Programming Paradigms 2022
Session 9: Functors

Problems for solving and discussing

Hans Hüttel

8 November 2022

Problems that we will definitely talk about

1. ***(Everyone at the table together – 20 minutes)***

   The type of unbounded trees UTree is given by

   ```
   data UTree a = Node a [UTree a]
   ```

   Define an instance of Functor for UTree.

2. ***(Work in pairs – 15 minutes)***

   The function type constructor ((->)r) is defined such that f a will be (r -> a).

   Define an instance of Functor for this type constructor.

3. ***(Everyone at the table together – 15 minutes)***

   For the applicative functor for lists we have a definion of the "funny star" composition <*> on page 160. Give an alternative *recursive* definition of it that uses fmap.

4. ***(Work in pairs – 20 minutes)***

   Here is an expression e in the applied λ-calculus. Note

   $$\lambda y : \mathsf{Int}.\mathsf{let}\ z : \mathsf{Int}\ = 17\ \mathsf{in}\ (x = \mathsf{plus}\ z\ y)$$

   Note that e has a free variable, x. What is the type t of the expression e? Make a qualified guess.

   After that, assuming that the type environment E is $x : \mathsf{Int}$ show that $E \vdash e : t$ by building a derivation tree for this type judgement.

More problems to solve at your own pace

a) Here is a type declaration for simple expressions.

```
data Exp a = Var a | Val Int | Add (Expr a) (Expr a)
```

Show how do make this type into an instance of Functor.

When would it be useful to think of Exp a as a functor? Think of a good example!

b) Show how to make the type Exp from the previous problem into an instance of Applicative.