Reinventing Haskell Backtracking

Sebastian Fischer
Christian-Albrechts University of Kiel, Germany
sebf@informatik.uni-kiel.de

Abstract: Almost ten years ago, Ralf Hinze has written a functional pearl on how to derive backtracking functionality for the purely functional programming language Haskell. In these notes, we show how to arrive at the efficient, two-continuation based backtracking monad derived by Hinze starting from an intuitive inefficient implementation that we subsequently refine using well known program transformations.

It turns out that the technique can be used to build monads for non-determinism from modular, independent parts which gives rise to various new implementations. Specifically, we show how the presented approach can be applied to obtain new implementations of breadth-first search and iterative deepening depth-first search.

We present a new method to implement monads for non-determinism in Haskell. With our approach, their implementation is split into two independent parts: one is identical for every implementation, the other captures the essence of the employed search strategy. The part that is identical for every implementation includes the monadic bind operation, which does not have to be reimplemented for every non-determinism monad. Programmers only need to define notions of *failure* and *choice* and can wrap these definitions in a parametrised type to obtain a monad for non-determinism.

As the implementation of monadic bind is usually the most involved part in the implementation of non-determinism monads, our approach gives rise to simpler implementations of search strategies that required a more complex implementation before. For example, difference lists are a natural choice to represent non-deterministic computations efficiently but do not support a natural implementation of monadic bind. We show that wrapping the type for difference lists in a continuation monad results in the well-known two-continuation-based backtracking monad derived by Hinze in his influential pearl [Hin00].

Wrapping different base types—which both lack a natural implementation of monadic bind—we obtain novel implementations of breadth-first search and iterative deepening depth-first search. Our main contribution is *the method* used to obtain these strategies from modular parts, not the simplified implementation of established search strategies.

References

[Hin00] Ralf Hinze. Deriving backtracking monad transformers. In *ICFP '00: Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, pages 186–197, New York, NY, USA, 2000. ACM.