CS39003 Compilers Laboratory, Autumn 2024–2025
Assignment No: 4
Date: 30-Sep-2024

Syntax-directed translation

In this assignment, we deal with polynomials in a single-variable x. A polynomial is written in the usual format as illustrated below:

–506x^7 + 9x^10 + x – 243015 – x^3 + 876 x^2 + 1

We write x^1 as x, and do not write x^0. If the coefficient of a term is 1 or –1, it is not explicitly written unless it is a constant term. We define a polynomial as a ***sum*** of non-zero (positive or negative) terms. The polynomial of the above example has seven terms: –506x^7, 9x^10, x, –243015, –x^3, 876x^2, and 1. A polynomial is allowed to have multiple terms with the same exponent of x (the above example shows two constant terms). You are not required to merge similar terms.

Grammar for polynomials

A grammar for polynomials is given now. Here, *S* is the start symbol, *P* stands for a polynomial with first coefficient positive, *T* is a term of the polynomial with positive coefficient, and *X* is a power of x. In order to put the above restrictions on coefficients and exponents, we use a terminal D to stand for the digits 2 through 9. Whenever needed, the digits 0 and 1 are handled separately. The non-terminal *N* expands to an integer ≥ 2 (without leading zeros), whereas the non-terminal *M* generates any sequence of all the digits 0 – 9, and stands for *N* without its most significant digit. The terminal symbols in the grammar below are printed in red.

S	⟶	*P* | **+** *P* | **–** *P*
P	⟶	*T* | *T* **+** *P* | *T* **–** *P*
T	⟶	**1** | *N* | *X* | *N X*
X	⟶	**x** | **x ^** *N*
N	⟶	**D** | **1** *M* | **D** *M*
M	⟶	**0** | **1** | **D** | **0** *M* | **1** *M* | **D** *M*

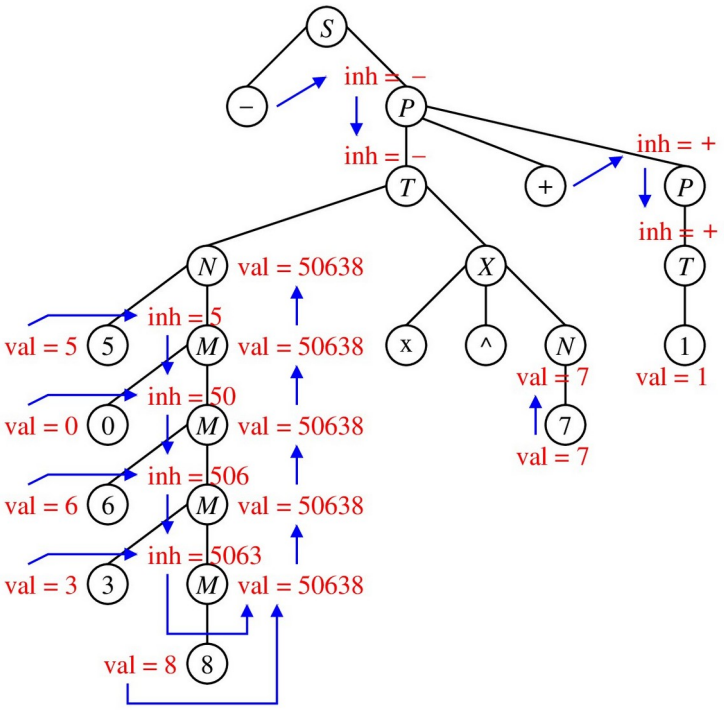
Note that in this grammar, the digits (like 9 and 1 in the above example) are terminal symbols. Multi-digit numbers (like 243015 or –506) are not. Strictly follow this grammar in this assignment. In particular, specify your lex patterns accordingly.

Attributes of the nodes in the parse tree

Each node in the parse tree has one inherited attribute *inh* and a synthesized attribute *val*. Not all nodes use these attributes.

- The leaf nodes standing for the terminal symbols +, –, x, and ^ use neither of these attributes.
- By definition, each term node *T* should store the sign of its coefficient. This is inherited from the *P* that creates it. Likewise, *P* inherits the sign from *S* or the *P* that generates it.
- Every leaf with a digit stores the corresponding integer value as its *val* attribute. The *inh* attribute is not used in a digit-type leaf node.
- Every number node *N* or *M* uses the attribute *val*. *M* additionally uses *inh*. The topmost *M* inherits the leftmost digit from its parent *N*. Each *M* node multiplies its inherited value by 10, and adds the next digit to it. Moreover, if that *M* has an *M* node as a child, then the value computed above is passed as the *inh* attribute of that child node. If not, the *val* attribute at that *M* node is set to the above computed value. This *val* then moves up the tree until it reaches the starting *N*.

The annotated parse tree and the dependency graph for the polynomial –50638 x^7 + 1 is shown to the right.



What you need to do

- Write a lex file *poly.l* for getting the terminal tokens. In this assignment, all terminals are single characters. Moreover, x is a placeholder, not an identifier to be stored in a symbol table. Here, you do not need to maintain any symbol table at all.

- Write a yacc/bison file *poly.y* to specify the grammar and the actions against each production, for creating the parse tree. Do **not** implement any function in the yacc file. Note also that the parse tree should be your data structure. You are **not** allowed to use any STL data types.

- Write a C/C++ file *polyutils.c* (or *cpp*) that performs the following tasks.

  (a) Implement the functions for creating the nodes in the parse tree. These functions are specified in the actions of the yacc file.

  (b) Write a function *setatt*( ) to set all the relevant attributes (inherited and synthesized) at the nodes of the parse tree, following the order prescribed by the dependency graph. There is no need to prepare that graph and/or implement topological sorting. A suitably implemented recursive function *setatt*( ) will solve our purpose.

  (c) Print the annotated parse tree in the format specified in the sample output. Write a recursive function to do this.

  (d) Write a recursive function *evalpoly*( ) that, given a numeric value (positive or negative integer or 0) of x, evaluates and returns the value of the polynomial with x substituted by the given numeric value. This function should use the inherited and synthesized attributes computed in Part (b).

  (e) Write a recursive function *printderivative*( ) that prints the derivative of the input polynomial. There is no need to simplify the polynomial. Write the derivatives of the terms in the same order as in the input. You must follow the format described at the beginning. For example, zero terms must not be printed. x^1 is printed as x. x^0 is not printed at all (only the corresponding coefficient is printed). Moreover, 1 is not printed as a coefficient unless it is the constant term.

  (f) Write a main() function to do the following. Call *yyparse*( ) to build the parse tree *PT*. Note that at this time, the attributes in the non-leaf nodes are not computed. The *val* attributes of digit-type leaf nodes may be set during parsing. Call *setatt*(*PT*) to annotate the parse tree *PT*. Print the annotated parse tree. Print the values of the input polynomial for all integer values of x in the range [–5, 5]. Finally, call *printderivative*(*PT*) to print the derivative of the input polynomial.

  (g) Write a sample file *sample.txt* storing a single polynomial in the given format.

  (h) Write a makefile with all, run, and clean targets.

- Submit an archive (zip or tar or tgz) containing only the files *poly.l*, *poly.y*, *polyutils.c* (or *cpp*), *sample.txt*, and *makefile*. No credit if one or more of these components is/are missing.

Sample output

Call the input polynomial f(x). For f(x) = –506x^7 + 9x^10 + x – 243015 – x^3 + 876 x^2 + 1, the output is given below.

```
+++ The annotated parse tree is
    S []
    ==> - []
    ==> P [inh = -]
        ==> T [inh = -]
            ==> N [val = 506]
                ==> 5 [val = 5]
                ==> M [inh = 5, val = 506]
                    ==> 0 [val = 0]
                    ==> M [inh = 50, val = 506]
                        ==> 6 [val = 6]
            ==> X []
                ==> x []
                ==> ^ []
                ==> N [val = 7]
                    ==> 7 [val = 7]
        ==> + []
        ==> P [inh = +]
            ==> T [inh = +]
                ==> N [val = 9]
                    ==> 9 [val = 9]
                ==> X []
                    ==> x []
                    ==> ^ []
                    ==> N [val = 10]
                        ==> 1 [val = 1]
                        ==> M [inh = 1, val = 10]
                            ==> 0 [val = 0]
            ==> + []
            ==> P [inh = +]
                ==> T [inh = +]
                    ==> X []
                        ==> x []
                ==> - []
                ==> P [inh = -]
                    ==> T [inh = -]
                        ==> N [val = 243015]
                            ==> 2 [val = 2]
                            ==> M [inh = 2, val = 243015]
                                ==> 4 [val = 4]
                                ==> M [inh = 24, val = 243015]
                                    ==> 3 [val = 3]
                                    ==> M [inh = 243, val = 243015]
                                        ==> 0 [val = 0]
                                        ==> M [inh = 2430, val = 243015]
                                            ==> 1 [val = 1]
                                            ==> M [inh = 24301, val = 243015]
                                                ==> 5 [val = 5]
                        ==> - []
                        ==> P [inh = -]
                            ==> T [inh = -]
                                ==> X []
                                    ==> x []
                                    ==> ^ []
                                    ==> N [val = 3]
                                        ==> 3 [val = 3]
                            ==> + []
                            ==> P [inh = +]
                                ==> T [inh = +]
                                    ==> N [val = 876]
                                        ==> 8 [val = 8]
                                        ==> M [inh = 8, val = 876]
                                            ==> 7 [val = 7]
                                            ==> M [inh = 87, val = 876]
                                                ==> 6 [val = 6]
                                    ==> X []
                                        ==> x []
                                        ==> ^ []
                                        ==> N [val = 2]
                                            ==> 2 [val = 2]
                                ==> + []
                                ==> P [inh = +]
                                    ==> T [inh = +]
                                        ==> 1 [val = 1]

+++ f(-5)  =   127200881
+++ f(-4)  =    17498550
+++ f(-3)  =     1402957
+++ f(-2)  =     -165520
+++ f(-1)  =     -241623
+++ f( 0)  =     -243014
+++ f( 1)  =     -242635
+++ f( 2)  =     -295068
+++ f( 3)  =     -810335
+++ f( 4)  =      917822
+++ f( 5)  =    48138141

+++ f'(x)  =  - 3542x^6 + 90x^9 + 1 - 3x^2 + 1752x
```