# Theory of Computation - Practical Homework 2 Report

Andreas - Theologos Spanopoulos (spanoa@usi.ch)

May 30, 2020

## A little bit about the Implementation

In this assignment, I implemented a python2 script that encodes any $n \times n$ Sudoku grid with $r \times c$ blocks to CNF for SAT solvers (specifically cadical). Note that the normal version of Sudoku consists of a $9 \times 9$ grid with $3 \times 3$ blocks. The script works by reading the Sudoku configuration from an input file, translates the constraints into CNF, and feeds the clauses to cadical. After that, if a solution exists, it gets printed; else it prints that the problem is unsatisfiable. More on the format of the input file can be found in the next pages of the report.

## Explanation of CNF Encoding

First we start by denoting that an $n \times n$ Sudoku grid has $n^2$ cells. Each cell can have $n$ different values. For this problem, a cell is considered a variable. Translating this to propositional logic, yields the set of literals

$$\text{cell\_has\_value}([i,j], v)$$

where $[i,j]$ are the coordinates of a cell and $v$ is the symbol (value) it has been assigned. It is quite obvious that we have $n^2 \times n = n^3$ different literals (variables for the SAT solver).

In order to convert the Sudoku problem in CNF, we have to model the family of constraints. There are 5 sets of constraints that are needed for this purpose, as shown below:

1. Every cell $[i,j]$ must have at least one value.
2. Every cell $[i,j]$ cannot have more than one value.
3. Every row must contain all values only once $\Leftrightarrow$ No 2 cells in the same row can have the same value.
4. Every column must contain all values only once $\Leftrightarrow$ No 2 cells in the same column can have the same value.
5. Every block must contain all values only once $\Leftrightarrow$ No 2 cells in the same block can have the same value.

Before the constraints are explained, we have to introduce some terminology.

- Let $\mathbb{N}_n$ be the set of the first $n$ natural numbers, that is

$$\mathbb{N}_n = \{x \mid x \in \mathbb{N} \land x \leq n\} = \{1, 2, ..., n\}$$

- Let $S = \{s_1, s_2, ..., s_n\}$ be the set of the available symbols (values) for the cells. It follows that $|S| = n$.
- Let $r, c$ represent the dimensions of a block (number of rows and columns respectively).

In the normal Sudoku, we have that $N_9 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $r = c = 3$ (blocks of dimenson $3 \times 3$).

In the next page we analyze every set of constraints.

## 1. Every cell [i, j] must have at least one value

This constraint can be achieved by stating that a cell $[i, j]$ either has to have the first possible value $s_1$, or the second $s_2$, or the third $s_3$, and so on until $s_n$. In propositional logic we have:

$$\text{cell\_has\_value}([i,j], s_1) \vee \text{cell\_has\_value}([i,j], s_2) \vee \cdots \vee \text{cell\_has\_value}([i,j], s_n)$$

In order to create the set of clauses for this constraint, we have to generalize the above case to every cell in the Sudoku:

$$C_1 = \bigwedge_{(i,j) \in \mathbb{N}_n \times \mathbb{N}_n} \bigvee_{s \in S} \text{cell\_has\_value}([i,j], s)$$

In a Sudoku grid we totally have $n^2$ cells and $n$ values. Therefore, the number of clauses in this set is $|C_1| = n^2$, where each clause is a disjunction of $n$ literals.

## 2. Every cell [i, j] cannot have more than one value

This constraint can be achieved by stating that a cell $[i, j]$ cannot have at the same time the values $s_1, s_2$, the values $s_1, s_3$, ..., the values $s_1, s_9$, ..., the values $s_8, s_9$. Basically we have to create all the $\binom{n}{2}$ pairs of 2 values. In propositional logic we have:

$$\begin{aligned}
&(\neg\text{cell\_has\_value}([i,j], s_1) \vee \neg\text{cell\_has\_value}([i,j], s_2)) \wedge \\
&(\neg\text{cell\_has\_value}([i,j], s_1) \vee \neg\text{cell\_has\_value}([i,j], s_3)) \wedge \ldots \wedge \\
&(\neg\text{cell\_has\_value}([i,j], s_1) \vee \neg\text{cell\_has\_value}([i,j], s_9)) \wedge \ldots \wedge \\
&(\neg\text{cell\_has\_value}([i,j], s_8) \vee \neg\text{cell\_has\_value}([i,j], s_9))
\end{aligned}$$

In order to create the set of clauses for this constraint, we have to generalize the above case to every cell in the Sudoku:

$$C_2 = \bigwedge_{(i,j) \in \mathbb{N}_n \times \mathbb{N}_n} \bigwedge_{k=1}^{n-1} \bigwedge_{m=k+1}^{n} \neg\text{cell\_has\_value}([i,j], s_k) \vee \neg\text{cell\_has\_value}([i,j], s_m)$$

In a Sudoku grid we totally have $n^2$ cells and $\binom{n}{2} = \frac{n(n-1)}{2}$ different pairs for the values. Therefore, the number of clauses in this set is $|C_2| = n^2 \times \frac{n(n-1)}{2} = \frac{n^3(n-1)}{2}$.

## 3. No 2 cells in the same row can have the same value

This constraint can be achieved by stating that 2 cells in the same row $([i, j_1], [i, j_2])$ cannot have the same value, for every possible value. In propositional logic we have:

$$\begin{aligned}
&(\neg\text{cell\_has\_value}([i,j_1], s_1) \vee \neg\text{cell\_has\_value}([i,j_2], s_1)) \wedge \\
&(\neg\text{cell\_has\_value}([i,j_1], s_2) \vee \neg\text{cell\_has\_value}([i,j_2], s_2)) \wedge \ldots \wedge \\
&(\neg\text{cell\_has\_value}([i,j_1], s_n) \vee \neg\text{cell\_has\_value}([i,j_2], s_n))
\end{aligned}$$

In order to create the set of clauses for this constraint, we have to generalize the above case for every row in the Sudoku:

$$C_3 = \bigwedge_{i \in \mathbb{N}_n} \bigwedge_{j_1=1}^{n-1} \bigwedge_{j_2=j_1+1}^{n} \bigwedge_{s \in S} \neg\text{cell\_has\_value}([i,j_1], s) \vee \neg\text{cell\_has\_value}([i,j_2], s)$$

In a Sudoku grid we totally have $n$ rows, $\binom{n}{2} = \frac{n(n-1)}{2}$ different pairs of columns in the same row, and $n$ values that can be assigned to a cell. Therefore, the number of clauses in this set is $|C_3| = n \times \frac{n(n-1)}{2} \times n = \frac{n^3(n-1)}{2}$.

### 4. No 2 cells in the same colunn can have the same value

This constraint can be achieved by stating that 2 cells in the same column ($[i_1, j], [i_2, j]$) cannot have the same value, for every possible value. In propositional logic we have:

$$(\neg \text{cell\_has\_value}([i_1, j], s_1) \lor \neg \text{cell\_has\_value}([i_2, j], s_1)) \land$$
$$(\neg \text{cell\_has\_value}([i_1, j], s_2) \lor \neg \text{cell\_has\_value}([i_2, j], s_2)) \land \dots \land$$
$$(\neg \text{cell\_has\_value}([i_1, j], s_n) \lor \neg \text{cell\_has\_value}([i_2, j], s_n))$$

In order to create the set of clauses for this constraint, we have to generalize the above case for every column in the Sudoku:

$$C_4 = \bigwedge_{j \in \mathbb{N}_n} \bigwedge_{i_1=1}^{n-1} \bigwedge_{i_2=j_1+1}^{n} \bigwedge_{s \in S} \neg \text{cell\_has\_value}([i_1, j], s) \lor \neg \text{cell\_has\_value}([i_2, j], s)$$

In a Sudoku grid we totally have $n$ columns, $\binom{n}{2} = \frac{n(n-1)}{2}$ different pairs of rows in the same column, and $n$ values that can be assigned to a cell. Therefore, the number of clauses in this set is $|C_4| = n \times \frac{n(n-1)}{2} \times n = \frac{n^3(n-1)}{2}$.

### 5. No 2 cells in the same block can have the same value

This constraint can be achieved by stating that 2 cells in the same block ($[i_1, j_1], [i_2, j_2]$) cannot have the same value, for every possible value. In propositional logic we have:

$$(\neg \text{cell\_has\_value}([i_1, j_1], s_1) \lor \neg \text{cell\_has\_value}([i_2, j_2], s_1)) \land$$
$$(\neg \text{cell\_has\_value}([i_1, j_1], s_2) \lor \neg \text{cell\_has\_value}([i_2, j_2], s_2)) \land \dots \land$$
$$(\neg \text{cell\_has\_value}([i_1, j_1], s_n) \lor \neg \text{cell\_has\_value}([i_2, j_2], s_n))$$

In order to create the set of clauses for this constraint, we have to generalize the above case for every block in the Sudoku:

$$C_5 = \bigwedge_{i=1}^{n/r-1} \bigwedge_{j=1}^{n/c-1} \bigwedge_{i_1=ir}^{ir+r-1} \bigwedge_{j_1=jc}^{jc+c-1} \bigwedge_{i_2=i_1}^{ir+r-1} \bigwedge_{\substack{j_2=jc,\\ i_1 \neq i_2 \lor \\ j_2 > j_1}}^{jc+c-1} \bigwedge_{s \in S} \neg \text{cell\_has\_value}([i_1, j_1], s) \lor \neg \text{cell\_has\_value}([i_2, j_2], s)$$

In a Sudoku grid we totally have $n$ blocks, $\binom{n}{2} = \frac{n(n-1)}{2}$ different pairs of cells in the same block, and $n$ values that can be assigned to a cell. Therefore, the number of clauses in this set is $|C_5| = n \times \frac{n(n-1)}{2} \times n = \frac{n^3(n-1)}{2}$.

## Summary and Complexity

The union $C = C_1 \cup C_2 \cup C_3 \cup C_4 \cup C_5$ of the above sets of constraints will give us all the clauses needed to fully translate Sudoku to CNF. The cardinality of $C$ can be calculated by adding the individual cardinalities of each set:

$$|C| = |C_1| + |C_2| + |C_3| + |C_4| + |C_5| = 2n^3(n-1) + n^2$$

Note than the above equation is not absolutely true, since $C_3 \cap C_5 \neq \varnothing$ and $C_4 \cap C_5 \neq \varnothing$. This happens because some cells in the same block happen to be in the same row or column, something that has already been taken into account in the previous sets. We could take care of this matter but I believe that modern SAT are able to check for overlapping clauses, so it's ok. The time needed to produce these clauses is proportional to their size ($\mathcal{O}(n^4)$).

# Explanation of python script

The basic idea of the script is that it reads input from a file (for example "input_sudoku.txt"), creates a file with the CNF encoding and then feeds it to cadical. Then it reads the output from cadical, and if there is a solution, it gets printed; else it prints that the problem is unsatisfiable. Note that the second line of the script has to be edited manually. The correct path of the solver (cadical) has to be provided in order for the program to work.

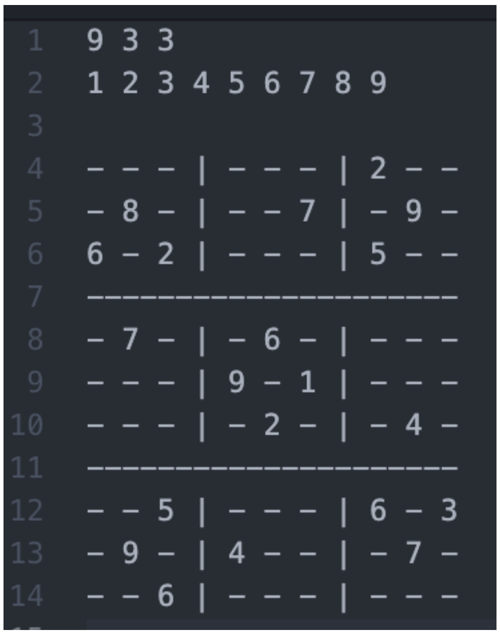
## Input file format

1. The first line of the input file must contain 3 integers separated by a space:

   Grid size $n$, rows per block $r$ and columns per block $c$

2. The second line contains all the $n$ symbols that should be used to solve the Sudoku, separated by a space each.
3. Third line is empty.
4. From the fourth line and on, only the Sudoku puzzle should exist. Each line contains $n$ cells. The cells who do not have a value, are represented with a dash "-". Those who have a value, are represented by it. Each cell is separated by a space. When a block ends (due to advancing in columns), then the character "|" should be put to indicate the boundaries of the block, and then proceed normally. When a block ends due to advancing in rows, a line of just dashes is placed again to indicate the boundaries of the block.

For the normal Sudoku where $n = 9, r = c = 3$, the input file should look like this:



```
9 3 3
1 2 3 4 5 6 7 8 9

- - - | - - - | 2 - -
- 8 - | - - 7 | - 9 -
6 - 2 | - - - | 5 - -
---------------------
- 7 - | - 6 - | - - -
- - - | 9 - 1 | - - -
- - - | - 2 - | - 4 -
---------------------
- - 5 | - - - | 6 - 3
- 9 - | 4 - - | - 7 -
- - 6 | - - - | - - -
```

## Running the script

The user just has to be in the directory that contains the python script sudoku_to_sat.py, and run it like this:

```
$ python sudoku_to_sat.py input_file
```