ToC Spring 2022

HOMEWORK 6 REPORT

Bettelini Carlo, Likollari Kelvin, Milanesi Claudio,
Alexandra Willi

Introduction

As the name suggests, SAT solvers can be used to find a solution to a problem. Because the solver has its own language, the problem and the set of constraints needs to be encoded into the solver's language before using the solver. The idea is to encode each constraint as a set of clauses (made of variables) that will then be concatenated, creating a single CNF for which the solver will try to find a solution. Upon feeding the CNF to the solver, if the CNF is satisfiable, the solver will return the values of the variables for which the CNF evaluates to true. For the Sudoku satisfiability problem, the following conditions must hold:

1. Each cell must have at least a number between 1 and 9
2. Each number must appear only once in each row
3. Each number must appear only once in each column
4. Each number must appear only once in each 3 by 3 box
5. The initial input numbers must be present

Implementation

To encode the problem, we chose to denote each variable as $InCell(i, j, n)$, where i denotes the row, j denotes the column and n denotes the number in the cell. Since each cell can have a value from 1 to 9, and all cells need to be filled, we have up to $9 \cdot 9 \cdot 9 = 729$ different variables. Furthermore, since the SAT solver we chose, Z3, works with integers as variables in the clauses, we created a map from each variable to its unique ID.
Upon creating the mapping, we encoded the constraints as follows: the first condition can be encoded straight forward. What this means is that for each cell (i, j), a number between 1 and 9 must be present. This can be written as follows:

$$InCell(1, 1, 1) \vee InCell(1, 1, 2) \vee InCell(1, 1, 3) \vee ...$$

For the second condition, saying that each number must appear only once in each row is equivalent to saying that each number cannot stay in the same row. This can be written as follows:

$$\neg(InCell(1,1,1) \wedge InCell(1,2,1) \wedge InCell(1,3,1) \wedge ...)$$

which is equivalent to:

$$\neg InCell(1,1,1) \vee \neg InCell(1,2,1) \vee \neg InCell(1,3,1) \vee ...$$

The same reasoning can be used for the third condition. That is, instead of incrementing j, we increment i first, like so:

$$\neg InCell(1,1,1) \vee \neg InCell(2,1,1) \vee \neg InCell(2,1,1) \vee ... \quad (1)$$

For the fourth constraint, we need to check that each number in each box is unique.
OH BOY
For the last condition, we just need to write one clause for each variable we have in the input. For example, given number 3 in cell $(1,2)$ and number 6 in cell $(5,5)$, we write:

$$InCell(1,2,3) \wedge InCell(5,5,6) \quad (2)$$