Monads

Shengkai Zhou

February 7, 2019

1 The essence of functional programming

```
@inproceedings{w:essence,
  title={The essence of functional programming},
  author={Wadler, Philip},
  booktitle={Proceedings of the 19th ACM SIGPLAN-SIGACT symposium
    on Principles of programming languages},
  pages={1--14},
  year={1992},
  organization={ACM}
}
```

Summary

This paper explores monads as a technique for structuring functional programs that produce effects. It shows multiple examples where a monad can be used to implement a certain imperative feature, and relates monads to CPS.

Evaluation

This is an elementary introduction to monads in functional programming. Most of its ideas were already present in Moggi's earlier paper (see below). However, Wadler describes them in the setting of a real world programming language and emphasizes the pragmatic aspects of implementing and using monads.

2 Notions of computation and monads

```
@article{m:notions,
  title={Notions of computation and monads},
  author={Moggi, Eugenio},
```

```
  journal={Information and computation},
  volume={93},
  number={1},
  pages={55--92},
  year={1991},
  publisher={Academic Press}
}
```

Summary

In this paper, Moggi gives a calculus for effectful programming based on category theory. Monads are the main structured element for modeling effectful computations.

Evaluation

This is a revised version of Moggi's 1989 paper *Computational lambda-calculus and monads*, which was the very first paper that relates the concept of monads from category theory with programming. This paper gives a method for modeling programs as composition of effectful computations rather than just pure functions. Features like non-determinism, exceptions, side-effects, continuations etc. are mentioned in this paper, which can be regarded as the foundation of later works on introducing monads to pure functional programming languages.

3 How to declare an imperative

```
@article{w:imperative,
  title={How to declare an imperative},
  author={Wadler, Philip},
  journal={ACM Computing Surveys (CSUR)},
  volume={29},
  number={3},
  pages={240--263},
  year={1997},
  publisher={ACM}
}
```

Summary

This paper gives a solution to describing imperative and interactive behaviors in a pure language based on a monad, namely the IO monad. The first part of the

paper is a detailed explanation of the mechanism of the IO monad and its uses. In the second part, by comparing monads with four other approaches, Wadler shows that monads can ease the composition of interactions and achieve higher simplicity and flexibility over earlier solutions.

Evaluation

This tutorial focuses on how to manage interactive behaviors in a pure functional language — a key concern for those who are new to such a language. Wadler presents specific examples with detailed explanation to show how monads work. Moreover, in the comparison part, Wadler discusses "synchronized stream", a method used in early versions of Haskell. This is also a reflection on Wadler's previous work on the design of Haskell in the early 1990s.