than just writing programs to instruct the computer to do something, it seeks formal and reliable means of ensuring that programs possess crucial properties. Therefore, developers must be concerned with many issues such as robustness, reliability, adaptability, and maintainability, while meeting program specifications. Advanced programming techniques, such as dependently typed programming, generic programming and bidirectional programming, have been developed to deal with these issues.

Programming languages with static type systems have had great success in ensuring that data and control structures are used in appropriate ways. *Dependently typed programming* is a powerful programming mechanism based on the dependent type system. By allowing types to refer to data, programmers can define more fine-grained program behaviors which is often dynamic in practice, being able to communicate the design of software to computers and negotiate their place in the spectrum of precision from basic memory safety to total correctness.

Generic programming techniques have been a specific focus of research in the functional and object-oriented programming communities. They make programs more adaptable by embodying non-traditional kinds of polymorphism; ordinary programs are obtained from them by suitably instantiating their parameters. In contrast with normal programs, the parameters of a generic program are often quite rich in structure, which could be other programs, types or type constructors, class hierarchies, or even programming paradigms. Generic programming has been gradually spreading to more and more mainstream languages, and today is widely used in industry.

Bidirectional programming is a recent techniques aiming to construct well-behaved bidirectional programs (bidirectional transformations) that can be executed both forwardly and backwardly. Bidirectional transformation, originated from the known *view updating* mechanism in the database community, has been attracting more and more attention from researchers in the communities of programming and programming languages. Bidirectional transformation provides a powerful mechanism for synchronizing and maintaining the consistency of information between input and output, and has seen many interesting applications, including the synchronization of replicated data in different formats, presentation-oriented structured document development, interactive user interface design, and coupled software transformation.

This special issue aims to publish high quality papers on these advanced programming techniques that can lead to practical and effective processes for constructing robust, general and evolutionary programs. In response to the Call for Papers for this special issue, eight papers were submitted, and out of them six papers have been accepted. Each submission was reviewed by at least three referees, and the decisions on selecting papers are based on originality, technical contribution, practical contribution and relevance. We believe that the accepted papers exhibit important aspects of advanced programming techniques. In addition to the research papers, this special issue includes the technical reports of two NII Shonan Meetings: "Automated Techniques for Higher-Order Program Verification" organized by David Van Horn (Northeastern University), Naoki Kobayashi (Tohoku University), and C.-H. Luke Ong (University of Oxford); and "Dependently Typed Programming" organized by Shin-Cheng Mu (Academia Sinica, Taiwan), Conor McBride (Univer-

Received January 8, 2013.
DOI: 10.2201/NiiPi.2013.10.1

sity of Strathclyde, UK), and Stephanie Weirich (University of Pennsylvania, USA).

We, the guest editors, would like to thank the authors contributed to the special issue. We would also like to express the deep appreciation to the referees who worked with us through the tight schedule. Finally, we thank Ayumi Shimizu for her support throughout the editorial process.

Zhenjiang HU

Zhenjiang HU is Professor of NationalInstitute of Informatics (NII) and The Graduate University for Advanced Studies in Japan. He received his BS and MS from Shanghai Jiao Tong University in 1988 and 1991 respectively, and PhD degree from University of Tokyo in 1996. He was a lecture (1997-1999) and an associate professor (2000-2007) in University of Tokyo, before joining NII as a full professor in 2008. His main interest is in programming languages and software engineering in general, and functional programming, parallel programming and bidirectional model-driven software development in particular. He is now serving on the steering committees of ACM ICFP, APLAS and FLOPS, and is the academic committee chair of the NII Shonan Meetings.

Shin-Cheng MU

Shin-Cheng MU is an associate research fellow in Academia Sinica, Taiwan. He graduated from the Algebra of Programming group in Oxford University, and once worked as a postdoc researcher in the Programmable Structured Document project in Information Processing Lab, University of Tokyo. His interests include program derivation, program calculation, and type theory.

Stephanie WEIRICH

Stephanie WEIRICH is an Associate Professor at the University of Pennsylvania. She joined Penn after receiving her PhD from Cornell University in 2002. Her research concerns the type systems of functional programming languages, generic programming, dependent type systems and type inference.