Why I use Haskell

Eric Kow

http://erickow.com/talks/5-min-haskell.pdf

2011-05-17

Things I care most about

1. Problem solving/identification:
   How do I solve X? What is X in the first place?
2. Correctness
   How do I solve X with as few bugs as possible?
3. Clarity
   Will Future-Eric understand this code? Will my colleagues?

Types

1. Problem solving: think about why types I need
   Easier for me than thinking about object hierarchies
2. Problem solving: sum types = thinking in terms of cases
3. Clarity - types == documentation
   I can sort of tell what a function does by its type signature
4. Correctness
   Compiler catches when I contradict myself

Types 2

Example

```
data Compound = And Compound Compound
              | Or Compound Compound
              | Product Compound Compound
              | Not Compound
              | Leaf Unitary
   deriving (Show, Eq)
```

Code that works with Compound must account for all cases.

Higher order functions

A higher order function is a function that takes function(s) as an argument

1. Easier to reuse my code!
2. Very general purpose traversals

Higher order function example

```
traverse :: (Unitary -> Compound) -> Compound -> Compound
traverse f (And x y) = And (traverse f x) (traverse f y)
traverse f (Or x  y) = Or  (traverse f x) (traverse f y)
traverse f (Product x y) = Product (traverse f x)
                                   (traverse f y)
traverse f (Not x) = Not (traverse f x)
traverse f (Leaf x) = Leaf (f x)
```

Immutability

1. Solve small problems (functions)
2. Combine them with powerful tools (higher order functions)
3. Not worry about side effects!