Università degli Studi di Pisa

Laurea Specialistica in Informatica

Laurea Specialistica in Informatica e Networking

Advanced Programming

Midterm Test

Start date: 7/11/2009
Submission deadline: 14/11/2009 (send a single PDF file to attardi@di.unipi.it)

Rules:

The paper must be produced personally by the student, who signs it via his mail address.

You are allowed to discuss with others the general lines of the problems, provided that eventually each student formulates his own solution. At least 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 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 test and a report to the Presidente del Consiglio di Corsi di Studio.
For the programming exercises you can choose a programming language among C++, C# and Java.
The paper should not exceed 10 pages, including the code: for each extra page the grade will be reduced by one point.

Exercise 1

Consider a propositional Horn clause logic theory consisting of a set of implications with one consequent (clauses).
For example:

```
philosopher greek -> greekPhilosopher
Socrates -> philosopher
Plato -> philosopher
Socrates -> greek
Plato -> greek
```

A predicate *P* satisfies predicate *Q*, if it satisfies all antecedents of one clause for *Q* or if *Q* = *P*.
Define classes to represent such a theory and implement an enumerator that given a predicate lists all possible predicates that satisfy it. The enumerator should not produce the actual list of solutions, nor perform preprocessing on the data, but it should compute one solution at a time at each invocation. In the example, the enumerator for `greekPhilosopher` should return successively `Socrates, Plato` and `greekPhilosopher`, in an unspecified order.
No parser is needed for the language: provide suitable constructors or methods for the classes.

Exercise 2

Define classes that extend those of the previous exercise, so that they can represent also the probability of an implication. Extend the enumerator so that it returns the probability of each predicate P that satisfies Q, defined as the product of the probabilities of all implications required for P to satisfy the predicate Q.

Exercise 3

Extend the previous classes in order to check whether there are cycles in the implications.

Exercise 4

Comment on the article: http://www.joelonsoftware.com/items/2003/10/13.html .

Exercise 5

Discuss what generators are and which languages provide constructs to support them. What is the relation between closures and generators?