Journal of Universal Computer Science, vol. 9, no. 3 (2003), 189-190
submitted: 3/3/03, accepted: 25/3/03, appeared: 28/3/03

J.UCS Special Issue on Compiler Optimization meets Compiler Verification (COCV 2002)

Jens Knoop
(Vienna University of Technology, Austria
knoop@complang.tuwien.ac.at)

Wolf Zimmermann
(Martin-Luther University Halle-Wittenberg, Germany
zimmer@informatik.uni-halle.de)

Semantics preservation between source and target program is the commonly accepted minimum requirement to be ensured by compilers. It is the key term compiler verification and optimization are centered around. The precise meaning, however, is often only implicit. As a rule of thumb, verification tends to interpret semantics preservation in a very tight sense, not only but also to simplify the verification task. Optimization generally prefers a more liberal view in order to enable more powerful transformations otherwise excluded. The surveyor's rod of admissibility is semantics preservation, and hence the language semantics. But the adequate interpretation varies fluently with the application context ("stand-alone" programs, communicating systems, reactive systems, etc.).

This special issue contains revised and extended versions of selected papers from the international workshop on *Compiler Optimization meets Compiler Verification (COCV 2002)*, which has been held in conjunction with the 5th European Joint Conferences on Theory and Practice of Software (ETAPS 2002), Grenoble, France, April 6 - 14, 2002. The aim was to bring together researchers and practitioners working on optimizing and verifying compilation as well as on programming language design and semantics in order to plumb the mutual impact of these fields on each other, the degrees of freedom optimizers and verifiers have, to bridge the gap between the communities, and to stimulate synergies.

The papers finally accepted after a second round of peer-reviewing discuss topics such as certifying compilation, verifying compilation, translation validation, and optimization. Glesner shows how to construct correct code-generators without proving the correctness of the compiler itself. This technique is called program checking or translation validation. The contribution of Zuck et al. uses also this technique to show how the correctness of optimizing loop transformations can be checked. Shashidar et al. also discuss correctness of loop transformations. In contrast to Zuck's approach they distinguish the correctness proof for transformations and their implementation. Nguyen and Irigoin discuss how to verify aliases in FORTRAN. The absence of aliases is an important pre-condition of many optimizations.

The papers in this issue were reviewed, besides the editors, by

- Michael Franz, University of California at Irvine, CA, USA
- Hans Langmaack, Universität Kiel, Germany
- Robert Morgan, DataPower, Cambridge, MA, USA
- Markus Müller-Olm, Universität Dortmund, Germany
- George C. Necula, University of California at Berkely, CA, USA

We are grateful to Dana Kaiser. Her help has been crucial for the success of the editing this special issue. We thank Hermann Maurer for giving us the opportunity to publish this special issue.

Wien, Halle, March 2003

Jens Knoop
Wolf Zimmermann