Università di Pisa

Laurea Specialistica in Informatica

Advanced Programming

Final Term Paper

Start Date: 24/08/2016
Submission deadline: 13/09/2016 (send a single PDF file to attardi@di.unipi.it)

Rules:

The paper must be produced personally by the student, signed implicitly via his mail address.

You are allowed to discuss with others the general lines of the problems, provided that each student eventually formulates his own solution. Each student is expected to understand and to be able to explain his solution.

You are allowed to consult documentation from any source, provided that references are mentioned.

It is not considered acceptable:

- **to consult or setup an online forum, to request help of consultants in producing the paper**
- to develop code or pseudo-code with others
- to use code written by others
- to let others use someone's code
- to show or to examine the work of other students.

Violation of these rules will result in the cancellation of the exam and a report to the Presidente del Consiglio di Corso di Studio.

For the programming exercises you can choose a programming language among C++, C# and Java.

The paper must:

1. be in a single PDF file, formatted readably (**font size ≥ 10 pt** with suitable margins, single column), of **no more than 10 numbered pages**, including code: for each extra page one point will be subtracted from the score.
2. include the student name
3. provide the solution and the code for each exercise separately, referring to the code of other exercises if necessary. **Do not include in an exercise code only needed for a later exercise**.
4. cite references to literature or Web pages from where information was taken.

Introduction

We will develop a library for defining, manipulating, compiling and evaluating mathematical computation graphs, similar to those of Theano (http://theano.readthedocs.io/en/latest/extending/graphstructures.html). For example:

```
Var x = new Double();
Var y = new Double();
Const c = new Const(3);
Expr e = c.mul(x.add(y));   // creates expression for 3 * (x + y)

Expr[] args = {x, y};
```

```
Function f = new Function(args, e);     // creates a function
Expr v = f.apply(1.5, 2.5);
String code = v.compile();
```

This should generate code (C or Java) for evaluating the expression.

Exercise 1

Design a hierarchy of classes to represent various kinds of mathematical expressions.

Exercise 2

Write a compiler for expressions using polymorphism and show the code generated for the expression

$2x\,e^{y}$

Exercise 3

Extend the library to perform symbolic differentiation. For example:

```
Function f = new Function(args, x.mult(2).mult(Exp(y)));
Expr df = f.differentiate(x);
Expr z = df.apply(1.5, 2.5);
String code = z.compile()
```

Provide rules to differentiate a polynomial, a product of expressions, a function composition.
Show the code generated for the example.

Exercise 4

Add simplification rules to the compiler in order to obtain optimized code. For example

$0\,x \rightarrow 0$
x +/- $0 \rightarrow x$
-(-x) $\rightarrow x$
$m\,x\,/\,m \rightarrow x$

Show the optimized code generated for the expression in Exercise 3.

Exercise 5

Extend the library to handle vectors. For example

```
Vector v1 = new Vector(3);
Vector v2 = new Vector(3);
Vector v3 = new Vector(3);
Expr s = v1.add(v2).add(v3);
```

Show how the compiler can fold the computation into a single loop.

Exercise 6

Describe how a similar expression compiler could be obtained using C++ template metaprogramming.