Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 364 (2006) 271–272
Theoretical Computer Science
www.elsevier.com/locate/tcs

# Preface

This special issue comprises selected papers answering an open call issued after the *Third Workshop on Applied Semantics* (*APPSEM05*), held in Frauenchiemsee, Germany, September 12–15. This was the final workshop of the thematic network on Applied Semantics (APPSEM-II), funded by the IST programme of the European Union, which was held from January 2003 to June 2006. In total the workshop featured 20 short presentations, 14 full presentations and three invited talks, given by Chris Hankin, Imperial College London; John O'Leary, Intel Portland; and Joe Stoy, Bluespec. The workshop was attended by 53 participants, and included a discussion session on industry applications as well as a steering committee meeting. The call for papers for this special issue was primarily directed to the workshop speakers but also open to other submissions within the range of themes covered by APPSEM-II.

The call resulted in 12 submissions, which were refereed by three experts each in their respective fields. After careful programme committee discussions the present five papers were selected for publication in this special issue. Three of these papers had been presented at the workshop in their preliminary form.

The programme committee for the refereeing process comprised:

- Gavin Bierman, Microsoft Research.
- Olivier Danvy, University of Aarhus.
- Peter Dybjer, Chalmers University of Technology.
- Martin Hofmann, Ludwig-Maximilians-Universität München (Chair).
- Neil Jones, University of Copenhagen.
- Hans-Wolfgang Loidl, Ludwig-Maximilians-Universität München.
- Peter O'Hearn, Queen Mary College, University of London.
- Uday Reddy, University of Birmingham.
- Didier Remy, INRIA Rocquencourt.
- Ian Stark, University of Edinburgh.
- Thomas Streicher, Technische Universität Darmstadt.
- Peter Thiemann, Universität Freiburg.

The primary objective of the APPSEM-II network was to promote research into application-oriented semantics of programming languages. It built on the success of APPSEM-I in bringing researchers from the area of theoretical computer science in contact with practitioners. This cooperation led to advances in using linear types for safe memory management or the application of dependent types in programming languages. In APPSEM-II the direction into application-oriented programming language semantics was continued, covering subtle programming language features, such as pointer aliasing, and advanced constructs in particular those from object-oriented languages. The research programme of APPSEM-II was structured into the following themes:

- Program structuring: object-oriented programming, modules.
- Proof assistants, functional programming, and dependent types.
- Program analysis, generation, and configuration.
- Specification and verification methods.
- Types and type inference in programming.
- Games, sequentiality, and abstract machines.
- Semantic methods for distributed computing.
- Resource models and web data.
- Continuous phenomena in Computer Science.

0304-3975/$ - see front matter 
doi:10.1016/j.tcs.2006.08.011

The papers in this special issue cover a range of foundations for programming languages from classical topics as underlying calculi and program analyses, over the structured modelling and development of complex systems to new trends such as the use of a Proof-Carrying-Code approach to improve security in distributed systems.

The paper by Besson, Jensen, and Pichardie is an attempt at unifying the high level of security warranted by Foundational Proof-Carrying Code (which proves everything formally with respect to a formalisation of the operational semantics) with more pragmatic approaches such as the original PCC by Necula and also *abstraction-carrying code* proposed by Hermenegildo and others. The idea is to formalise a given program analysis in the Coq proof assistant and then to extract an efficient OCAML program from that proof, which can be used at the consumer side to verify certificates without having to run a theorem prover at his end. The important contribution of the paper is not so much this idea in itself but rather the convincing demonstration that it is indeed viable in nontrivial cases.

Laud, Uustalu, and Vene develop and use a connection between abstract interpretation and type systems to obtain a type system for secure information flow, which is stronger than the classical Volpano–Smith system yet admits efficient checking and inference. In addition to its technical merit the paper thus contributes to the interaction between hitherto rather disjoint communities.

The paper by Andrew Kennedy from Microsoft makes an interesting connection between a theoretical concept known as full abstraction and the practical security of Microsofts .NET framework. It is shown that various intuitive security flaws can be formally understood as failures of full abstraction and are thus amenable to a rigorous analysis. This is exemplified by the exhibition of a number of concrete security problems of the .NET platform. Several of the suggested fixes have been taken up in the 2.0 version of .NET.

Cansell and Méry develop a distributed reference counting algorithm by refinement, using the event B model. They start from an abstract system model, gradually refine this model, and use a proof assistant to verify the proof obligations introduced in each refinement step. By contrasting the development and proof with that of a similar distributed reference counting algorithm, proven correct in Coq, this paper highlights the strengths of the entire refinement approach: the proof obligations in each step are much simpler than the invariants in the post-hoc proof, and can often be automatically discharged by the proof assistant. The incremental development of the algorithm, backed by more easily comprehensible invariants in each step, helps to structure the overall algorithm.

The paper by Niehren, Schwinghammer, and Smolka develops a lambda calculus and a linear-type system for a concurrent programming language based on futures: placeholders for possibly unevaluated expressions, with implicit synchronisation, that achieve a data-driven evaluation of concurrent programs. Futures are a powerful language concept in concurrent and distributed systems, and therefore this formalisation has relevance beyond the concurrent language Alice ML, for which it has been originally developed. A linear-type system is used to prove a write-once property on such futures, turning them into I-structures as known from parallel programming. Based on this linear-type system the safety of important concurrency abstractions, such as mutexes and channels, is proven. Crucial meta-properties, in particular subject reduction and confluence, are proven.

Our thanks go to the programme committee and all referees for providing feedback on the submitted papers, and all work-package coordinators in APPSEM-II for building a sustainable research network in this area. We are grateful for the funding of the thematic network on Applied Semantics (APPSEM-II) by the European Union under contract (IST-2001-38957). A summary of the work done in the APPSEM-II network is available from the APPSEM-II web site: http://www.appsem.org/.

Martin Hofmann
Hans-Wolfgang Loidl
*Institut für Informatik,*
*Ludwig-Maximilians-Universität München,*
*Oettingenstrasse 67,*
*D 80538 München, Germany*
*E-mail address*: hwloidl@tcs.ifi.lmu.de (H.-W. Loidl).