CSP Assignment: practical information

- to implement and describe a CSP solver that can solve Sudoku problems.
- Teams of two people
- Period: 3 weeks

Sudoku Problem as CSP

- The grid as 9x9=81 variables
- Where each variable has the domain {1, 2, 3, 4, 5, 6, 7, 8, 9} and
- the Sudoku rules have to be encoded as constraints: sets of nine variables (in a row, a column, or a region) have to be pairwise disjoint.
- The initial given cells are encoded as unary constraints

How to implement a CSP solver?

Your implementation should build on the general concepts of CSP theory:

- Variables,
- Domains
- Constraints
- ConstraintNetwork
- Search and backtracking
- Constraint propagation,
- Splitting strategy

In other words: your code must be recognisably based on the main theoretical constructs about CSP's that we discussed in class.

How generic should your CSP solver be?

You may assume the following:

- **Domains consist of discrete values**.
  (But you are not allowed to assume that domains always have 9 elements)
- **Constraints always have the form of inequalities** between variables
  (but you are not allowed to assume they are only the sudoku constraints)

The **Sudoku constraints must be recognisable** in your code so that we can change them if we want (eg to add the constraints for hypersudoku's, diagonal sudoku's, or any other constraints we might think of)

Technical Information

- Representation:
  - Each Sudoku is represented on a single line with 81 numbers and dots
  - Cells are enumerated from top-left to bottom-right.
  - Example:
    ..5.8.7..7..2.4..32.....84.6.1.5.4...8...5...7.8.3.1.45.....916..5.8..7..3.1.6..
  - A single file can contain multiple sudoku's (= multiple lines)
  - The output should also be in this format (except that the lines will of course no longer contain dots)
- Implementation:
  - Make sure your program runs on the command line:
    > my-csp-solver sudoku-1000.txt sudoku-1000-solutions.txt
  - The first argument contains unsolved Sudokus; the second file is the output file

What to hand in

- What to hand in: software and a short documentation report
  - Report includes
    - A short description of the architecture of your implementation
    - The main part describes the optimization techniques you implemented and the results of your own evaluation, i.e. how your application performed with the various optimisations (or combinations of them).
    - Around 5 pages, hand in as PDF
  - Software includes
    - All programs and data-files
    - README file explaining how to run your programs

Scoring

- We will evaluate your program with two files with Sudoku examples in the specified format.
- One file will be published
- The other file will not be published but we will use it in our own evaluation.
- Your mark will be (equally) determined by two criteria:
  1. Quality of the work, i.e. how you developed and evaluated your optimization techniques
  2. Quality of the report