Workshop Testing and Formal Methods, Week 1

September 4, 2017

```
> module Workshop1 where
> import Data.List
```

This workshop is about proving things by induction, and about the connection between recursion and induction. More precisely, this workshop is about how you prove things by induction on data-types defined by recursion. Background reading for this topic is The Haskell Road, chapter 7.

For further instruction on this topic from YouTube, have a look at this Khan Academy video.

The simplest case of proof by (mathematical) induction is induction on the natural numbers. The task is to prove that a property P holds of all natural number. The proof has two steps:

1. Show that the property P holds for the number 0.
2. Show that if the property P holds for the number n, then it will also hold for $n + 1$.

In the second case, the assumption that P holds for n is called the **induction hypothesis**. The best way to learn proof by induction is by practice. Here we go. (And again: watch the video mentioned above.)

1.

Prove by induction that it holds for all natural numbers n that

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

2.

Prove by induction that it holds for all natural numbers n that

$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}.$$

3.

Prove by induction that it holds for all natural numbers n that

$$1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2.$$

4.

Prove by induction that if A is a finite set with $|A| = n$, then $|\mathcal{P}(A)| = 2^n$.

5.

A permutation of a list is a reordering of the members of a list. Here is a Haskell implementation:

```
> perms :: [a] ->[[a]]
> perms [] = [[]]
> perms (x:xs) = concat (map (insrt x) (perms xs)) where
>   insrt x [] = [[x]]
>   insrt x (y:ys) = (x:y:ys) : map (y:) (insrt x ys)
```

You can also use the Data.List function `permutations`. Find a formula (closed form) for the number of permutations of a list of n distinct objects, and prove your guess by induction.

6.

Prove by induction that it holds for all natural numbers n that

$$3^{2n+3} + 2^n \text{ is divisible by } 7.$$

It is not necessary to have 0 as base case. Here are some examples where the base case is a different number.

7.

Show by induction that for all natural numbers n with $n \geq 3$ it holds that $n^2 > 2n$.

8.

Show by induction that for all natural numbers n with $n \geq 5$ it holds that $2^n > n^2$.

9.

Consider the following game for two players. Starting situation: a number of matches is on a stack. The players take turns. A move consists in removing 1, 2 or 3 matches from the stack. The player who can make the last move (the move that leaves the stack empty) has won the game. Suppose there are $4N$ matches on the stack, and the other player moves. How should you respond? Prove by induction that your strategy assures that you will win the game.

A useful generalisation of mathematical induction on the natural numbers is **structural induction**, where we wish to prove that some property P holds of all members of a recursively defined datatype such as trees, lists or formulas.

10.

Recall the definition of formulas of propositional logic:

$$\phi ::= p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\phi \leftrightarrow \phi)$$

Give a recursive definition of the number of occurrences of connectives in a propositional formula.

11.

Give a recursive definition of the number of occurrences of atoms (atomic subformulas) in a formula of propositional logic.

12.

Let $S(\phi)$ be the number of occurrences of subformulas of a propositional formula ϕ, let $A(\phi)$ be the number of atoms of ϕ, and let $C(\phi)$ be the number of connectives of ϕ. Prove by structural induction:

$$S(\phi) = A(\phi) + C(\phi), \text{ for all propositional formulas } \phi.$$

Consider the following definition of binary trees:

```
> data Btree a = Leaf a | Node (Btree a) (Btree a) deriving (Eq,Show)
```

The depth of a binary tree is given by:

```
> depth :: Btree a -> Int
> depth (Leaf _) = 0
> depth (Node t1 t2) = max (depth t1) (depth t2) + 1
```

13.

What is the *minimum* number of internal nodes (non leaf nodes) that can occur in a binary tree of depth n? First guess, by looking at examples. Next prove your guess by induction.

14.

What is the *minimum* number of leaf nodes that can occur in a binary tree of depth n? First guess, by looking at examples. Next prove your guess by induction.

15.

What is the *maximum* number of internal nodes (non leaf nodes) that can occur in a binary tree of depth n. First guess, by looking at examples. Next prove your guess by induction.

16.

What is the *maximum* number of leaf nodes that can occur in a binary tree of depth n. First guess, by looking at examples. Next prove your guess by induction.

17.

Consider the following program for merging two lists:

```haskell
> merge :: Ord a => [a] -> [a] -> [a]
> merge xs [] = xs
> merge [] ys = ys
> merge (x:xs) (y:ys) = if x <= y
>                       then x : merge xs (y:ys)
>                       else y : merge (x:xs) ys
```

Show with induction that if `xs` and `ys` are finite and sorted, then `merge xs ys` is sorted.