Lab 4: Sudoku Solver

To solve a given sudoku board using a SAT solver, we first had to formulate the game state in boolean logic. There were two approaches of doing it.

Approach 1: Pruning

Let's say we want to generate the CNF for the case of a single row. We will have to ensure that each number exists exactly once in the row.

For this we can check that a value exists atleast once (and atmost once) in the row for each value. But let's say we know that the 3rd column of the 1st row is filled with 4, this means when generating the CNF for the 1st row, we can skip (prune) the part responsible for checking that 4 exists atleast once.

Similar approaches can be taken for each constraint to generate a CNF that has been made using reduction of the base CNF depending on the initial state

Approach 2: Adding clauses after base case

Let's first create a base case representation for the sudoku solver, that is, assuming all spaces are empty. This CNF will capture all the rules of the game.

Now for each initial state, add the clauses [ijk] for each row i, column j, where `k != '.'` .

This will ensure that the answer provided will satisfy both the rules and the initial state.

Why Approach 2?

By using approach 2, we only need to create the bulk of the CNF (the base case) **once**. Then, we can simply add the required clauses for each base case later.

How to calculate Base CNF?

1. Atleast one and Atmost one value per cell

By ensuring that `[ijk for k in 1..9]` for each i and j, we ensure that atleast one value is present in each cell.

By ensuring that no two values are present in a cell at the same time, we ensure that atmost one value is present in each cell

2. Row and Column

For each row/column, we check if each value exists

- atleast once: `[ijk for j in 1..9]` for each i and k
- atmost once (similar to point 1.)

3. 3x3 Blocks

For each block, we check if each value exists atleast once and atmost once (similar to above)

Submitted By

- Kaushik Rawat, 112201015
- Hemant Pathak, 112201024