The Sudoku Puzzle

Thomas Del Prete,
Alessandra Hädener,
Simone Masiero,
Yanna Poncioni,
Damiano Pugliesi

May 22, 2019

Introduction

Sudoku is a placement puzzle, also known as Number Place in the U.S.A.
The game consists most frequently of a 9 x 9 grid, divided in 9 subgrids with dimension 3 x 3 called "regions".
The purpose is to enter a digit from 1 to 9 (or other symbols e.g. letters, icons) in each cell of the grid so that each row, column and region contains only one instance of each digit.
We implemented a **SAT solver** (instead of combining backtracking and methods for constraint propagation as other Sudoku solver) to figure out a correct solution for the Sudoku.
Basically the Sudoku is translated into a propositional formula that can be satisfied only if the Sudoku has a solution.
Once the propositional formula is generated, the SAT solver tries to find a satisfying assignment that will become the solution for the original Sudoku.

Reduces Sudoku problem to a SAT clause

Digits are modelled by a datatype with nine elements $(1, .., 9)$.
We can say that the grid cells $(x1, .., x9)$ are valid if they contain at least and at

most 1 digit each.

Definition 1

$$valid\left(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\right) \equiv \bigwedge_{d=1}^{9} \bigvee_{i=1}^{9} x_i = d$$

Labeling the 81 cells, we can check if the cells generate a correct solution for the Sudoku puzzle.

Sudoku definition

$$sudoku\left(\{x_{ij}\}_{i,j\in\{1,\ldots,9\}}\right) \equiv \bigwedge_{i=1}^{9} valid\left(x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, x_{i6}, x_{i7}, x_{i8}, x_{i9}\right)$$

$$\wedge \bigwedge_{j=1}^{9} valid\left(x_{1j}, x_{2j}, x_{3j}, x_{4j}, x_{5j}, x_{6j}, x_{7j}, x_{8j}, x_{9j}\right)$$

$$\wedge \bigwedge_{i,j\in\{1,4,7\}} valid\left(x_{ij}, x_{i(j+1)}, x_{i(j+2)}, x_{(i+1)j}, x_{(i+1)(j+1)}, x_{(i+1)(j+2)},\right.$$

$$\left. x_{(i+2)j}, x_{(i+2)(j+1)}, x_{(i+2)(j+2)}\right)$$

SAT Solver

We're now introducing 9 boolean variables for each cell in the 9x9 grid ($9^3 = 729$ variables in total) in order to encode a Sudoku.
Each boolean value holds the truth value of the equation $x_{i,j} = d$.
A clause

$$\bigvee_{d=1}^{9} p_{ij}^{d}$$

assures that a cell contains one of the nine accepted digits, whilst 36 clauses

$$\bigwedge_{1\leq d<d'\leq 9} \neg p_{ij}^{d} \vee \neg p_{ij}^{d'}$$

assure that a cell doesn't hold two different digits.
Since the number of digits is equal to the number of cells in every row, column or region, then the nine grid cells (x_1, ..., x_9) hold distinct values.

Lemma 1

$$valid\,(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) \iff \bigwedge_{1 \le i < j \le 9} x_i \neq x_j$$

$$\iff \bigwedge_{1 \le i < j \le 9} \bigwedge_{d=1}^{9} x_i \neq d \vee x_j \neq d$$

The given formula, when converted into `SAT`, is translated into 9 clauses * 36 inequations = 324 clauses, each of length 2. This allows more unit propagation at the boolean level than what was previously stated in **Def. 1**, which gives us the possibility of cross hatching digits (a technique used in Sudoku to reduce the search space).
Summarising, up to now, our `SAT` is composed of:

1. 81 definedness clauses of length 9
2. $81 \cdot 36$ uniqueness clauses of length 2
3. $27 \cdot 324$ validity clauses of length 2

For a total of **11745** clauses.
Since we know a priori the value of some cells (user input), we can just add these condition to our clause list.
Finally, our encoding produces a propositional formula already in `CNF`, so the conversion into `DIMACS CNF` (the input format for most sat solvers) is trivial.