Haskell Exercises 10: Proofs by Induction

Antoni Diller

26 July 2011

1 Mathematical Induction Proofs

(1) Prove, using mathematical induction, that, for all $n \geq 1$,

$$\sum_{i=0}^{n-1} 2i + 1 = n^2.$$

(2) Prove, using mathematical induction, that, for all $n \geq 1$,

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1.$$

(3) Prove, using mathematical induction, that, for all $n \geq 1$,

$$\sum_{i=0}^{n-1} 3^i = \frac{3^n - 1}{2}.$$

(4) Prove, using mathematical induction, that, for all $n \geq 1$,

$$\sum_{i=0}^{n-1} 1^2 = \frac{n(n+1)(2n+1)}{6}.$$

(5) Prove, using mathematical induction, that, for all $n \geq 1$,

$$\sum_{i=0}^{n-1} i \times 2^i = (n-1) \times 2^{n+1} + 2.$$

(6) Prove, using mathematical induction, that, for all $n \geq 3$, $2n + 1 < 2^n$.

2 Structural Induction Proofs

In the following proofs use the following definitions:

```
(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x : (xs ++ ys)

concat :: [[a]] -> [a]
concat []       = []
concat (xs:xss) = xs ++ concat xss

reverse :: [a] -> [a]
reverse []     = []
reverse (x:xs) = reverse xs ++ [x]

map :: (a -> b) -> [a] -> [b]
map f []     = []
map f (x:xs) = f x : map f xs

(.) :: (b -> c) -> (a -> b) -> a -> c
(f.g) x = f (g x)

filter :: (a -> Bool) -> [a] -> [a]
filter p []   = []
filter p (x:xs)
  | p x       = x : filter p xs
  | otherwise = filter p xs

foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f e [] = e
foldr f e (x:xs) = f x (foldr f e xs)

foldl :: (b -> a -> b) -> b -> [a] -> b
foldl f e [] = e
foldl f e (x:xs) = foldl f (f e x) xs

flip :: (b -> a -> c) -> a -> b -> c
flip f x y = f y x
```

(7) Prove, using structural induction, that

```
xs ++ [] = xs
```

(8) Prove, using structural induction, that

```
concat (xss ++ yss) = concat xss ++ concat yss
```

(9) Prove, using structural induction, that

```
reverse (xs ++ ys) = reverse ys ++ reverse xs
```

(10) Prove, using structural induction, that

```
map (f.g) = map f . map g
```

(11) Prove, using structural induction, that

```
filter p . concat = concat . map (filter p)
```

(12) Prove, using structural induction, that

```
foldr f e xs = foldl (flip f) e (reverse xs)
```

(13) Prove, using structural induction, that

```
foldr f e . concat = foldr (flip (foldr f)) e
```

(14) Prove, using structural induction, that

```
foldr f e . concat = foldr f a . map (foldr f e)
```