CSCI 3155

Interpreting Lettuce Programs into LaTex

Michael Donovan

For my final project we decided to add functionality to the Lettuce language to allow for outputting the program's output as LaTex text. The original goal was to use the JLaTexMath to allow for rendering the text to a window, however we were unable to get the library to work in Scala.

The project was built on top of the Lettuce architecture from Project2. The code to allow for Canvas and Figure functionality was removed and was in essence replaced by our Formula functionality. This functionality was enabled by adding the concept of a "Symbol" into the AST, one type for our letter variables, and one type for numbers we want converted as well. We also incorporated the Cosine, Sine, and Exponent functions, though they are not calculationally functional. We are able to add, subtract, multiply, and divide our Formulas to enable a combined output string. There is also the option to find the derivative of your Formula.

In order to actually enable Lettuce to understand these inputs, we added some functionality to the existing Lettuce Parser to allow it to recognize our keywords.

Usage:

The underlying functionality of the Lettuce language was untouched; let, function, letrec should still function as they did before.

To declare a symbolic variable in your lettuce program do: let x = symbol(x)

To declare a symbolic number in your lettuce program: let x = num(2)

Once a symbolic variable is declared you may incorporate it into our symbolic functions:

- let y = sin(x)
- let y = cos(x)
- let z = exp(x)

Once you have declared your variables you are able to use the elementary operators +,-,*,/ to combine the Formula objects into a larger Formula object.

Provided the output of the lettuce program is of the correct Formula value, then the correct LaTex string corresponding to your program will be printed to the terminal.

Examples:

```
def program1():String =
  """
    |let x = symbol(x) in
    |exp(x)
    |""".stripMargin
```

```
Returned value : FormulaValue(MyFormula(List(MyExp(x))))
MyFormula(List(MyExp(x)))
\exp{x}
```

$$exp(x)$$

```
def program2():String =
  """
    |let x = symbol(x) in
    | let y = exp(x) in
    |   let z = sin(x) in
    |     y-z
    |""".stripMargin
```

```
Returned value : FormulaValue(MyFormula(List(FormMinus(MyFormula(List(MyExp(x))),MyFormula(List(MySin(x)))))))
MyFormula(List(FormMinus(MyFormula(List(MyExp(x))),MyFormula(List(MySin(x))))))
\exp{x} - \sin{x}
```

$$\exp(x) - \sin x$$

```
def program4():String = {
  """
    |let addThrice = function (f)
    |   (f+f+f) in
    |   let x = symbol(x) in
    |   addThrice(sin(x))
    |""".stripMargin
}
```

```
MyFormula(List(FormPlus(MyFormula(List(MySin(x))),MyFormula(List(FormPlus(MyFormula(List(MySin(x))),MyFormula(List(MySin(x))))))))
\sin{x} + \sin{x} + \sin{x}
```

$$\sin x + \sin x + \sin x$$

```
def program5():String = {
  """
    |let x = symbol(x) in
    | let y = exp(x) in
    |   let z = sin(x) in
    |     y/z
    |""".stripMargin
}
```

```
MyFormula(List(FormDiv(MyFormula(List(MyExp(x))),MyFormula(List(MySin(x))))))
\frac{\exp{x}}{\sin{x}}
```

$$\frac{\exp\{x\}}{\sin x}$$

```
def program6():String =  {
  """
    |let x = symbol(x) in
    | let y = sin(x) in
    |   derivative(y)
    |""".stripMargin
}
```

```
Returned value : FormulaValue(MyFormula(List(MyDeriv(MySin(x)))))
MyFormula(List(MyDeriv(MySin(x))))
\cos{x}
```

$$\cos x$$

```
def program7():String = {
  """
    |let x = symbol(x) in
    | let y = sin(x) in
    |   let z = sin(x) in
    |     let a = y-z in
    |       derivative(a)
    |""".stripMargin
}
```

```
MyFormula(List(MyDeriv(FormMinus(MyFormula(List(MySin(x))),MyFormula(List(MySin(x)))))))
\cos{x} - \cos{x}
```

$$\cos x - \cos x$$

```
def program8():String = {
  """
    |let x = symbol(x) in
    | let y = sin(x) in
    |   let z = exp(x) in
    |     let a = y/z in
    |       derivative(a)
    |""".stripMargin
}
```

```
MyFormula(List(MyDeriv(FormDiv(MyFormula(List(MySin(x))),MyFormula(List(MyExp(x)))))))
\frac{\exp{x} * \cos{x} - \sin{x} * \exp{x}}{\exp{x} * \exp{x}}
```

$$\frac{\exp\{x\} * \cos x - \sin x * \exp\{x\}}{\exp\{x\} * \exp\{x\}}$$

```
def program10():String = {
  """
    |let x = symbol(x) in
    | let y = num(2) in
    |   let z = symbol(y) in
    |   x + y + z
    |""".stripMargin
}
```

```
MyFormula(List(FormPlus(MyFormula(List(x)),MyFormula(List(FormPlus(MyFormula(List(MyNum(2))),MyFormula(List(y)))))))
x + 2 + y
```

$$x + 2 + y$$

Future Work

There is still a lot of room left to improve upon the Formula Functionality. If we were to continue our work on it there are a few things we would add.

- Displaying Integrals
- Enabling our symbolic functions to hold more than just a symbolic variable (enable coefficients)
- Work on simplifying our Formulas
-