List Manipulation

Michael P. Fourman

October 29, 2006

Aims

In this practical, you will learn to use lists.

Assessment

Your work will be assessed on the basis of the correctness of the two structures you implement. These should be placed in the files `CS201/Prac3/Poly.ML` and `CS201/Prac3/SparsePoly.ML` under your home directory. This time, 70% of the marks will be allotted to the first part, and 30% to the second.

Introduction

This practical continues the mathematical flavour of the first two practicals. This time, the exercise involves implementing a package for manipulating polnomials.

1 Polynomial Arithmetic

Consider the problem of performing arithmetic on polynomials over the integers. For simplicity, we will restrict ourselves to the most familiar type: polynomials having just one indeterminate. Examples of such polynomials include:

$$x^5 + 2x^4 + 3x^2 - 2x - 5 \qquad \text{and} \qquad x^{100} + 2x^2 + 1.$$

The first is an example of a dense polynomial, as it has non-zero coefficients for most powers of x, whereas the second example is sparse. At first, we will restrict our attention to dense polynomials. In such cases, a good representation for a polynomial is just a list of its coefficients. For example, listing the coefficients in ascending order of powers of x, the polynomial $x^5 + 2x^4 + 3x^2 - 2x - 5$ would be represented by the list `[ 5,  2, 3, 0, 2, 1]`. These examples have integer coefficients, but you are asked to implement polynomials with real coefficients.

Here is a signature specifying what you should implement.

```
infix 6 ++ ;
infix 7 ** ;

signature PolySig =
sig
    type Poly

    val ++ : Poly * Poly -> Poly
    val ** : Poly * Poly -> Poly

    val diff : Poly -> Poly (* derivative *)
    val int  : Poly -> Poly (* indefinite integral *)

    val eval : Poly -> real -> real (* evaluate *)
end;
```

As usual, this signature is provided, built-in, and the normal infix precedences have been set up for `++` and `**`, if you start your ML session with the command `ml prac3`.

The exercise involves doing two things:

1. Provide a structure `Poly:PolySig` based on the representation:

   ```
   type Poly = real list (* coefficient list, constant at head *)
   ```

   - Define functions for polynomial addition and multiplication.
   - Define a curried function `eval` that computes the result of subsituting a real for the indeterminate in a polynomial.
   - Define functions for differentiation and integration of polynomials.

2. The representation we have chosen is not very efficient for sparse polynomials. Here is a more efficient representation for such cases:

   ```
   type Poly = {coeff: real, power:int} list
   ```

Provide another implementation, `SparsePoly:PolySig`, using this represetation.