Monadic Reflection in Haskell

Andrzej Filinski

Datalogisk institut, Københavns Universitet
Universitetsparken 1, DK-2100 København Ø
andrzej@diku.dk

Abstract

A monadic-style functional program allows the imperative, behavioral view of a computational effect (e.g., backtracking) to be identified with a declarative, data-based view (e.g., lists) in a uniform way. Monadic reflection refines this identification into an observational isomorphism, with explicit reification and reflection functions mediating between the two views. Such a separation allows the programmer to reason robustly about monadic effects according to the declarative view, while implementing the imperative view much more efficiently in terms of lower-level control and state manipulations.

Monadic reflection has so far mostly been considered systematically in the context of eager, effectful languages, such as ML or Scheme. We examine the prospects of the technique in the setting of Haskell-like languages, from both a theoretical and a practical perspective.

Keywords: Monadic Reflection, Lazy Effectful Languages