AI: Principles and techniques - Programming assignment 2

The goal of this assignment is to get a thorough understanding of constraint satisfaction problems, and specifically using arc consistency on them, and the role of heuristics in this process. For this you must implement the AC-3 algorithm as discussed in the lecture and apply it on various Sudoku puzzles. Consequently, you will apply heuristics on the AC-3 algorithm and measure the complexity of the algorithm for each heuristic and Sudoku.

There is code available in Java and Python for reading a Sudoku from the given files, and some structure to help you focus on the core of this assignment: efficiently implementing the AC-3 algorithm and experimenting with heuristics.

To help you get an overview, here is the assignment broken down in parts:

0. Become familiar with the given code and understand the AC-3 algorithm. What are the Variables, where are the Domains stored? How can you access these?
1. Find a good representation of the constraints and initialize all constraints.
2. Implement the AC-3 algorithm as discussed in the lecture, build a function that can verify the output of AC-3, and output some complexity measure of AC-3.
3. Experiment with multiple heuristics for the queue (minimum remaining values, priority to constraints that have arcs to finalized fields, etc.) and how these influence the complexity of the algorithm. Do so for all the given Sudoku files.

Bonus points are offered for implementing back-tracking so solve more difficult Sudoku's (1 bonus point extra; grade becomes 0.5 higher), and if you experiment with different constraints (also 1 bonus point).

Be sure to use informative names for variables, constants, and functions, and to document your code well. You may use libraries and pre-defined data structures etc. of course, as long as you implement the AC-3 algorithm yourself (e.g., using a PriorityQueue is OK). It is advised to make this assignment in pairs.

Report

Write a short scientific report on your implementation, describing the project, your code, your experiments, their results, and your interpretation & conclusions. You can use the various parts of the assignment to help you structure the report. For a full list of elements to include in your report, check out the "writing a programming report" section at the course guide (learning task 0) and the assessment form for this assignment. We expect a formal (!) report of ca. 4 – 8 pages (not counting possible appendices).

To hand in

Both a report (pdf) and the code (zip). Do not forget to list your name(s), student number(s), course, number of the task, date, etc. in your code and report.

The deadline of this assignment will be communicated through Brightspace.

Programming questions can be sent to the teaching assistants either during the practical sessions (preferably) or by email. Content-wise questions can be sent to the lecturer.