UNIVERSITÀ DI PISA

Università degli Studi di Pisa

Laurea Specialistica in Informatica

Advanced Programming

Final Term Paper

Start Date: 6/07/2017
Submission deadline: 26/07/2017 (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

In this project, you will develop a DSL for constraint solving. The DSL allows defining variables and relations. For example:

```
x = { x1, x2, x3 }
y = { y1, y2, y3 }
z = { z1, z2, z3 }
{ (y1, z1), (y2, z2), (y3, z3) }
!{ (x1, y2), (x1, y3), (x2, y1), (x2, y3), (x3, y1), (x3, y2) }
```

The first three lines define the possible values for variables `x`, `y` and `z` respectively. The fourth line defines a relation that the values of variables must satisfy. The fifth line defines a relation that the variables should not satisfy, or equivalently it defines the complement of the relation that the variables should satisfy.

For example if `y = y1`, then `z = z1`. Viceversa, if `x = x1`, then `y != y2`.

Exercise 1

Design a set of classes to represent the DSL.

Exercise 2

Implement a recursive descent parser for DSL without using external libraries or parser generators. Split the parser into a lexical analyzer and a syntax analyzer, as presented in the slides of the course.

Exercise 3

Implement a constraint solver for the DSL using recursion that computes and returns just the first possible solution.

Exercise 4

Implement a constraint solver for the DSL in the form of an enumerator (generator), which computes one solution at a time. Hint: at each step the algorithm should choose one possible value for a variable and propagate the consequences of the choice.

Exercise 5

Extend the classes with a mechanism to provide an explanation for each solution, i.e. which constraints where fulfilled to achieve the solution.

Exercise 6

Use the DSL to express the constraints for a Sudoku of order 2, i.e. to fill four 2x2 matrix with values 1-4 so that each value occurs just once in each row and once in each column.
Present the constraints and the solution computed with the DSL for a Sudoku with a single initial value in each of the four matrices.

Exercise 7

Explain the notion of Reflection in programming languages. Discuss what kind of support is required to provide reflection facilities and list which major programming languages provide reflection. Provide an example where polymorphism can be used instead of reflection and a case where reflection is needed.