Programming Language and Translators
Project Proposal

Darrell Bethea
Michael G. Dougherty
Santosh Thammana
Mohit Chandru Vazirani

Our team will implement a language that enables the representation and manipulation of polynomials. Programs in the language will be written in ASCII text, with one language statement per line, with whitespace being used only to separate tokens and being otherwise ignored.

It will be statically typed, having three base types: Integer numbers ("int"), floating-point numbers ("float"), and a type representing the polynomials themselves ("poly"). The polynomial type will be represented in the language as a list of coefficients (floating point numbers) separated by commas, with the lowest order coefficients first. In a polynomial context, variables (which are polynomials themselves) can be inserted into a list of coefficients and will be replaced exactly by their own coefficients in the list. For example, if A and B are polynomial variables, and A equals 1,2,3, then "B = 6,7,A,8,9" will assign 6,7,1,2,3,8,9 to the variable B. The "let" keyword will signify the declaration of a new variable. Variables must be assigned an initial value when declared, thereby avoiding any risk of using uninitialized variables. Variable names will consist of one letter and are case-insensitive.

All iteration will be done via a "loop-endloop" block that will loop infinitely, unless a "breakloop" statement is reached. A "breakloop" statement will always shift control immediately just beyond the end of the most deeply nested loop currently running.

Conditional statements will utilize the built-in "if" construct to test the value of binary comparison operations on expressions. These operators ("=", "<", ">", and "!") can be used to compare numbers to numbers and polynomials to polynomials.

Inequality operations on pairs of polynomials will rank them by their relative asymptotic values.

Numbers support the arithmetic operations addition, subtraction, multiplication, and division. Polynomials can only be added or subtracted from one another. They can, however, be multiplied or divided by regular numbers.

There are two built-in operations on polynomials. First, expressions of the form "[A:i]" can extract a specific term (in this case, the i^{th} order term) of the polynomial stored in the variable A. Second, expressions of the form "|A|" will return the order of the polynomial stored in the variable A.

Lines beginning with "//" are ignored as comments.

Many interesting algorithms can be written in the language. For example, although polynomials cannot be multiplied amongst themselves automatically, a program can easily implement the algorithm to perform the operation using only the built-in language features. In addition, other algorithms that can be implemented include polynomial evaluation at a specific point, as well as differentiation and indefinite integration of polynomials. With the combination of polynomial evaluation and integration, a program can perform definite integration.

The following is a sample algorithm that assigns the indefinite integral of a polynomial A to a new polynomial B (with the constant of integration being fixed at 0):

```
// The variable A is an existing polynomial.
let poly B = 0
let int x = 0
let float z = 0
loop
   z = [A:x] / ( x + 1 )
   B = B,z
   if ( |A| = ( x + 1 ) {
      breakloop
   }
   x = x + 1
endloop
// The polynomial B now contains the indefinite integral of A.
```