CIS 391 Homework 3: Sudoku & Constraint Satisfaction

October 15, 2013

Due: Thursday, October 31, 2013

Sudoku is a simple puzzle game which is very popular in Japan and Britain and here in the US; a puzzle is published in every issue of *The Daily Pennsylvanian*. You are given a partially filled-in 9x9 grid, grouped into a 3x3 grid of 3x3 blocks, where each square is to be filled with a digit from 1 to 9, subject to the requirement that each row, column, and block must contain each digit exactly once. (For more details and helpful hints to solving Sudoku puzzles useful for this assignment, see http://www.sudoku.org.uk.)

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

1. **(10 points)** Sudoku can be viewed as a binary constraint satisfaction problem.

   (a) What are the variables of this CSP?

   (b) What are their domains?

   (c) How would you translate the requirement that no two of the same digit may occur in the same row, column, or block into binary constraints?

   (d) Do you need to explicitly constrain each digit to occur at least once in the row, col and block? Why or why not?

2. **(40 points)** Using the AC-3 algorithm, write a program to solve Sudoku problems. Come to office hours if you have any trouble.

Your input will be in the format of a 9x9 grid of characters, with a digit for each given square and a * for each unknown square. Your output should be in the same format if you find a solution; otherwise, print "no solution".

Turn in: Your code, and the results of running your program on **`ac3solvable_example`** and **`dp_puzzle`** (the latter being a puzzle from *The Daily Pennsylvanian*) (this and all other Sudoku puzzles mentioned here will be linked to from the class web site).

3. **(5 points)** Observe that your program solved the first (hand-designed) puzzle well, but made little or no progress on the second (real) Sudoku puzzle. To see why, consider the following example of one of the most basic Sudoku strategies:

Look in the lower-right hand block of the Sudoku problem on page one. For each square in this block, say whether or not the 8 could be placed there, and if not, why not? What can we conclude about where the 8 goes, and why? What general strategy does this example suggest?

4. **(20 points)** Observe that even though the requirement that each row, column, and block contain each number exactly once is implicit in the formulation of the problem as a binary CSP, arc-consistency doesn't take advantage of this. Using the trick from the previous step, code an improved version of your Sudoku solver with a better algorithm which uses AC-3 as a subroutine (which may be called more than once). Your program should be able to solve **`dp_puzzle,`** **`gentle_sudoku`**, and **`moderate_sudoku.`**

Turn in: Your code, and its output on **`dp_puzzle`**.

5. **(5 points)** Sudoku players often divide Sudoku problems into those solvable 'by logic alone' and those which 'require guessing' (with gentle and moderate problems being of the former type, and tough and diabolical of the latter). Sketch (in words, at a high-level) how you would go about building a Sudoku solver which could deal with diabolical problems.

6. **(20 points)** Implement your solution to the previous problem to make a complete Sudoku solver.

Turn in: Your code, and its output on **`guessing_puzzle`**

A word of caution - this assignment will take a fair amount of thought and some good coding, particularly the last three problems. *Don't leave it until the last minute!*