PLMMS Preface

This volume contains the papers presented at PLMMS-2013: 5th International Workshop on Programming Languages for Mechanised Mathematical Systems 2013 held on July 9, 2013 in Bath.

There were 3 submissions. Each submission was reviewed by at 2 program committee members. The committee decided to accept 2 papers. The program also includes 4 invited speakers:

- **Edwin Brady** – Dependently Typed Functional Programming in Idris
  Idris is a general purpose pure functional programming language with dependent types. Its syntax is influenced by Haskell and its features include full dependent types and records, type classes, tactic based theorem proving, totality checking and an optimising compiler with a foreign function interface. One of the goals of the Idris project is to bring type-based program verification techniques to functional programmers while still supporting efficient systems programming via an optimising compiler and interaction with external libraries. In this talk I will introduce dependently typed programming using Idris, and demonstrate its features using several examples including an interpreter for the simply typed lambda calculus, and a verified binary adder.
- **Gilles Dowek** – Checking classical proofs in an constructive proof-checker
  The Dedukti project aims at using a single proof-checker to check proofs developed in many other provers. As some of these provers are classical and other constructive, we need a way to express proofs of one logic into the other. In this talk I will sketch various ways to express classical proofs in a constructive setting, focussing on the possibility to design a single logic mixing classical and constructive connectors and on the possibility to recognize classical proofs that are constructive by chance. This talk will be based on joint work with Olivier Hermant and Frédéric Gilbert.
- **Conor McBride** – Problems as types
  James McKinna coined the phrase "Problems as types" to characterise the presentation of programming and proof (combined) as a problem-solving dialogue, mediating the underlying task of constructing a well typed term. The resulting documents should record both sides of the story—the problems posed, corresponding to the type to be inhabited, and the solution strategy by which those problems are refined to zero or more subproblems, which are elaborated to terms. Our language, Epigram, took a problems-as-types approach to programming, based on the key realisation that the type for a "programming problem" can be more than just the type of a program, also giving the template for its invocation. Whilst dependent types are beginning to catch on, language designers have been at pains to make programming look as much like ordinary functional programming as possible. The problems-as-types approach remains underexplored. In this talk, I shall resume that exploration, looking at problems-as-types approaches to type, program, and proof construction, given recent developments in relevant underlying theories.

– **Sergei Meshveliani** – Dependent Types for an Adequate Programming of Algebra
This research compares the author's experience in programming algebra in Haskell and in Agda (currently the former experience is large, and the latter is small). There are discussed certain hopes and doubts related to the dependently typed and verified programming of symbolic computation. This concerns the 1) author's experience history, 2) algebraic class hierarchy design, 3) proof cost overhead in evaluation and in coding, 4) other subjects. Various examples are considered, some questions are put.

The first paper in this volume is an invited paper by Sergei Meshveliani, also titled Dependent Types for an Adequate Programming of Algebra.

We would like to thank our peer reviewers for carefully reviewing the submissions and giving constructive feedback. We would also like to thank Christoph Lange for his help in efforts in putting together this volume.

June 24, 2013 — Florian Rabe
Newcastle — Iain Whiteside

Program Committee