CAT Language Specification

Michael Faulkner

April 29, 2024

Introduction

CAT (Calculus and Algebra Tutor) is a domain-specific language designed to assist individuals with mathematical calculations. More specifically, the language allows users to input mathematical expressions and operations to perform on the expressions. Running a program in this language will then produce a textual represenation of the ensuing calculation as well as an optional LaTeX file and PDF that displays the calculation in a pretty manner.

While there exist computational tools for generalized mathematical expressions (for example Mathematica), such tools tend to focus only on getting to an answer. The purpose of CAT is to assist its users in understanding *how* to get the answer by outputting the intermediate steps of the calculation. This addresses another point of annoyance with tools like Mathematica that will sometimes output, seemingly arbitrary, restrictions on the parameters alongside its final expression. By working through the intermediate steps of the calculation, CAT should allow its users to understand why such restrictions were imposed and what the limits of the calculation would be if such assumptions were not made.

Design Principles

The primary goal of CAT is to produce easily understandable calculations. This goal corresponds with a prioritization of simpler techniques that are easier to follow over more powerful techniques that might be able to find more general answers but are less decipherable. This likely also means a tradeoff in terms of speed as it is necessary to work through the intermediate details of a calculation sufficiently to make them human readable.

Examples

1. The following example illustrates how each line of a CAT program is treated separately so long as there are no assignments. The first line would simplify to 1, and the second line would simplify to $2x$.
   ```
   (x+1)(x-1) - x^2
   Differentiate x^2 + y^2 wrt x
   ```
2. The following example illustrates a function definition and how it might be used on later lines. Note that the particular variables used in the function definition have no affect on what variables it is passed later. This example should expand the term inside the integral to $x + y + x + x = 3x + y$. Then, after performing the integration should find $\frac{3}{2}x^2 + xy\Big|_0^1 = \frac{3}{2} + y$
   ```
   z(x,y) = x + y
   Integrate from 0 to 1 z(x,y) + z(x,x) wrt x
   ```
3. This last example highlights some synonyms for particular language constructs and additionally showcases how undefined variables will be treated as any symbolic variable in a mathematical expression would be. Evaluating the first integral requires splitting the integral across the discontinuity in the function. And evaluating the derivative in the second line requires symbolically representing the derivative of y with respect to x.
   ```
   Int (x^2-x-6)/(x-3) from a to b wrt x
   Differentiate (x^2+2xy-y^2)/(z-3) with respect to x, where y(x)
   ```

Language Concepts

At it's core, CAT is a way of expressing and manipulating mathematical expressions. Each line of a CAT program constitutes an expression that will be evaluated. Evaluating an expression involves reducing it to the simplist form possible by applying operators and performing algebraic manipulations. The primitives of CAT are variables (i.e. x, y, z) and numbers (i.e. $1, -10, 3.141592$). These primitives can be combined with mathematical operations such as +, -, *, /, ^, sin, cos, log, sqrt, Differentiate, and Integrate in order to form expressions. Additionally, users are able to define variables and functions (i.e. $a = \text{sqrt}(2)$, or $f(x) = x + 1$) that can be used in later lines.

Formal Syntax

```
<instruction>  ::=  <expr>
                 |  <expr> '\n' <instruction>
<expr>         ::=  <ws>*<parens><ws>*
                 |  <ws>*<literal><ws>*
<parens>       ::=  (<expr>)
<literal>      ::=  <number>
                 |  <variable>
<number>       ::=  <digits>
                 |  -<digits>
                 |  <digits>.<digits>
                 |  -<digits>.<digits>
<digits>       ::=  <d><digits>
                 |  <d>
<d>            ::=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<variable>     ::=  a | b | c |...| x | y | z
<ws>           ::=  Any non-newline whitespace character.
```

Semantics

(i) The primitives in CAT are numbers and variables (i.e. 1.234 or x)

(ii) CAT supports many combining forms. These combining forms correspond with mathematical operations that take two or more operations. These include: +, -, *, /, ^, Differentiate, and Integrate. Additionally, each line of a CAT program corresponds to a separate expression that will be evaluated in sequence. Thus, a newline can be understood as a sequence operator. Assignments are also combining forms that bind a variable to some other expression. These are the only kinds of statements that interact with future lines (i.e. the line $z = x + y$ would affect all future uses of the variable z in subsequent lines).

(iii) (A) CAT programs are executed by passing them to the interpreter within a text file. For example, if I wanted to run the program `math.cat`, I would run `dotnet run math.cat`.

(B) Evaluating a CAT program will result in two outputs representing the same calculation. For each line in a CAT program, the language will attempt to simplify the mathematical expression as much as it can. This simplification occurs in a sequence of steps. Each step will be printed as a textual output. Additionally, a LaTeX file will be produced (and hopefully automatically compiled into a pdf) representing the same calculation but in an easier to read form.

For example, the program:

```
(x+1)(x-1) - x^2
Differentiate x^2 + y^2 wrt x
```

would produce the output:

```
Simplifying:  (x+1)(x-1) - x^2
==> x^2 + x - x - 1 - x^2
==> 1
```

```
Simplifying:  Differentiate x^2 + y^2 wrt x
==> 2x + 0
==> 2x
```