Haskell Exercises 6: *fold* functions

Antoni Diller

26 July 2011

(1) Using the higher-order function *foldr* define a function *sumsq* which takes an integer n as its argument and returns the sum of the squares of the first n integers. That is to say,

$$sumsq\ n = 1^2 + 2^2 + 3^2 + \ldots + n^2.$$

Do not use the function *map*.

(2) Define *length*, which returns the number of elements in a list, using *foldr*. Redefine it using *foldl*.

(3) Define *minlist*, which returns the smallest integer in a non-empty list of integers, using *foldr1*. Redefine it using *foldl1*.

(4) Define *reverse*, which reverses a list, using *foldr*.

(5) Using *foldr*, define a function *remove* which takes two strings as its arguments and removes every letter from the second list that occurs in the first list. For example, `remove "first" "second" = "econd"`.

(6) Define *filter* using *foldr*. Define *filter* again using *foldl*.

(7) The function *remdups* removes adjacent duplicates from a list. For example,

$$remdups\ [1, 2, 2, 3, 3, 3, 1, 1] = [1, 2, 3, 1].$$

Define *remdups* using *foldr*. Give another definition using *foldl*.

(8) The function *inits* returns the list of all initial segments of a list. Thus, `inits "ate" = [[], "a", "at", "ate"]`. Define *inits* using *foldr*.

(9) Using *foldl* define *approxe* n such that

$$approxe\ n = \sum_{i=0}^{i=n} \frac{1}{i!}.$$

For example,

$$\begin{aligned} \mathit{approxe}\ 4 &= \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!}, \\ &= 1 + 1 + 0.5 + 0.\dot{3} + 0.25, \\ &= 3.08\dot{3}, \end{aligned}$$

where $0.\dot{3}$ means 'point 3 recurring'.

(10) Using *scanl* define a function *sae* (successive approximations to e) such that

$$\mathit{sae}\ n = \left[\sum_{i=0}^{i=1} \frac{1}{i!}, \sum_{i=0}^{i=2} \frac{1}{i!}, \sum_{i=0}^{i=3} \frac{1}{i!}, \ldots, \sum_{i=0}^{i=n} \frac{1}{i!}\right].$$

(11) Define *iterate* using *scanl*.

(12) Define *shift*, which sticks the first element of a list at the end. Thus, `shift [1, 2, 3] = [2, 3, 1]` and `shift "eat" = "ate"`. Unsing *foldl* and *shift* define *rotate*, which produces all the rotations of a list. Thus, `rotate [1, 2, 3] = [[1, 2, 3], [2, 3, 1], [3, 1, 2]]`.

(13) The function *add* can be defined in terms of

```
succ i = i + 1
pred i = i - 1
```

by the equations

```
add i 0 = i
add i j = succ (add i (pred j))
```

(a) Give a similar definition of *mult* which uses only *add* and *prede*. Give a definition of *exp* which uses only *mult* and *prede*. What is the next function in this sequence?

(b) The *fold* function on integers *foldi* can be defined as follows:

```
foldi :: (a -> a) -> a -> Int -> a
foldi f q 0 = q
foldi f q i = f (foldi f q (pred i))
```

Define the functions *add*, *mult* and *exp* in terms of *foldi*.

(c) Define the functions *fact* (factorial) and *fib* (Fibonacci numbers) using the function *foldi*.