Monads for functional programming

Philip Wadler, University of Glasgow[⋆]

Department of Computing Science, University of Glasgow, G12 8QQ, Scotland
(wadler@dcs.glasgow.ac.uk)

Abstract. The use of monads to structure functional programs is described. Monads provide a convenient framework for simulating effects found in other languages, such as global state, exception handling, output, or non-determinism. Three case studies are looked at in detail: how monads ease the modification of a simple evaluator; how monads act as the basis of a datatype of arrays subject to in-place update; and how monads can be used to build parsers.

1 Introduction

Shall I be pure or impure?

The functional programming community divides into two camps. *Pure* languages, such as Miranda[0] and Haskell, are lambda calculus pure and simple. *Impure* languages, such as Scheme and Standard ML, augment lambda calculus with a number of possible *effects*, such as assignment, exceptions, or continuations. Pure languages are easier to reason about and may benefit from lazy evaluation, while impure languages offer efficiency benefits and sometimes make possible a more compact mode of expression.

Recent advances in theoretical computing science, notably in the areas of type theory and category theory, have suggested new approaches that may integrate the benefits of the pure and impure schools. These notes describe one, the use of *monads* to integrate impure effects into pure functional languages.

The concept of a monad, which arises from category theory, has been applied by Moggi to structure the denotational semantics of programming languages [13, 14]. The same technique can be applied to structure functional programs [21, 23].

The applications of monads will be illustrated with three case studies. Section 2 introduces monads by showing how they can be used to structure a simple evaluator so that it is easy to modify. Section 3 describes the laws satisfied by

[⋆] Appears in: J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, Proceedings of the Båstad Spring School, May 1995, Springer Verlag Lecture Notes in Computer Science 925. A previous version of this note appeared in: M. Broy, editor, *Program Design Calculi*, Proceedings of the Marktoberdorf Summer School, 30 July–8 August 1992. Some errata fixed August 2001; thanks to Dan Friedman for pointing these out.

[0] Miranda is a trademark of Research Software Limited.