Programming With Algebra

@bkyrlach
bkyrlach@gmail.com

What is FP?

- Programming with functions?

- Avoiding side effects?

- Separating data and behavior?

Algebra

- From wiki "…a specific mathematical structure…having addition, multiplication, and scalar multiplication"

- Or, from Universal Algebra "…a set A together with a collection of operations on A…"

Why Algebra?

- Provides structure/guidance

- Free stuff!

- Vindication for your high-school math teacher (sorry, mom!)

Robot Algebra 101

- Robots have state

- Robots accept commands

- Robots can either succeed or fail at executing a command

Robot Algebra 101

- Limitations

- Difficulty in executing sequential commands

- Validation is hard

- Lots of repeated code

Monoid

- mappend :: a -> a -> a

- mempty :: a

Monoid Laws

- mappend mempty x = x

- mappend x (mappend y z) = mappend (mappend x y) z

Noop & AndThen

- Noop is our identity value

- AndThen combines two commands

Improve Error Handling

- Commands should "short circuit" if earlier commands (or command internals) fail

- Programmer shouldn't have to think about it

- Problem: s -> (m a, s) sucks

Monad

- pure/return: a -> M a

- bind: M a -> (a -> M b) -> M b

- fmap: M a -> (a -> b) -> M b

- Intuition: Do I have a parametric constructor over a single type?

Monad Laws

- return a >>= f ≡ f a

- m >>= return ≡ m

- (m >>= f) >>= g ≡ m >>= (\x -> f x >>= g)

Monads all the way Down

- Think of the partially applied type ((->) a)

- You're right, we can define a monad for it!

- What about (s -> (a, s)?

- Factor out s, and it's the same thing.

Transformers: Monads In Disguise

- CommandOutcome could be a monad

- Starting out less abstract actually hurt us

- Use monad transformer to combine with State for robot magic

To Infinity and Beyond

- Just scratching the surface

- Alternative, MonadPlus, and much, much more…