Programming with Effect Handlers in Links

PLInG, Autumn 2015, University of Edinburgh

Daniel Hillerström
daniel.hillerstrom@ed.ac.uk
http://homepages.inf.ed.ac.uk/s1467124/

November 3, 2015

School of Informatics
The University of Edinburgh

What this talk is about

Handlers for algebraic effects provide a compelling alternative to monads as a basis for effectful programming.

- **Key idea:** Separate effect signatures from their implementation.
- **Consequent:** High-degree of modularity.

Definitions will follow later...

Programs are inherently effectful

Programs may...

- ...halt prematurely
- ...diverge
- ...be stateful (e.g. modify a global state)
- ...communicate via a network
- ...print to standard out

A pure[1] program is not much fun.

[1] By pure we mean a program that has no effects.

Effectful computations (I)

$f : \mathbb{Z} \to \mathbb{Z}$	Mathematical pure function
$\text{int } f(\text{int } x)$	C/C++ (impure) function
$f : \text{int} \to \text{int}$	ML (impure) function

Let's be explicit about effects

Effect annotation

An effect annotation gives a static description of the potential run-time behaviour of a computation.

Benefits

- Serves as documentation (clarity)
- Compiler can apply specific optimisations
- Possible to reason more precisely about programs

Enter the Monad

"Have you considered using a monad?"

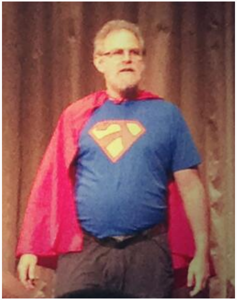


Figure 1: Philip Wadler aka. Lambda Man

- The Essence of Functional Programming (1992)
- The Marriage of Effects and Monads (1998/2003)

Effectful computations (II)

$f : \mathbb{Z} \to \mathbb{Z}$	Mathematical pure function
int f(int x)	C/C++ (impure) function
f : int $\to$ int	ML (impure) function
f : Int $\to$ IO Int	Haskell effect annotation

Effectful computations (II)

$f : \mathbb{Z} \to \mathbb{Z}$	Mathematical pure function
int f(int x)	C/C++ (impure) function
f : int $\to$ int	ML (impure) function
f : Int $\to$ IO Int	Haskell effect annotation

IO can be considered an effect annotation

Effectful computations (II)

$f : \mathbb{Z} \to \mathbb{Z}$	Mathematical pure function
int f(int x)	C/C++ (impure) function
f : int $\to$ int	ML (impure) function
f : Int $\to$ IO Int	Haskell effect annotation
int f(int x) throws Exception	Java effect annotation

Great! Many monads, many effects

A couple of monads and their "effect interpretation"

IO a	May perform I/O, returns a
Reader r a	May read from r, returns a
Writer w a	May write to w, returns a
State s a	May read/write some state s, returns a
Maybe a	May fail, returns a on success

Sadly, monads do not compose well.

Algebraic effects

Figure 2: Gordon Plotkin

Figure 3: John Power

Adequacy for Algebraic Effects (2001)

Algebraic effects and computations

Definition (Algebraic effect)

An algebraic effect is a collection of abstract operations. For example $\mathit{Choice} \stackrel{\text{def}}{=} \{\mathit{Choose} : \mathsf{Bool}\}$.

Definition (Abstract computation)

An abstract computation is composed from abstract operations.

Effect handlers

Figure 4: Gordon Plotkin

Figure 5: Matija Pretnar

Handling Algebraic Effects (2003)

Effect handler

Definition (Handler)

A handler interprets an abstract computation.

An example

Essentially, a handler pattern-matches on operations occurring in a computation m:

```
handler choice(m) {
  case Choose(k) -> ...
  case Return(x) -> ...
}
```

WARNING

The following code examples may be reproduced at home.

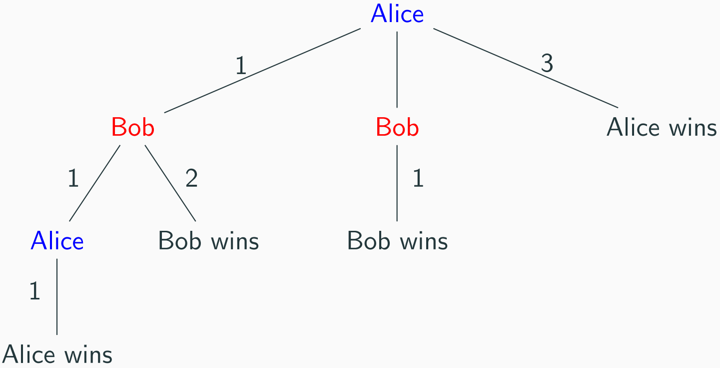
Nim: A game with sticks

Set-up

- Two players: Alice and Bob; Alice always starts.
- One heap of n sticks.
- Turn-based. Each player take between 1-3 sticks.
- The one, who takes the last stick, wins.

We'll demonstrate how to encode strategic behaviour, compute game data, and cheat using handlers.

Game tree generated by `mtGen` with $n = 3$



References I

Gordon D. Plotkin and Matija Pretnar.
Handling algebraic effects.
Logical Methods in Computer Science, 9(4), 2013.

Philip Wadler, Sam Lindley, Garrett Morris, et al.
Links: Linking theory to practice for the web.
`http://groups.inf.ed.ac.uk/links/`, 2005.

Ohad Kammar, Sam Lindley, and Nicolas Oury.
Handlers in action.
In *ICFP'13*, pages 145–158, 2013.

References II

Sam Lindley.
Algebraic effects and effect handlers for idioms and arrows.
In *Proceedings of the 10th ACM SIGPLAN workshop on Generic programming, WGP 2014, Gothenburg, Sweden, August 31, 2014*, pages 47–58, 2014.

Philip Wadler and Peter Thiemann.
The marriage of effects and monads.
ACM Trans. Comput. Log., 4(1):1–32, 2003.

Philip Wadler.
The essence of functional programming.
In *Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '92, pages 1–14, New York, NY, USA, 1992. ACM.

References III

Gordon D. Plotkin and John Power.
Adequacy for algebraic effects.
In *Foundations of Software Science and Computation Structures, 4th International Conference, FOSSACS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, pages 1–24, 2001.