Team members: Phong Tran, Santos Solorzano, Brandon Arroyo

Class: CSCE 411 - 502

Sudoku Solver

Extra Credit

I. Introduction

After trying out other SAT solvers, we decided to use the pre-built A SAT-based Sudoku Solver from Tjark Weber. The idea is to solve Sudoku using SAT solvers and study the impact of clause generation. If the SAT solver finds a satisfying assignment, this assignment can readily be transformed into a solution for the original Sudoku.

PicoSAT is an SAT-based Sudoku solver written in pure C, and we used the pycosat package [3]. It is based on a paper Dr. Klappenecker provided, which explains in detail how the Sudoku problem is translated into a satisfiability problem. Basically, for each possible digit (1-n^2) in the n^2 x n^2 Sudoku cells, we assign a Boolean variable. The pycosat source code contains the Sudoku solver (40 lines of actual Python code) [3].

II. Generation of clauses

If we were using variables with arbitrary finite domains, then 9·9 variables with domain [1..9] would be the most adequate option. Encoding Sudoku puzzles into CNF requires $9 \cdot 9 \cdot 9 = 729$ propositional variables. For each entry in the 9×9 grid S, we associate 9 variables [1].

III. Translation to SAT

Each cell will have n^2 variables in the n^2 x n^2 grid, thus n^6 in total. Each Boolean variable p^d_{ij} (with $1 <= i, j, d <= n^2$) represents the truth value of the equation $x_{ij} = d$. A clause is represented as:

$$\bigvee_{d=1}^{n^2} p_{ij}^d$$

It ensures that the cell x_{ij} denotes one of the n^2 digits, and the clauses

$$\bigwedge_{1 \le d < d' \le n^2} \neg p_{ij}^d \vee \neg p_{ij}^{d'}$$

make sure that the cell does not denote two different digits at the same time [2].

The encoding already yields a propositional formula in conjunctive normal form (CNF). Therefore, conversion to DIMACS CNF is trivial. It can search for a satisfying assignment using an internal DPLL-based SAT solver [2].

Sources

[1] http://ldc.usb.ve/~meza/ci-5651/e-m2008/articulosYsoftware/SudokuAsSAT.pdf

[2] https://www.lri.fr/~conchon/mpri/weber.pdf

[3] https://www.continuum.io/blog/developer/sat-based-sudoku-solver-python