Lab 3 - Expression

Weichen Chai

Spring 2023

Introduction

This paper seeks to answer the questions presented in"Evaluating an expression" for ID1019. In this assignment, we will explore how the functional language Elixir can be utilized to evaluate mathematical expressions containing variables

Tasks

This assignment will be quite short as it deals with things that we have performed in previous tasks.

Expressions

This task is relatively simple with concepts taken from the previous two assignments, which are Derivatives and environment respectively. The goal is to have add, subtract, multiplication and division operators used along side rational numbers this time. Of course, with the help of maps for checking in key-value database.

We of course start with the variable and number, but more importantly, rational numbers.

```
@type literal() :: {:num, number()} | {:var, atom()}
| {:q, number(), number()}
```

We can then begin with filling in the skeleton code given to us from the assignment: to make eval functional, use the following definition, the difference with previous tasks is the inclusion of the conditions for numbers and variable, used to evaluate elements from numerical operations, of which the variable condition uses the in-built map function from elixir.

```
def eval({:num, n}, _) do n end
def eval({:var, v}, env) do
 Map.get(env, v) end //use map.get to access keys and find the value.in env
```

```
def eval({:add, e1, e2}, env) do
 add(eval(e1, env), eval(e2, env)) end
...
//The others will be similar to add function.
```

Similar to derivatives, the add, subtract, multiply and divide functions are written in a manner that considers all possibilities of that operation in including rational ones.

```
def add(a, b) do {:q, a + b, 1} end
def add({:q, n, d}, {:q, a, b}) do {:q, n*b + d*a, d*b} end
def add({:q, n, d}, c) do {:q, c*d + n, d} end
def add(c, {:q, n, d}) do {:q, c*d + n, d} end
```

In order to find the lowest common denominator, we can use the gcd (greatest common divisor) function, which is in built to the elixir system.

With the help of simplification function, we can make the outputs more readable to the human eye, we can run tests on it and see if it works as intended, for example, using the map of

```
env = %{x: 1, y: 2, z: 3}
```

where the equation is

```
{:add, {:mul, {:num, 2}, {:var, :y}}, {:sub, {:q, 3, 6}, {:q, 2, 12}}}
```

we get the result: 13/3 which proves that it works as intended(the answer was checked via a calculator)