Introduction to Artificial Intelligence

Exercise Sheet 4

Handed out: 18.12.2020
Due: 08.01.2021

Prof. Dr. Susanne Biundo-Stephan
Conny Olz

Exercise 1 (4 + 4 = **8 points**)

In this exercise, you shall implement two programs that solve Sudoku puzzles[1] by using the techniques you know from the lecture on *Constraint Satisfaction Search*. Both programs get as input a Sudoku puzzle (passed via std-in) that is described by 9 lines, each containing 9 numbers. Each number defines the initialization of one cell of the puzzle. A zero (0) indicates a non-initialized cell. You can assume that the puzzles that are passed on to your program are solvable.

An example puzzle as well as the corresponding input could look like:

9			2				5	7
4	5							3
7		6	3	4				
2	9	3					4	1
			5	1				6
		5		9			2	
		8	7				9	
1			8	2				
	7			3	6	1		

(a) Graphical representation

```
900200057
450000003
706340000
293000041
000510006
005090020
008700090
100820000
070036100
```

(b) Corresponding text file

Figure 1: Example Sudoku puzzle.

Exercise 1.1

Implement a program that represents the Sudoku puzzle as CSP and *obtains arc consistency*. Output the result in the following way:

- Output 9 lines, each containing 9 sets {...}. Each set represents one cell of the puzzle.
- The first line of the output represents the top-most row of the puzzle, the second the row below that top-most one, and so on.
- The first set of a line represents the left-most cell of a row, the second set the cell right of this one, and so on.
- Each set contains, in ascending order, the values that might be assigned to the corresponding variable.

An output could look like the following (this is *not* the solution to the problem given above, just an example):

```
{4,7,8,9}{5,7,8,9}{6}{1,3,7,9}{1,2,3,4,5,7,8,9}{1,2,4,5,9}{4,7,8,9}{1,3,4}{1,3,4,7,8,9}
{4,7,8,9}{2}{1}{3,7,9}{3,4,7,8,9}{6}{4,7,8,9}{5}{3,4,7,8,9}
{4,7,8,9}{3}{4,5,7,9}{1,7,9}{1,4,5,7,8,9}{1,4,5,9}{2}{1,4,6}{1,4,6,7,8,9}
{4,6,7,9}{1}{4,7,9}{5}{6,9}{8}{3}{4,6}{2}
{2,6,7,8,9}{6,7,8,9}{2,7,9}{4}{1,6,9}{3}{6,7,9}{1,6}{1,5,6,7,9}
{5}{6,9}{3}{2}{1,6,9}{7}{4,6,9}{8}{1,4,6,9}
{1,2,3,6,7}{5,6,7}{8}{1,3,6,7}{1,2,3,4,5,6,7}{1,2,4,5}{4,6}{9}{3,4,6}
{2,3,6,9}{4}{2,5,9}{8}{2,3,5,6,9}{2,5,9}{1}{7}{3,6}
{1,2,3,6,7,9}{6,7,9}{2,7,9}{1,3,6,7,9}{1,2,3,4,6,7,9}{1,2,4,9}{5}{2,3,4,6}{3,4,6,8}
```

Please be aware that...

- there are no spaces in the output,
- there are no commas between the sets, and
- all lines end with a line break.

[1] https://en.wikipedia.org/wiki/Sudoku

Exercise 1.2

Implement a program that *solves* Sudoku puzzles by combining arc consistency and backtrack search. In every step, it first obtains arc consistency (you can reuse your program from the last exercise, of course). When the puzzle is not solved afterwards, a variable is set to a value (backtrack search), and so on. Select the variable with the smallest domain (*Minimum Remaining Values* strategy). Output your solution in the way the input is given (9 lines, each containing exactly 9 numbers).

Exercise 2 (no points)

In Figure 2 you can see a *cryptoarithmetic puzzle*. Different letters represent different digits. The aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct. We assume that no leading zeros are allowed.

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

Figure 2: A Cryptoarithmetic Puzzle

1. Model the cryptoarithmetic problem given in Figure 2 as a CSP. Therefore, define the set of variables V and the domains for the variables. Furthermore, draw the respective constraint graph. Clarify the meaning of each of the nodes and edges.
2. Solve the problem by hand, using backtracking search with forward checking and ordering heuristics you have seen in the lecture.

A video with a discussion of a possible solution will be uploaded after the deadline.