Programming Paradigms 2022
Session 13 : Reasoning about programs

Hans Hüttel

6 December 2022

Our plan for today

1. The learning goals
2. Presentations of the discussion problems
3. Problem no. 1
4. Problem no. 2
5. Break
6. Problem no. 3
7. If time allows: More problems at your own pace.
8. We evaluate the course as a whole

Learning goals

- To be able to use induction on the natural numbers to reason about Haskell programs
- To be able to use induction on lists to reason about Haskell programs
- To be able to use induction to prove properties of functors and other structures

Discussion problem – How long is a reversed list?

Prove by induction on lists that

$$\text{length } (xs \mathbin{+\!\!+} ys) = \text{length } xs + \text{length } ys$$

Discussion problem – Tuesday bingo

On page 240 we see the definition of the flatten function. Define the function

```
bingo :: Tree -> Integer
```

by

```
bingo (Leaf n) = 1
bingo (Node l r) = bingo l + bingo r
```

and explain what it computes. Prove by induction on trees that

```
length (flatten t) = bingo t
```

Problem 1 – Everyone at the table together (10 minutes)

Prove by induction on lists that

$$\text{length}\ (\text{reverse}\ xs) = \text{length}\ xs$$

(One of the discussion problems from today is your friend.)

Problem 2 – Work in pairs (30 minutes)

Here is our type declaration for binary trees with a-labelled trees and a declaration that this type as a functor.

```
data Tree a = Leaf a | Node (Tree a) (Tree
    a)

instance Functor Tree where
-- fmap :: (a -> b) -> Tree a -> Tree b
fmap g (Leaf x) = Leaf (g x)
fmap g (Node l r) = Node (fmap g l) (fmap
   g r)
```

Prove by induction on trees that the two functor laws for this type hold (you need to look up these laws in the textbook!).

Problem 3 – Everyone at the table (30 minutes)

Below is the usual definition of a function defining the Fibonacci numbers, now written in Haskell.

```
fib 0 = 1
fib 1 = 1
fib n = fib (n-1) + fib (n-2)
```

Prove by induction that if $n > 1$ then fib $n \geq \phi^{n-2}$, where $\phi = \frac{1+\sqrt{5}}{2}$. It is useful to remember that $\phi^2 = 1 + \phi$ (you may want to check this).

Next, prove by induction that one needs fib n recursive calls to find fib n if $n \geq 2$. What does this tell us about the Haskell implementation of the Fibonacci numbers? (Try finding fib 50.)