Homework 11

November 21, 2018

1 Derivation of arithmetic expressions

For this homework, you will use Scala to implement the functions of homework 8 to take derivatives of arithmetic expressions, print the results, and simplify them.

You will define Scala classes to represent the following ML data types for arithmetic expressions.

```
datatype exp = Const of int
             | Var of string
             | Plus of exp * exp
             | Times of exp * exp
             | Pow of exp * int;
```

You should define an abstract class called `Exp` and some case classes such as `Const` and `Plus` to represent each of the data constructors.

For example, the variable e as defined below

```
val e = Times (Times (Var("x"), Var("y")), Plus (Var("x"), Const(3)))
```

represents the expression $(x \times y) \times (x + 3)$. The variable $e1$ as defined below

```
val e1 = Pow (Var("x"), 4)
```

represents the expression x^4.

The following are some rules for derivations.

$$\begin{aligned}
\frac{dc}{dx} &= 0 && \text{where } c \text{ is a constant} \\
\frac{dx}{dx} &= 1 \\
\frac{dy}{dx} &= 0 && \text{where } y \neq x \\
\frac{d(u+v)}{dx} &= \frac{du}{dx} + \frac{dv}{dx} \\
\frac{d(u \times v)}{dx} &= (\frac{du}{dx}) \times v + u \times (\frac{dv}{dx}) \\
\frac{d(u^n)}{dx} &= n \times u^{n-1} \times (\frac{du}{dx})
\end{aligned}$$

For the following methods, you should use `def` instead of `val`.

1. Implement a method `toString: string` in `Exp` class to convert **this** expression to its string representation.

   For example, `e.toString` should return the string

   ```
   "((x * y) * (x + 3))"
   ```

   and `print(e1)` should return the string

   ```
   "(x^4)"
   ```

2. Implement a method `deriv: (x: String)Exp` in `Exp` class to derive **this** expression u against the input string x and return the derivative $\frac{du}{dx}$. Note that the second parameter of the method `deriv` is the string of some variable name.

   For example, `e.deriv("x").toString` should return

   ```
   "((((1 * y) + (x * 0)) * (x + 3)) + ((x * y) * (1 + 0)))"
   ```

   while `e1.deriv("x").toString` should return

   ```
   "((4 * (x^3)) * 1)"
   ```

3. Implement a method `simplify: Exp` in `Exp` class to simplify **this** expression as much as possible.

   For example, `e.deriv("x").simplify.toString` should return

`"((y * (x + 3)) + (x * y))"`

while `e1.deriv("x").simplify.toString` should return

`"(4 * (x^3))"`

Also, if `val e2 = Pow (Plus (Var("x"), Const(0)), 2)`, then `e2.toString` should return `"((x + 0)^2)"` while `e2.simplify.toString` should return `"(x^2)"`.

Hint: for this question, you may want to define a helper method `simp` to simplify obvious expressions. $simp(e \times 0) = 0$, $simp(e \times 1) = e$, $simp(e + 0) = e$, etc. The method `simplify` should call `simp` after recursively calls itself on components of plus, times, and pow expressions.

2 Driver

Please submit all your code inside one file `Hwk11.scala` that contains the following object: **Do not submit multiple files. Do not put each class in its own file.**

```
// all your classes here

object Hwk11 {
def main(args: Array[String])  {
  val e = Times (Times (Var("x"), Var("y")), Plus (Var("x"), Const(3)))
  val e1 = Pow (Var("x"), 4)
  println(e)
  println(e1)
  println(e.deriv("x"))
  println(e1.deriv("x"))
  println(e.deriv("x").simplify)
  println(e1.deriv("x").simplify)
  val e2 = Pow (Plus (Var("x"), Const(0)), 2)
  println(e2)
  println(e2.simplify)
}
}
```

When you run your code, you should expect the following output:

```
((x * y) * (x + 3))
(x^4)
((((1 * y) + (x * 0)) * (x + 3)) + ((x * y) * (1 + 0)))
((4 * (x^3)) * 1)
((y * (x + 3)) + (x * y))
(4 * (x^3))
```

```
((x + 0)^2)
(x^2)
```