Buğra Gedik's Courses
@ Bilkent University

cs315:fall2016:hw2

Homework 2

In this homework, you will build a simple symbolic expression execution engine using operator overloading.

Take a look at the Python code in the file symbolic.py, which implements a simple symbolic expression evaluation engine. The idea is to represent expressions as mathematical objects that can be evaluated on demand, rather than having the typical immediate evaluation semantics. This is best explained with examples:

```
def main():
    x = Var("x") # create a variable named x
    y = Var("y") # create a variable named y
    z = Var("z") # create a variable named z
    e = 3 + (x + 2 * y) * z;  # create an expression from x, y, and

    print evaluate(x, x=3) # evaluate the simple expression x
    # 3
    print evaluate(e, x=3, y=-1, z=4) # evaluate expression e for t
    # 7
    print evaluate(e, **{"x":3, "y":4, "z":2}) # same as above, wit
    # 25
    print

    print e.string() # pretty print the expression
    # 3+(x+2*y)*z
    print e.desc() # print the object structure
    # BinaryExpr(__add__,LiteralExpr(3),BinaryExpr(__mul__,BinaryEx
    print

    print x.string()
    # x
    print x.desc()
    # Var(x)
    print

    print (x+3).string()
    # x+3
    print (x+3).desc()
    # BinaryExpr(__add__,VarExpr(Var(x)),LiteralExpr(3))
    print

    print (3+x).string()
    # 3+x
    print (3+x).desc()
    # BinaryExpr(__add__,LiteralExpr(3),VarExpr(Var(x)))
    print

    print (y*(x+1)).string()
    # y*(x+1)
    print (y*(x+1)).desc()
    # BinaryExpr(__mul__,VarExpr(Var(y)),BinaryExpr(__add__,VarExpr
```

```
43      print
44
45      print ((x+1)*y).string()
46      # (x+1)*y
47      print ((x+1)*y).desc()
48      # BinaryExpr(__mul__,BinaryExpr(__add__,VarExpr(Var(x)),Literal
49      print
50
51      print (x*y+3).string()
52      # x*y+3
53      print (x*y+3).desc()
54      # BinaryExpr(__add__,BinaryExpr(__mul__,VarExpr(Var(x)),VarExpr
55      print
56
57      print (x+y+3).string()
58      # x+y+3
59      print (x+y+3).desc()
60      # BinaryExpr(__add__,BinaryExpr(__add__,VarExpr(Var(x)),VarExpr
61      print
62
63      print ((x+y)+3).string()
64      # x+y+3
65      print ((x+y)+3).desc()
66      # BinaryExpr(__add__,BinaryExpr(__add__,VarExpr(Var(x)),VarExpr
67      print
68
69      print (x+(y+3)).string()
70      # x+(y+3)
71      print (x+(y+3)).desc()
72      # BinaryExpr(__add__,VarExpr(Var(x)),BinaryExpr(__add__,VarExpr
73      print
74
75      print (((x)+y)*(3+x)).string()
76      # (x+y)*(3+x)
77      print (((x)+y)*(3+x)).desc()
78      # BinaryExpr(__mul__,BinaryExpr(__add__,VarExpr(Var(x)),VarExpr
79      print
80
81      print ((4 + 3 * x) * 3 + 5).string()
82      # (4+3*x)*3+5
83      print ((4 + 3 * x) * 3 + 5).desc()
84      # BinaryExpr(__add__,BinaryExpr(__mul__,BinaryExpr(__add__,Lite
85      print
86
```

We ask you the following in the homework:

1) (50pts) Extend the code to handle `-` (subtraction) and `/` (division) operations. Test your implementation to make sure that the precedence is respected during expression construction and evaluation.

2) (50pts) Implement a `derivative` function that takes the derivative of expressions for a given variable. For instance:

```
1  print derivative(x*y, x).string()
2  # x*0+1*y
3  print derivative(2*x*x+3*x+5, x).string()
4  # 2*x*1+(2*1+0*x)*x+(3*1+0*x)+0
```

Consider `x*y`. This can be looked at as `f(x)*g(x)` where `f(x)=x` and `g(x)=y`. The derivative of `f(x)*g(x)` is `f(x)*g'(x)+f'(x)*g(x)`, which would be `x*0+1*y`.

Logistics

It is best if you install Python (2 series) on your own computer. But you could also find it pre-installed on dijkstra.ug.bcc.bilkent.edu.tr

Once you are done, put your code under a directory named lastname_givenname_hw3 and make an archive from that directory. For example, the following Unix commands could be used:

```
mkdir lastname_givenname_hw3
cd lastname_givenname
    ...
    (edit and test your files in this directory)
    ...
cd ..
tar -cvzf lastname_givenname_hw3.tar.gz lastname_givenname_hw3
```

Then upload this newly generated file to the course Moodle.

cs315/fall2016/hw2.txt · Last modified: 2016/11/21 08:16 by bgedik