The Grammars of the Atlas Engine

Jonathan D. Spengeman

April 23, 2016

Abstract

Atlas is an exploratory project meant to strengthen knowledge in the fields related to parsing and implementation of algorithms related to mathematics. Atlas will be utilized as a computation engine that will calculate a variety of mathematical equations. These equations are going to be limited to simple infix equations as well as concepts from the first semester of Calculus, initially. Atlas accepts a specific Context Free Grammar as it's input and parses that input using a recursive decent pattern. This paper discusses all of the grammars that are used in the Atlas engine and details the implementations. As stated, this project is going to service as exploration to enhance knowledge in specific fields but may not be making any ground breaking finds.

1 Introduction

A substantial part of this project includes the grammars that are utilized by the interpreter so it is logical to spend some time discussing those grammars. For the Atlas Engine to function properly it requires an input grammar and and as the engine grows in power it will require an output grammar but for now the primary focus will be the input grammar.

A Context Free Grammar is a set that contains 3 other sets and a singleton. The first to be described is a set of all terminal symbols which are the characters that appear in the string generated by the grammar. Next, we will discuss the set of all non-terminal symbols which are the placeholders for the patterns of the elements from the set of terminals. The last set we have is a list of productions which define the rules for replacing non-terminals with other non-terminals or terminals. The last element of a grammar is simply the starting symbol which defines which non-terminal the pattern will begin with.

$$G = \left\{ \, V, \, \Sigma, \, R, \, S \, \right\} \tag{1}$$

1.1 Intermediate Grammar

We will begin by defining an intermediate grammar that will be using to evaluate definite integrals. Imagine we want to integrate the following polynomial:

$$\int_1^5 x^3 + x^2 + x \, dx = x^4 + x^3 + x^2 \Big|_1^5 \tag{2}$$

Now imagine that our interpreter can evaluate this integral by using the input string it was given and it returns an a string that represents our evaluated integral but we have not solved for our limits of integration yet. We know we need to evaluate this integral between 0 and 5 but we currently have a string of text that represents our output so how could we finish the evaluation of this integral. Lets begin by writing out the function with the limits of integration plugged in the equation.

$$5^4 + 5^3 + 5^2 - 1^4 + 1^3 + 1^2 \tag{3}$$

If we were evaluating an indefinite integral then we would be done but in this case we still need to evaluate the integral between its limits of integration. So how would we actually take our string representation of our integral and turn it into an actual answer. We have the following string "$x^4 + x^3 + x^2$" as our initial output. Now what if we replaced all instances of x with 5 and that would give us "$5^4 + 5^3 + 5^2$" as you can see above. Then if repeat the process we get "$1^4 + 1^3 + 1^2$". If we concatenated an instance of " - " onto the first string then concatenated the second string onto the new string we would get a string that resembles "$5^4 + 5^3 + 5^2 - 1^4 + 1^3 + 1^2$" which initially may not seem like it is very helpful. Although, we can easily define an intermediate grammar that can be used in another part of our interpreter which is responsible for evaluating simple infix expressions like the new string we have. The interpreter will implement this intermediate grammar and it will take a string as input and return a number, this will allow the interpreter to evaluate most simple definite integrals.

Now let's begin with the formal definition of our intermediate grammar's productions:

$$Expression \rightarrow Expression + Term \mid Expression - Term \mid Term \tag{4}$$

$$Term \rightarrow Term \times Factor \mid Term \div Factor \mid Factor \tag{5}$$

$$Factor \rightarrow Number \mid Number \wedge Factor \tag{6}$$

$$Number \rightarrow Digits \mid (Expression) \tag{7}$$

$$Digits \to 0 \mid 1 \mid 2 \mid ... \mid 9 \mid Digits \tag{8}$$

The start symbol would be Expression and the terminals would be the set $\{\ 0,\ 1,\ 2...\ 9\}$ while the nonterminals are the set that consist of:

$$\{\ Expression,\ Term,\ Factor,\ Number,\ Digits\ \}. \tag{9}$$

With this grammar and an interpreter that implements that grammar we would be able to input any definite integral that has been expanded at both of its limits of integration as we displayed earlier. This grammar and part of the interpreter is utilitarian and does not define the grammar that will be used as the input to this interpreter but it does not mean the interpreter could not evaluate an infix expression.

1.2 Input Grammar

Now that we have an intermediate grammar that we can use to evaluate our definite integrals, we can define our input grammar. This grammar is throughly more complex then the intermediate grammar discussed above. It is a continual work in progress currently and I am leaving parts of the grammar open for expansion so we can add more functionality to our interpreter later. It is worth mentioning that some non-terminals are used from the above grammar as well and they remain the same as the above grammar unless specified other wise. The starting symbol for our grammar will be the non-terminal named Input the following elements make up the grammars productions:

$$Input \to Integral \mid Derivative \mid Expression \tag{10}$$

$$Integral \to int\ Equation\ dx\ [\ between\ Digits\ and\ Digits\ ] \tag{11}$$

$$Equation \to Polynomial \mid ... \mid ? \tag{12}$$

$$Polynomial \to Polynomial + Term | Polynomial - Term | Term \tag{13}$$

$$Term \to Term \times Factor \mid Term \div Factor \mid Factor \tag{14}$$

$$Factor \to Nomial | Nomial \wedge (Digits | (Expression)) \tag{15}$$

$$Monomial \to [Digits]\ A \mid B |\ C \mid ...\ z \tag{16}$$

$$Digits \to 0 \mid 1 \mid 2 \mid ... \mid 9 \mid Digits \tag{17}$$

The set that makes up our non-terminals would be:

$$\big\{\, Input,\ Integral,\ Equation,\ Polynomial,\ Term,\\ Factor,\ Monomial,\ Digits,\ Expression,\ \big\}$$

And the following containing all alphanumeric characters makes up our set of terminals:

$$\big\{\, 0,\ 1,\ 2 \ldots 9,\ A,\ B,\ C \ldots z \,\big\} \tag{18}$$

Lastly, are start state would simply be an instance of Input which can be a derivative, Integral or an Expression.

2 The Interpreter

Based on the specified input grammar we should define a set of functions that recursively iterate through the string that was given as input. This is currently not implemented correctly but it is referred to as recursive decent pattern and it is used to turn any context free grammar into programming logic. The difference between the input grammar and the output grammar is that the input grammar just interprets the string and checks for validity while as the interpreter for the intermediate grammar interprets and also evaluates the infix expression.