Using The Minisat Algorithm To Solve Sudoku Puzzles

Goals

1. Getting an input from the user and translate it to the suitable Sudoku borad.
2. Encoding the board to the optimal boolean expression in CNF form, and writing that expression to a file to pass it on to the minisat.
3. Taking the output file from the Minisat and decide if the expression that represents the Sudoku board can be satisfied.
4. If so then translate this output to the solution a solution of the board and print it to the user.
5. The most important: Encoding the Sudoku in the optimal way that the computer can't reach it's maximum capacity and then handle the number of literlas that we used to encode the board.

Demands

- Using files – input and output stream.
- Exploring the minisat algorithm and dealing with the input and output forms for it.
- Reading articles and researches about sat solvers, minisat algorithm and encoding sudoku boards.
- Patience ☺

Theoretical material

What is SAT ??

- **SAT - Satisfiability Problem is a is a decision problem.**
- **Input: a boolean expression written by logic operations: AND, OR, NOT, variables and parenthesis by a CNF form.**
- **The decision: the sat algorithms decide if there is an assignment with TRUE and FALSE values that satisfies the given Boolean expression, this means that the assignment makes whole expression true.**
- **Output: an assignment as described before or the output that means that there is no assingmnts that can satisfy the given expression.**

Some definitions:

- A **literal**: is either a variable or the negation of a variable.
- A **clause** is a disjunction of literals.
- A Boolean expression: is a conjunctions of clauses

What is CNF?

- CNF is the form a writing boolean expression that almost all SAT solvers use.
- It's written in the way that each clause has only OR operation between it's literals, and between the clause only AND operations.

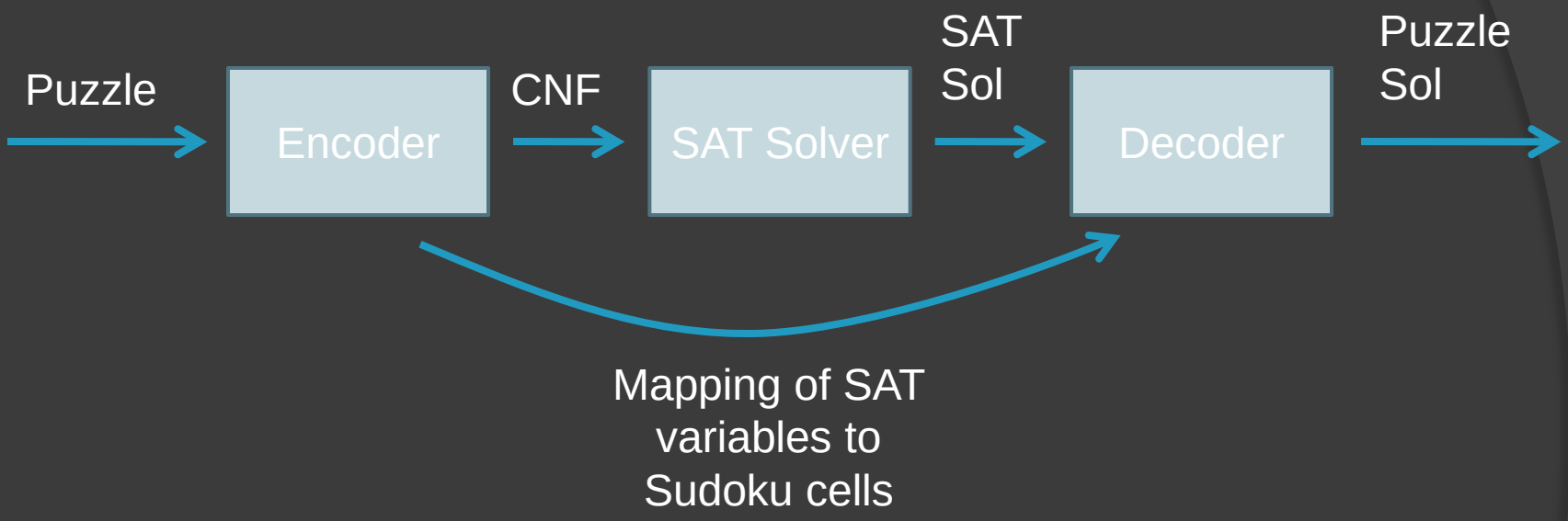
What is sudoku?

- Played on a $n \times n$ board.
- A single number from 1 to n must be put in each cell - some cells are pre-filled.
- Board is subdivided into $\sqrt{n} \times \sqrt{n}$ blocks.
- Each number must appear exactly once in each row, column, and block.

Example:

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

The process



The encoding part :encode the board to a boolean expression in the optimal way as we can.

- In our project we use a 3D matrix that each cell in that matrix is described by the tupple: (raw , column , value).
- We use this form because each cell, raw, column and block contains a number between 1→n.
- We devide the encoding process to 4 parts:
  - Cells part
  - Raws part
  - Columns part
  - Blocks part

- All the parts described before are Identical by the encoding, so we will describe the cells part.
- As described before each cell has to contain a number between 1→n,
- A cell is the array: (raw, column, v) such that “v” is a value from the range: 1→ sqrt(n).
- So only one cell of that array gets the true value. And all the others has to get false.

- In order to do the encoding we devide that to two parts:
  1) At least one true value in that array
  2) At most one true value in that array.

- And by doing the And logical operation between 1 and 2 we will get that only one cell has a true value.

Example of encoding the cells part

- As an example we will Simplify the array to three cells: X, Y and Z. such that these are three different values of a certain cell.
- in order to encode the first part: “At least one true value in that array” we do the above:
- X + Y + Z (+ = OR)
- This expression is satisfied only and only if at least one of these literals is true

- in order to encode the second part: “At most one true value in that array” we do the above:
- (!X + !Y) * (!X + !Z) * (!Y + !Z)

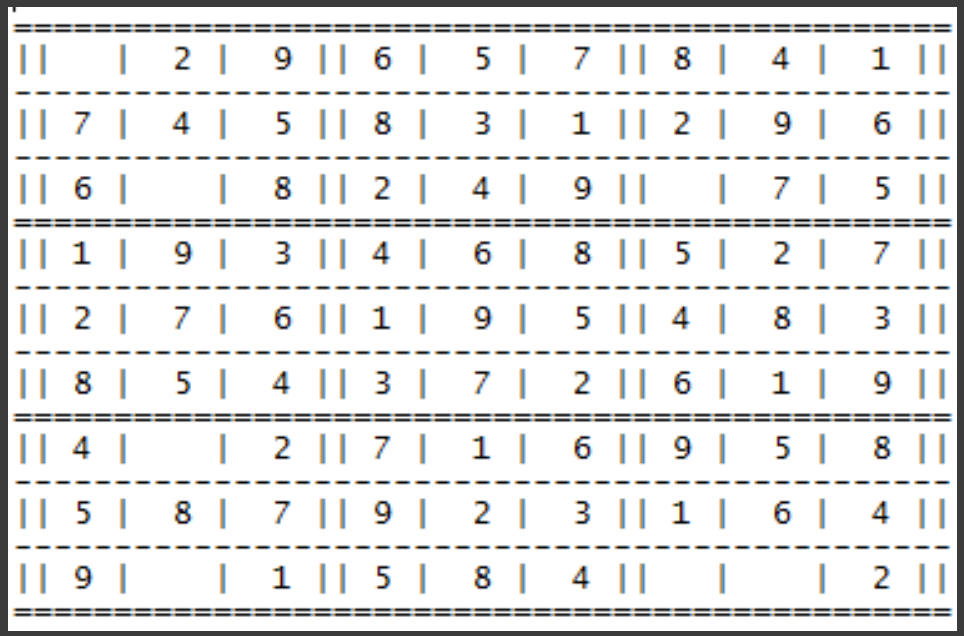
- The above expression is satisfied only and only if at most one cell is true, because if there are two cells are true then at least one clause won’t be satisfied.
- Then part1 * part2 → exactly one value is true.

- The last part of encoding is putting the And operation between all the clauses in order to ensure that the sudoku rules still alive ☺

- Next we use the minisat algorithm that decide of the expression is satisfied or not, if not that means that there is no solution to the given' input board and print this conculsion to the user, otherwise we take the minisat output and entering it to each cell in the matrix and in the last step we print the solution- the full sudoku board.

Example OF Board incoding

- As an iput we took this board:



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

- Then the incoding in this case is:

```
3 0
172 174 0
219 0
498 0
3 0
172 0
174 219 0
498 0
3 0
172 0
174 498 0
219 0
172 0
3 174 0
219 0
498 0
-172 -174 0
-174 -219 0
-3 -174 0
-174 -3 0
```

- And the output is in file that I will present to you and show you how we used it.

- Thanks ☺