Taking the Derivative

Nariman Haidar

January 22, 2023

1 Introduction:

This assignment is about the derivative in Elixir. In this task we are going to construct functions regarding algorithms that can produce straightforward mathematical expressions.

2 Representing functions:

Many built-in types in Elixir, such integer, can be used to describe function signatures. Elixir has functions, but that doesn't imply that are have to use them to express general "functions". But in our case we are going to use functions that can simplify a mathematical expression to the simplest form.

3 Expressions:

First we are going to represent all operations using tuples where the first element is an atom that identifies the operation. We are going to handle four of situations differentiates a variable, number, expression of addition and an expression of multiplication.

```
@type literal() ::  {:num, number()}  | {:var, atom()}
  @type expr() :: {:exp, literal(), {:num, number()}}
  | {:add, expr(), expr()}
  | {:mul, expr(), expr()}
  | literal()
```

4 The derivative of:

In this section we are going to take the derivative of a few expressions. How will we calculate the values? First, we'll assign a value to the variable, and then we'll just use that value to calculate the output of the supplied expression.

Example of a few expressions like ln(x). In this expression we are going to represent a function ln(x) and to write the derive function so we are going to use pattern matching with the atom :ln and then we are going to define the derivative function and then use pretty print function to print the output of the derivative function in a pretty expression 1/x. To print out the output we are going to use function to simplify the expression and in this function there are a few situations like when the an expression is addition, multiplication, a number or a variable. So when it is addition so we called , function simplify-add, that is going to calculate the addition and then return the result, and when it is multiplication so we called a function simplify mul, that is going to calculate the multiplication and then return the result.

In the expression $\sqrt{x}$ we are going to represent a function $\sqrt{x}$ and to write the derive function so we are going to use pattern matching with the atom :sqrt and then use pretty print function to print the output of the derivative function in a pretty expression $1/2\sqrt{x}$. To print out the output we are going to use function to simplify the expression simplify$_sqrt$.

In the expression sin (x) we are going to represent a function sin (x) and to write the derive function so we are going to use pattern matching with the atom :sin and then we are going to define the derivative function and then use pretty print function to print the output of the derivative function in a pretty expression cos (x). The function simplify cos calls :math.cos(n) to evaluate the final value if there is one value of n.

In the expression 1/x we are going to represent a function and to write the derive function so we are going to use pattern matching with the atom :div and because of the derivative of 1/x is -1/x^2 so we are going to multiply the expression with the value of the reversed exponent then we expanded the exponent with one to get -1/x^2 , and then use pretty print function to print the output of the derivative function in a pretty expression -1/x^2. Then use the simplify function to write this value.