J Autom Reasoning (2010) 44:1–2
DOI 10.1007/s10817-009-9141-x

# Preface

**Jacques Carette · Makarius Wenzel · Freek Wiedijk**

Published online: 24 June 2009


This volume contains papers dealing with issues in the intersection between programming languages (PL) and mechanized mathematics systems (MMS). These papers were submitted in response to a call for papers after the inaugural PLMMS workshops, held in Hagenburg, Austria in 2007 and Birmingham, UK in 2008 (both associated with the Calculemus conference). PLMMS 2009 will be held in Munich, Germany, in conjunction with the TPHOLs conference.

By mechanized mathematics systems, we understand the category which subsumes present-day computer algebra systems (CAS), interactive proof assistants (PA), and automated theorem provers (ATP), all heading towards *fully integrated mechanized mathematical assistants* that are expected to emerge eventually.

The two subjects of PL and MMS meet in a number of interesting ways, and the original call mentioned a few.

– *Dedicated input languages for MMS*: covers all aspects of languages intended for the user to deploy or extend the system, both algorithmic and declarative ones. Typical examples are tactic definition languages such as Ltac in Coq, mathematical proof languages as in Mizar or Isar, or specialized programming

J. Carette (✉)
Department of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, Ontario, L8S 4K1, Canada
e-mail: carette@mcmaster.ca

M. Wenzel (✉)
Institut für Informatik, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany
e-mail: makarius@sketis.net

F. Wiedijk
Institute for Computing and Information Sciences, Radboud University Nijmegen, Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands
e-mail: freek@cs.ru.nl

languages built into CA systems. Of particular interest are the semantics of those languages, especially when current ones are untyped.
- *Mathematical modeling languages used for programming*: covers the relation of logical descriptions versus algorithmic content. For instance the logic of ACL2 extends a version of Lisp, that of Coq is close to Haskell, and some portions of HOL are similar to ML and Haskell, while Maple tries to do both simultaneously. Such mathematical languages offer rich specification capabilities, which are rarely available in regular programming languages. How can programming benefit from mathematical concepts, without limiting mathematics to the computational world view?
- *Programming languages with mathematical specifications*: covers advanced "mathematical" concepts in programming languages that improve the expressive power of functional specifications, type systems, module systems etc. Programming languages with dependent types are of particular interest here, as is intensionality versus extensionality.
- *Language elements for program verification*: covers specific means built into a language to facilitate correctness proofs using MMS. For example, logical annotations within programs may be turned into verification conditions to be solved in a proof assistant eventually. How need MMS and PL to be improved to make this work conveniently and in a mathematically appealing way?

These issues have a very colourful history. Many programming language innovations first appeared in either CASes or Proof Assistants, before migrating towards more mainstream languages. One can cite (in no particular order) type inference, dependent types, generics, term-rewriting, first-class types, first-class expressions, first-class modules, code extraction, and so on. However, a number of these innovations were never aggressively pursued by system builders, letting them instead be developed (slowly) by programming language researchers. Some, like type inference and generics have flourished. Others, like first-class types and first-class expressions, are not seemingly being researched by anyone.

We felt that there was an untapped vein of valuable research in the PLMMS area, and were rewarded by a substantial number of submissions for this journal issue. Although there have been workshop presentations covering computer algebra related topics as well, we regret that none of the follow-up papers were ready for journal publication.

In this volume, the papers of Sacerdoti Coen and Guidi are extended versions of papers originally presented at PLMMS 2007, and that of Dietrich and Schulz was presented at PLMMS 2008. The other three papers have emerged independently of the two PLMMS meetings.

*Jacques Carette and Freek Wiedijk*
*Workshop Chairs of PLMMS 2007*

*Jacques Carette and Makarius Wenzel*
*Workshop Chairs of PLMMS 2008*