Haskell exercise: Polynomials

Department of Mathematics and Computer Science
University of Southern Denmark

September 15, 2017

In this exercise, we are going to implement the mathematical concept of a polynomial in Haskell, and relevant functions regarding this.

A polynomial $p : \mathbb{R} \to \mathbb{R}$ with degree n is a function $p(x) = a_0x^0 + a_1x^1 + \ldots + a_nx^n$ where $a_0 \ldots a_n$ are constants in $\mathbb{R}$, $a_n \neq 0$.

To represent a polynomial in Haskell, we use lists. More precisely, we let the list [*a0*, *a1* ... *an*] correspond to the polynomial $a_0 + a_1x + \ldots a_nx^n$.

Examples

- $5 + 2x + 3x^2$ is represented by $[5, 2, 3]$
- $-2 + x^2$ is represented by $[-2, 0, 1]$
- 0 is represented by []

Exercises

1. We discover that a polynomial can be represented by infinitely many lists - e.g. $-2 + x^2$ can be represented by

   $$[-2, 0, 1], [-2, 0, 1, 0], [-2, 0, 1, 0, 0], [-2, 0, 1, 0, 0, 0] \ldots.$$

   We will therefore require the list representation of a polynomial not to end with a 0. We call this the *canonical* list representation of the polynomial. Define a function

   $canonical :: (Num\ a, Eq\ a) \Rightarrow [\,a\,] \to [\,a\,]$

which removes all zeros from the end of the input list. Your other functions are expected only to output polynomials in canonical form. Your other functions are not expected to work on polynomials which are not in canonical form.

Examples:

- $\mathit{canonical}\ [0,0,0] \equiv [\,]$
- $\mathit{canonical}\ [1,2,3] \equiv [1,2,3]$
- $\mathit{canonical}\ [0,0,2,0] \equiv [0,0,2]$
- $\mathit{canonical}\ [0,1,2,0,0] \equiv [0,1,2]$
- $\mathit{canonical}\ [1,2,0,1] \equiv [1,2,0,1]$

2. Define the function
   $\mathit{deg} :: [\,a\,] \to \mathit{Int}$
   which outputs the degree of a polynomial. We define the zero polynomial $z(x) = 0$ (represented by the empty list) to have degree -1. HINT: Use the *length* function. You can assume that the input polynomial is in canonical form.

3. Define the function
   $\mathit{lead} :: \mathit{Num}\ a \Rightarrow [\,a\,] \to a$
   which for a polynomial $p(x) = a_0x^0 + a_1x^1 + \ldots + a_nx^n$ outputs a_n. In terms of the list representation, *lead* returns the last element of the list (assuming the input list is in canonical form). If the given list is empty, return 0.

4. Define the function
   $\mathit{neg} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \to [\,a\,]$ which negates each element in the input list.
   Examples:

   - $\mathit{neg}\ [1,0,2] \equiv [-1,0,-2]$,
   - $\mathit{neg}\ [\,] \equiv [\,]$,
   - $\mathit{neg}\ [-1,0,2] \equiv [1,0,-2]$.

5. Implement the function
   $\mathit{add} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \to [\,a\,] \to [\,a\,]$
   which implements addition of polynomials.
   Examples:

- $\mathit{add}\ [5,2,3]\ [-2,1] = [5+(-2), 2+1, 3] = [3,3,3]$
- $\mathit{add}\ [1,2,3]\ [0,-2,-3] \equiv [1]$

HINT: Consider modifying *zip*. You need to apply *canonical* to the result (see the second example above)

6. Implement $\mathit{sub} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \to [\,a\,] \to [\,a\,]$
   which implements subtraction of polynomials (HINT: Use *add* and *neg*).

7. Implement $\mathit{addMany} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [[\,a\,]] \to [\,a\,]$ which adds a list of polynomials.

   Example:

   - $\mathit{addMany}\ [\,] = [\,]$
   - $\mathit{addMany}\ [[1],[1,2],[1,2,3]] \equiv [3,4,3]$

8. Implement $\mathit{mulconstant} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow a \to [\,a\,] \to [\,a\,]$ which multiplies each element of the list with a constant, i.e.
   $\mathit{mulconstant}\ a\ [\,\mathit{b1}, \mathit{b2}, ...\mathit{bn}\,] \equiv [\,a * \mathit{b1}, a * \mathit{b2}, ..., a * \mathit{bn}\,]$

9. Implement $\mathit{mulpower} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow \mathit{Int} \to [\,a\,] \to [\,a\,]$, for which $\mathit{mulpower}\ i\ \mathit{xs}$ appends i zeros to the beginning of the list xs. (with the exception, that if xs is empty, the output list should be empty as well)

10. Implement the functions
    $\mathit{diff} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \to [\,a\,]$
    $\mathit{int} :: \mathit{Fractional}\ a \Rightarrow [\,a\,] \to [\,a\,]$
    which implements differentiation and integration of polynomials.

    In terms of the list representation,

    - $\mathit{diff}\ [\,\mathit{a0}, \mathit{a1}, \mathit{a2}\ ...\ \mathit{an}\,] \equiv [\,\mathit{a1} * 1, \mathit{a2} * 2, \mathit{a3} * 3, ...\mathit{an} * n\,]$
    - $\mathit{int}\ [\,\mathit{a0}, \mathit{a1}, \mathit{a2}\ ...\ \mathit{an}\,] \equiv [0, \mathit{a1}\ /\ 1, \mathit{a2}\ /\ 2, \mathit{a3}\ /\ 3, ...\mathit{an}\ /\ n\,]$

11. Implement the function
    $\mathit{mul} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \rightarrow [\,a\,] \rightarrow [\,a\,]$
    which implements multiplication of two polynomials.

    HINT: Consider the following implementation strategy:

    ```
    $mul\ [\,a0, a1\ ...\ an\,]\ bs \equiv addMany\ [$
      $mulpower\ 0\ (mulconstant\ a0\ bs),$
      $mulpower\ 1\ (mulconstant\ a1\ bs),$
      $...,$
      $mulpower\ n\ (mulconstant\ an\ bs)$
      $]$
    ```

12. Implement the function
    $\mathit{eval} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \rightarrow a \rightarrow a$

    which implements evaluation of a polynomial, i.e. *eval p x0* corresponds to the value $p(x_0)$ where $x_0 \in \mathbb{R}$.

    HINT: Consider applying Horner's Rule, i.e. $\mathit{eval}\ [\,a0, a1, ...an\,]\ x = a0 + x * (a1 + x * (a2 + ... + x * (a\ \{n-1\} + x * an)...))$

13. Implement the function
    $\mathit{compose} :: (\mathit{Num}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \rightarrow [\,a\,] \rightarrow [\,a\,]$
    which implements composition of two polynomials, i.e. for polynomials $p(x), q(x)$, output $p \circ q(x) = p(q(x))$.

    HINT: *compose p q* can be seen as a generalization of *eval p x0*, where the polynomial p is not evaluated at a point *x0* but instead at a polynomial q.

14. Implement the function
    $\mathit{polydiv} :: (\mathit{Fractional}\ a, \mathit{Eq}\ a) \Rightarrow [\,a\,] \rightarrow ([\,a\,], [\,a\,]) \rightarrow ([\,a\,], [\,a\,])$
    which implements polynomial long division, i.e. for input polynomials - the divisor $d(x)$ and the divident $p(x)$ it outputs the quotient $q(x)$ and remainder $r(x)$ (polynomials satisfying $p(x) = d(x)q(x) + r(x)$) - in Haskell notation: $\mathit{polydiv}\ d\ (\mathit{zero}, p) = (q, r)$.

    Refer to `http://en.wikipedia.org/wiki/Polynomial_long_division` for pseudo-code for the algorithm.