CS202 Assignment - 1

Janhvi Rochwani (200467)
Vedasree Tatimanu (201049)

January 31, 2022

Question 1

The program takes a parameter k (dimension of sudoku) and a CSV file that contains two sudokus.
We wished to output a unique pair of sudokus such that all corresponding cells are differently filled.

Firstly, the sudokus must be valid. We made use of the CardEnc class (from the Card module) to write clauses for the same. We also used the IDPool class from the formula module to create IDs for each cell and its corresponding value.

Our solver considers the restrictions that the filled cells of each sudoku must not be changed.
We wrote our code in such a way that each cell has exactly one digit in the range 1 to k^2 (by using the CardEnc.equals() method.) Similarly, each row, column and kxk block has every digit exactly once and we obtain two valid sudokus.

With the additional constraint that every cell of sudoku-1 and sudoku-2 be different, we obtain a unique pair of sudokus.

If the CNF is unsatisfiable, the Solver returns False. Otherwise, it returns True and can give a model solution by the get_model() method. This model solution is a list of IDs of literals that are true or false. By extracting the IDs of literals that are true (using vpool.obj()), we can print the two unique sudokus.

Limitations:

Runtime:
Number of clauses for row/column/cell/block constraints is of order k^6
Time for adding clauses is O(k^6)
For larger k values run time increases exponentially.

Question 2

With k as input and two empty k-dimension sudokus, we made use of Question-1 to create a unique sudoku pair (*unique_pair* function).
We call a cell "redundant" if filling that cell with 0 still leaves behind a unique pair of sudokus.

For every filled cell, we check if the cell is redundant by using the *is_redundant_cell* and the *Solutions* functions. *is_redundant_cell* just replaces the current cell with 0 and calls the *Solutions* function on

the pair. The *Solutions* function solves the pair with the *unique_pair* generator and returns 1 if the number of solutions is 1.
This will result in a unique sudoku pair with maximal holes.

Limitations:

Runtime:
Code calls unique_pair function k^4 times because number of cells = k^4
Runtime of unique_pair function = k^6
Therefore, runtime of code of a sudoku of dimension k = O(k^{10})
For larger k values run time increases exponentially.