Assignment 1

September 30, 2018

1 Objective

1.1 Create a SUDOKU+ solver

A SUDOKU+ puzzle has all the constraints for a 9×9 SUDOKU with an additional constraint that all the numbers along the main diagonal occur only once. You should be able to enter a SUDOKU+ puzzle from a text file as shown below:

```
1 . . 2 . . 4 . .
. . . . . . . . .
. . . . . . . . .
. . . . . . . 7 .
. . . . . . . . .
. . . . . . . . .
. . 8 . . . . . . .
. . . . . . . . .
. . . . . 2 . . .
```

Your program should emit the solution by printing the corresponding number instead of the dots. You are allowed to use any programming language of your choice. Your program should:

- emit the SAT encodings in the DIMACS representation
- invoke **minisat** (using calls like system() or exec() in C)
- parse the result from **minisat**
- decode the result and print the solution in the required format

1.2 Create a SUDOKU+ puzzle generator

You should emit a minimal SUDOKU+ generator in the format as shown above (i.e. the respective number or a dot for an unfilled cell). You should use the encoding used in the above problem and you may require to invoke the sat solver multiple times on different queries. Please note:

1. The puzzle should have a unique solution
2. The solution must be minimal, i.e. it should not be possible to remove any of the filled entries without allowing multiple solutions
3. Use randomization: every invocation of your program should generate a different puzzle
4. Play with your encoding: some encoding may take more time than others; experiment to find out what works best

2 Deliverables

The following should be archived and uploaded either as a .tar.gz or .zip file; no other format is allowed.

1. The program source code. Do not include the source of the SAT solver or any executable in your package. You may assume that the SAT solver is placed in the same directory as your program for execution.
2. A README file (a plain text file) describing how to build and run your program.