Under consideration for publication in J. Functional Programming

Special Issue on Generic Programming Editorial

RALF HINZE
Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
ralf.hinze@comlab.ox.ac.uk

Generic programming is about making programs more adaptable by making them more general. Generic programs often embody non-traditional kinds of polymorphism; ordinary programs are obtained from them by suitably instantiating their parameters. In contrast to normal programs, the parameters of a generic program are often quite rich in structure; for example they may be other programs, types or type constructors, classes, concepts, or even programming paradigms.

This special issue documents state-of-the-art research in the broad field of Generic Programming. It is an outgrowth of the series of Workshops on Generic Programming, which started in 1998 and which continues this year with an ICFP affiliated workshop in Baltimore. Participants of the workshops were invited to submit a suitably revised and expanded version of their workshop paper to the special issue. The call for papers was, however, open. Other contributions were equally welcomed and were encouraged.

Eleven papers were submitted in response to the call. Each submission was reviewed by at least four referees, including an expert and an informed outsider. The following five articles were finally selected for inclusion in this special issue:

- Generic programs enjoy generic proofs. The article "Formal Polytypic Programs and Proofs" by Wendy Verbruggen, Edsko de Vries and Arthur Hughes describes a verified implementation of Generic Haskell in the proof assistant Coq, with added support for conducting machine-verified generic proofs.
- Different programming languages differ in their support for generic programming. Haskell scores well in this regard because of its type classes, so does C++ extended with the notion of concepts. The article "Generic Programming with C++ Concepts and Haskell Type Classes" by Jean-Philippe Bernardy, Patrik Jansson, Marcin Zalewski and Sibylle Schupp provides an in-depth comparison between these two features and is potentially useful both to language designers and practising programmers.
- Another hot competitor in this arena is Scala. The article "Scala for Generic Programmers" by Bruno Oliveira and Jeremy Gibbons compares Haskell and Scala support for generic programming, arguing that Scala is in many ways a better choice.
- Generic programs enjoy generic optimisations. The article "Factorising Folds for Faster Functions" by Graham Hutton, Mauro Jaskelioff and Andy Gill introduces a

generic variant of the worker/wrapper transformation, illustrating the technique with numerous examples.

- The article "A Lightweight Approach to Datatype-generic Rewriting" by Thomas van Noort, Alexey Rodriguez Yakushev, Stefan Holdermans, Johan Jeuring, Bastiaan Heeren and José Pedro Magalhaẽs presents an application of generic programming to term rewriting. The authors carefully describe the design and the implementation of a generic rewriting library, written in Haskell extended with GADTs and type families.

I would like to thank the authors and the numerous referees for their efforts in producing and reviewing these articles. Furthermore, special thanks go to Matthias Felleisen and Xavier Leroy for the opportunity to publish the articles as a special issue of the *Journal of Functional Programming*.

Ralf Hinze
Special Issue Editor