Activity 1 CPO - Computational Logic 18-19

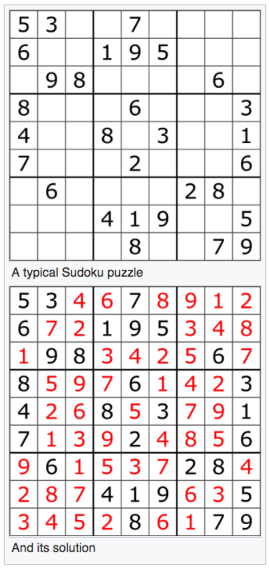
Carlos Ansótegui, Santiago Martínez, Josep Pon

October 2018

1 Introduction

The objective of this exercise is to solve Sudokus with SATisfiability technology.

As described in Wikipedia (`https://en.wikipedia.org/wiki/Sudoku`): The objective of a sudoku is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 subgrids that compose the grid (also called "boxes", "blocks", or "regions") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. In Figure 1 we can see a typical Sudoku puzzle and its solution.



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

A typical Sudoku puzzle

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

And its solution

Figure 1: A 9x9 sudoku.

In general, sudoku regions may have any rectangular shape (i.e., not necessarily square) but the resulting sudoku will still be a square shaped grid. So, a sudoku with 3x2 regions, will be a 6x6 sudoku.

There are more complicated variants, but they will not be considered in this exercise.

We will use a sudoku.sdk input text file to describe the partially filled sudoku we want to solve as follows:

```
3 3
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

Figure 2: Text file describing a sudoku.

The first line describes the height (h) and width (w) of the regions. The rest of the lines describe the content of the rows in the sudoku where each cell $c \in \{1 \ldots h \cdot w\}$ and 0 is used to describe an empty cell.

In `http://hardlog.udl.cat/static/software/Educational/sudoku2sat/sudoku2sat.zip` you can find the package that contains several utilities to help you in the development of this activity. In order to complete this activity you just need to edit the main.c file. All your code must be inserted in the main.c file.

In the utilities (sudoku.h and sudoku.c) you will find a function to read the sudoku.sdk file in your project, as follows:

```
Sudoku *sudoku = sudoku_new ();
int error_code = sudoku_parse_file (argv[1], sudoku);
```

Given a sudoku.sdk file describing a partially filled sudoku, your goal is to output a sudoku.cnf encoding a SAT formula in Dimacs format that describes the sudoku as a SAT problem. Initially, you may assume that sudokus will have square shaped regions.

In the utilities (run_solver.h and run_solver.c) you will find a function to execute the glucose SAT solver and retrieve the solutions as follows:

```
int *model = (int*) malloc (sizeof (int) * num_vars);
RunSolverCode rs_code = run_solver("./glucose -model",
                                    "instance.cnf", model);
```

These are the tasks you need to accomplish (all line numbers refer to the main.c file):

- (3 points) Given a set of Boolean variables B, program a function that encodes as a set of SAT clauses:

   1. (1 point) At Least One variable in B is true (ALO) (Line 17).
   2. (1.5 points) At Most One variable in B is true (AMO) (Line 22).
   3. (0.5 point) Exactly One variable in B is true (EO) (Line 27).

- (Provided) Program the necessary code that fulfills that each cell gets exactly one value (Line 63).
- (1.5 points) Program the necessary code that fulfills the sudoku row constraints (Line 73).
- (1.5 points) Program the necessary code that fulfills the sudoku column constraints (Line 76).
- (2 points) Program the necessary code that fulfills the sudoku region constraints (Line 79).
- (1 point) Program the necessary code that fulfills the sudoku fixed number constraints (Line 82).
- (0.5) Modify the code that sets the number of variables and clauses (Lines 53 and 54).
- (0.5 points) There are several ways to fulfill the previous constraints involving different numbers of clauses. Try to reduce as much as possible the number of clauses of your encoding.
- Optional: extend your project to work on arbitrary $h \times m$ sudokus.

In the utilities (teacher.h and teacher_linux.o) you will find the teacher version of the alo, amo and eo functions. These are provided so that you can experiment with the sudoku constraints before programming your own version of these functions. Notice that in the provided code to fulfill the cell constraints there is a call to teacher_eo. Be sure to change it for a call to eo when you have programmed it.

2 Deliverables

You will upload a log-activity1.tgz file, with the following structure: (i) a document.pdf (max. 2 pages) that describes how you addressed the required tasks, and (ii) a subdirectory src containing the main.c file (you can find it in the sudoku project) filled with the source code corresponding to the above tasks.

```
log-activity1.[tgz|tar.gz|zip]
 ├─ document.pdf
 └─ src
     └─ main.c
```