Homework 8

November 1, 2018

1 Derivation of arithmetic expressions

For this homework, you will implement functions to evaluate arithmetic expressions and also take derivatives, print the results, and simplify them.

You will use the following algebraic data types defined for arithmetic expressions.

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

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.

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

1. Implement a function `eval : exp -> (string * int) list -> int` to evaluate an arithmetic expression with a context for the variables in the expression. A context is a list of string and integer tuples.

   For example `eval e [("x", 2), ("y", 3)]` evaluates to 30 because $(x \times y) \times (x + 3)$ is $(2 \times 3) \times (2 + 3) = 6 \times 5 = 30$.

   Also, `eval e1 [("x", 2)]` evaluates to 16 because x^4 is $2^4 = 16$.

   For this eval function, you also need helper function `lookup` to look up the value of a variable in a context and `pow` function to calculate the power expression. For example $pow(2, 4)$ should return 16.

   The variable look-up is allowed to fail.

2. Implement a function `print: exp -> string` to convert an arithmetic expression to its string representation.

   For example, `print e` should return the string

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

   and `print e1` should return the string

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

3. Implement a function `deriv: exp -> string -> exp` that takes an arithmetic expression u and a string x and return the derivative $\frac{du}{dx}$. **Note that the second parameter of the function `deriv` is a variable as string.**

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

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

   while `print (deriv e1 "x")` should return

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

4. Implement a function `simplify: exp -> exp` to simplify an arithmetic expression as much as possible.

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

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

   while `print (simplify (deriv e1 "x"))` should return

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

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

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