JFP **14** (1): 1–2, January 2004. © 2004 Cambridge University Press
DOI: 10.1017/S0956796803004866 Printed in the United Kingdom

Introduction to the Special Issue on Dependent Type Theory Meets Practical Programming

GILLES BARTHE
INRIA Sophia-Antipolis, France
(*e-mail:* Gilles.Barthe@inria.fr)

PETER DYBJEN
Department of Computing Science, Chalmers University, Göteberg, Sweden
(*e-mail:* peterd@cs.chalmers.se)

PETER THIEMANN
Institut für Informatik, Universität Freiberg, Germany
(*e-mail:* thiemann@informatik.uni-freiburg.de)

Modern programming languages rely on advanced type systems that detect errors at compile-time. While the benefits of type systems have long been recognized, there are some areas where the standard systems in programming languages are not expressive enough. Language designers usually trade expressiveness for decidability of the type system. Some interesting programs will always be rejected (despite their semantical soundness) or be assigned uninformative types.

One promising approach for improving the expressiveness of type systems while still maintaining decidability is provided by dependent type systems. Dependent types may explicitly depend on other types or values. Dependent type systems have been thoroughly investigated from a theoretical point of view by logicians and type theorists. Moreover, sophisticated proof assistants for constructing proofs in dependent type theories have been developed, and some recent developments aim to decrease the gap between dependent type theory and practical programming.

To further promote the communication between programming language researchers and dependent type theorists a meeting entitled "Dependent Type Theory Meets Practical Programming" was held in Schloss Dagstuhl in Germany in August 2001. Participants at this seminar were invited to submit full articles to this special issue of the *Journal of Functional Programming*.

Three articles were selected for publication, each of which is briefly summarized below:

- The article by Appel and Felty shows how dependent types can be used for implementing a theorem prover. They work in a logic programming setting and show how dependent type correctness ensures that any proof that the theorem prover builds is valid.
- The article by Kreitz shows how the NUPRL proof assistant for dependent type theory was used for designing reliable high-performance networks.

NUPRL was used for reasoning about optimizations in the ENSEMBLE group communication tool kit.

- The article by McBride and McKinna develops a new high-level style for programming with dependent types. They focus on pattern matching and develop a generalized version of Wadler's notion of 'view'. They conclude by developing a type-checker for the simply typed lambda calculus written in this high-level style.

We would like to thank the authors and the referees for their efforts in producing and reviewing these articles, Simon Peyton Jones and Phil Wadler for offering the opportunity to publish the articles in a special issue of the *Journal of Functional Programming*, and Greg Morrisett for his editorial advice. Finally, we would like to thank Reinhard Wilhelm and the Dagstuhl Office for their help in organizing the seminar in Schloss Dagstuhl.