PLClub Homework for November 4th

Yao Li

October 31, 2016

Contents

0. Get yourself familiar with the concepts of algebraic types, type classes, functors, if you haven't already. You don't need to fully understand these concepts, but having some intuition of what they are would help you understand the talk better. I, personally, find the following materials helpful:

   - Algebraic Data Types:
     `http://learnyouahaskell.com/making-our-own-types-and-typeclasses#algebraic-data-types`
   - Typeclasses:
     `http://learnyouahaskell.com/making-our-own-types-and-typeclasses#typeclasses-102`
   - Functors:
     `http://learnyouahaskell.com/making-our-own-types-and-typeclasses#the-functor-typeclass`

1. Read the following program (a binary search tree) written in Haskell, and think about how you would implement it in a statically typed object-oriented programming language, for example, Java:

```
data Tree a = Empty | Node (Tree a) a (Tree a)
  deriving Show

insert :: (Eq a, Ord a) => a -> Tree a -> Tree a
insert x Empty = Node Empty x Empty
insert x (Node l v r)
  | x >  v = Node l v (insert x r)
```

```
  | x == v = Node l v r
  | x <  v = Node (insert x l) v r
```

Notice that we don't really need algebraic types to implement this example in Java. What's the problem if we don't use algebraic types here?

You don't need to actually implement this program in Java, but feel free to send me a copy of your code if you did.

2. You may have used subtype polymorphism to implement the above example in Java, because most comparable stuff in Java would implement the `Comparable` interface. Now suppose we have a `Nat` data type from a third-party library, which, for some reason, does not implement the `Comparable` interface. How would you use this `Nat` data type on your binary search tree?

   You don't need to actually implement this program in Java, but feel free to send me a copy of your code if you did.