Taking the Derivative

Axel Månson Lokrantz

Spring Term 2023

Introduction

The task for this assignment was to work with symbolic expressions and compute the derivative of a given function. The program was written in elixir, a functional, concurrent, general-purpose programming language together with Axel Lystem at KTH.

The task

Derivative of $ln(x)$

The code for the derivative of expressions containing `addition`, `multiplication` and `exponents` was already implemented. The task was to create the remaining functions for the derivative of the following mathematical operations, $ln(x)$, $1/x$, $\sqrt{x}$ and $sin(x)$. However, output for an expression such as $2x + 4$ would yield the following result:

```
{:add, {:add, {:mul, {:num, 0}, {:var, :x}}, {:mul, {:num, 2},
{:num, 1}}}, {:num, 0}}
```

To get a more reasonable syntax we simplified the output through a `prettier_print` function. Which would print 2 as an answer. With this as our foundation, writing functions to calculate the actual value of an expression with a given x was easily done using basic arithmetic functions for all our mathematical operations.

To calculate the derivative of $ln(x)$ we created two functions.

```
def derive({:ln, {:num, _}}, _) do {:num, 0} end
def derive({:ln, e}, v) do {:mul, {:exp, e, {:num, -1}},
derive(e, v)} end
```

The first function handles the derivative of $ln(n)$ where `n` is any given number which always results in 0. The second function handles the derivative of $ln(e)$ where `e` equals an expression with the variable v we want to derivative with respect to. Since the derivative of $ln(x)$ is equal to $1/x$ which is the same as x^{-1} we chose rewrite the expression as an exponent. The multiplication operation takes the derivative of the inner and outer function to fulfill the chain-rule. The expression $ln(2x) + 4$ with a given value of $x = 5$ gave the following output:

```
Expression: (ln(2 * x) + 4)
Derivate: ((2 * x) ^ (-1) * (0 * x + 2 * 1) + 0)
Simplified: (2 * x) ^ (-1) * 2
Calculated: 0.2
```

The derivative of $1/x$ was similarly implemented using the exponent functions rather than implementing separate functions for division.

Derivative of $\sqrt{x}$

```
def derive({:sqrt, {:num, _}}, _) do {:num, 0} end
def derive({:sqrt, e}, v) do {:mul, derive(e, v), {:mul,
{:exp, e, {:num, -0.5}}, {:num, 0.5}}} end
```

The first function handles the derivative of $\sqrt{n}$ where `n` is any given number which always results in 0. The second function handles the derivative of $\sqrt{e}$ where `e` equals an expression with the variable v we want to derivative with respect to. Since the derivative of $\sqrt{x}$ is equal to $1/2 * x^{1/2}$ we simply adjusted the exponent of our expression and added a multiplication between the inner and outer derivative of the the expression.

The expression $\sqrt{2x}$ with a given value of $x = 5$ gave the following output:

```
Expression: sqrt(2 * x)
Derivate: (0 * x + 2 * 1) * (2 * x) ^ (-0.5) * 0.5
Simplified: 2 * (2 * x) ^ (-0.5) * 0.5
Calculated: 0.31622776601683794
```

Derivative of $sin(x)$

```
def derive({:sin, {:num, _}}, _) do {:num, 0} end
def derive({:sin, e}, v) do {:mul, derive(e, v), {:cos, e}} end
def derive({:cos, {:num, _}}, _) do {:num, 0} end
def derive({:cos, e}, v) do {:mul, derive(e, v), {:mul, -1,
{:sin, e}}} end
```

Since the derivative of $sin(x)$ is $cos(x)$ we had to implement separate functions for $cos(x)$. Just as before, $sin(n)$ where `n` is any given number will be considered a constant and therefor result in a derivative of 0, same goes for $cos(n)$. The expression $sin(2x) + 4$ with a given value of x = 5 gave the following output:

```
Expression: (sin(2 * x) + 4)
Derivate: ((0 * x + 2 * 1) * cos(2 * x) + 0)
Simplified: 2 * cos(2 * x)
Calculated: -1.6781430581529049
```