CMPT 477/777 Formal Verification

Programming Assignment 2

1 Problem Description

(100 points) Sudoku is a puzzle that fills digits 1 – 9 to a 9×9 grid. Typically, the initial grid is partially filled with some digits. The goal of Sudoku is to complete the grid such that

- Each row must contain each of the digits 1 – 9 *exactly* once.
- Each column must contain each of the digits 1 – 9 *exactly* once.
- Each of the nine 3×3 blocks (see Figure 1) must contain each of the digits 1 – 9 *exactly* once.

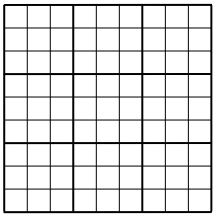

Figure 1: Blocks in empty Sudoku.

Please write a **Java** program with Z3 APIs to solve the Sudoku problem. The input is a file called `input.txt` that describes the initial grid

$$\begin{array}{cccc} d_{1,1} & d_{1,2} & \dots & d_{1,9} \\ d_{2,1} & d_{2,2} & \dots & d_{2,9} \\ \dots & & & \\ d_{9,1} & d_{9,2} & \dots & d_{9,9} \end{array}$$

where each $d_{i,j} (1 \leq i \leq 9, 1 \leq j \leq 9)$ is a digit from 0 to 9. $d_{i,j} = k$ means the cell at row i and column j is filled with k in the initial grid (0 means not filled). Digits are separated by a single space.

The output is also a file in the same format that describes a solved Sudoku. If the grid with provided digits cannot be solved, write "`No Solution`" in the output file.

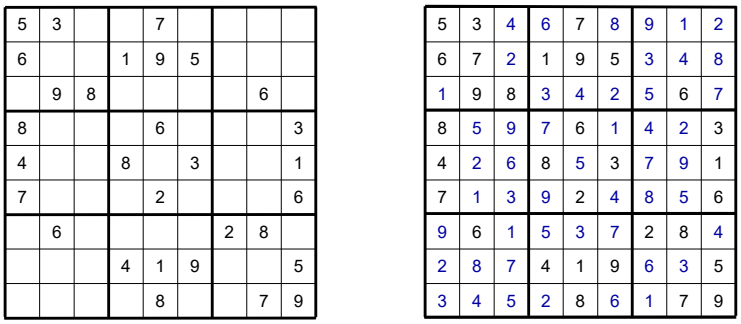

Figure 2: Example input and output (not unique).

2 Sample Input and Output

Suppose we have an input file `input.txt` that contains the following nine lines

```
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
```

which represents the Sudoku problem on the left side of Figure 2. One possible output is a file that contains the following lines

```
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
```

which represents the solved Sudoku on the right side of Figure 2.

3 Compilation and Execution

Compilation. The provided codebase uses the Maven build system. After you enter the `verif-smt` directory, the project can be easily compiled with one command

```
$ mvn package
```

Then you should be able to see the message "`BUILD SUCCESS`". A directory called `target` will be created and a jar file called `verif-smt-1.0.jar` will be generated inside the `target`.

Execution. In the `verif-smt` directory, you can execute the program using the following command (put the library path after `-cp` in quotes and use `;` instead of `:` on Windows)

```
$ java -cp lib/com.microsoft.z3.jar:target/verif-smt-1.0.jar smt.Sudoku <in-path> <out-path>
```

where `<in-path>` is the path to the input file and `<out-path>` is the path to the output file.

For example, you can run

```
$ java -cp lib/com.microsoft.z3.jar:target/verif-smt-1.0.jar smt.Sudoku input.txt output.txt
```

You will see a runtime exception with message "`To be implemented`", because the program is not implemented yet. After you finish the implementation, you should see a file named `output.txt` with the content as shown in Section 2.