On the relation of aspects and monads

Christian Hofer and Klaus Ostermann

"Since I've already got monads [...] I don't see any advantage to OOP plus AOP"
(Shae Erisson on Lambda the Ultimate)

What is the link between aspects and monads?

Outline

- What are monads?
- Using monads to implement cross-cutting concerns
- Monads as aspects of computations

What are monads?

- Not every problem apt for functional decomposition
  - non-deterministic computations
  - computations that may fail
  - stateful computations
  - ...

What are monads?

- Example: Maybe monad

```
eqVal key1 key2 db =
    do val1 <- lookup key1 db
       val2 <- lookup key2 db
       return (val1 == val2)
```

```
data Maybe a = Just a | Nothing
```

What are monads?

- *Monads*
  - abstracting over kinds of computations
  - by hiding the parameter structure
- *Monad transformers*
  - to combine different kinds of computations in a modular way

Monads for AOP

- Monads as natural mechanism to handle cross-cutting concerns in functional programming
- Display update example:

```
movePointBy :: Point -> Int -> Int ->
               IO ()
movePointBy (Point p) dx dy =

      modifyIORef p (\(x,y) -> (x+dx,y+dy))
```

Monads for AOP

- Crosscutting concern: trigger display refresh
- Hides the parameter structure of display aspect

```
movePointBy :: Point -> Int -> Int ->
                  StateT DisplayInfo IO ()
movePointBy (Point p) dx dy =
    withDisplayRefresh
      (modifyIORef p (\(x,y) -> (x+dx,y+dy)))
```

Monads for AOP

- Is this AOP?
  - Adding action before, after, around points in program execution? Yes!
  - Static quantification? No!
  - Obliviousness? No!
  - Dynamic quantification? Yes!
    e.g.: control flow quantification & reader monad
  - Declarativeness? No!

Monads for AOP

- Comparable to annotation-based AOP
  - but not declarative!
- Advantage of monads:
  - *Confining* concerns: restricting access

Monads as aspects of computations

- Can AOP represent those aspects?
  - no way to redefine sequencing of operations
  - only interfering at specific points
- Thus: aspects in general *cannot*
  - manipulate control flow
  - enrich parameter structure

Monads as aspects of computations

- Aspects cannot capture essence of different kinds of computations
- But:
  - Dynamic quantification can be used to simulate their behavior
  - Aspects might be better able to separate them