Purely Functional Effect-Handling

Henry Blanchette

Contents

Chapter 1

Introduction

1.1 Outline

1. Definition of a simple, typed lambda calculus
2. Context and definition of what *effects*
3. Explanation of how effects are handle'd in ML's style i.e. the program is strictly evaluated and effectual functions are treated like normal functions
4. Explain how this style is not strictly functional, because it requires a globally-mutable and globall-accessible state that is carried along implicitly at run-time

Chapter 2

Monadic Effects

2.1 Outline

1. Definition of monads
2. Explanation of how monads can model effects in general
3. Demonstration of the stateful monad

Chapter 3

Algebraic Effect Handlers

3.1 Outline

1. Definition of algebraic effect handlers
2. example of implementation in Eff programming langauge
   (a) subset of semantics
   (b) examples
3. comparison of effect handlers to the usual monadic approach
   (a) effect handlers allow for the effectual program and the implementations of the actual effects of the program to be defined separately
   (b) effect handlers directly allow for arbitrary compositions and stacking, given a certain framework for effect stacks, but monads to not generally compose or stack in this way

Chapter 4

Freer Monadic Effects

4.1 Outline

1. Consideration of the problem of *composing monads*
2. Explanation of the monad-transformer approach, with pros and cons
3. Definition of freer monads
4. Explanation of how freer monads can model effects, by acting as a sort of monadic implementation of algebraic effects
5. Demonstration of stateful freer monad, paralleling monadic example

Chapter 5

Effigy

1. Explanation of motivations for a new language, incorporating the ideas of previous chapters
2. Specification of new language
3. Demonstration of new language