Logic and Model Checking Fall 2021

Project 1: SAT

Lecturer: Jaime Ramos *Department of Mathematics, Técnico*

1 Sudoku

1.1 Classic Sudoku

A *Sudoku puzzle* consists of a 9×9 grid made up of 3×3 sub grids, sometimes known as *regions*. The aim of the puzzle is to ensure that in each row, column and region, there is exactly one instance of the numbers 1, 2, 3, 4, 5, 6, 7, 8 and 9. Initially, some numbers, referred to as *clues*, are provided in certain cells. A well-posed puzzle has a unique solution. On the left is an example of such a puzzle, where some clues are given, and on the right is the solution of the puzzle.

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

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

Encode the problem of determining the solution of a Sudoku puzzle as a SAT problem. Based on this encoding, and using Z3, define the following functions in Python:

1. `sudoku(P)` – function that given a Sudoku puzzle `P`, determines the solution, if there is one;
2. `well_posed(P)` – function that given a Sudoku puzzle `P`, determines if the puzzle is well posed, that is, if the puzzle has a unique solution;
3. `generate(S, pat)` – function that given a solution for a Sudoku puzzle `S` and a *pattern* `pat`, determines if removing from `S` the numbers in the cells given by the pattern, yields a well posed Sudoku puzzle, and, if so, generates the corresponding puzzle. A pattern is a 9×9 grid with binary entries such that 0 entries correspond to the positions to be removed, and 1 entries correspond to the positions to be maintained.

A puzzle is represented by a list of 9 lists of length 9. Each of these lists corresponds to a row (the first list corresponds to row 1, the second list corresponds to row 2, and so on). In each position of the list, corresponding to a column, there is either a number from 1 to 9, representing a clue, or 0 representing an empty cell. The puzzle above is represented by the list

```
[[5,3,0,0,7,0,0,0,0],
 [6,0,0,1,9,5,0,0,0],
 [0,9,8,0,0,0,0,6,0],
 [8,0,0,0,6,0,0,0,3],
 [4,0,0,8,0,3,0,0,1],
 [7,0,0,0,2,0,0,0,6],
 [0,6,0,0,0,0,2,8,0],
 [0,0,0,4,1,9,0,0,5],
 [0,0,0,0,8,0,0,7,9]]
```

1.2 Variants

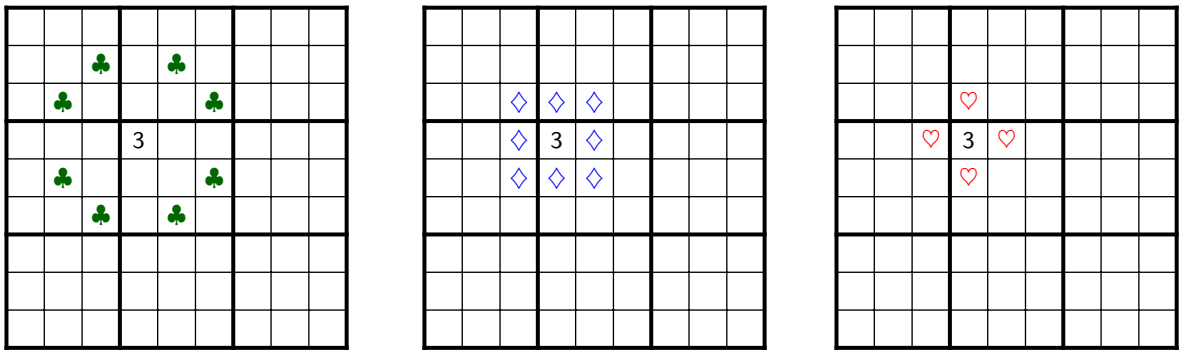
There are many variants to the Sudoku puzzle, where *additional* restrictions are added. Consider the following additional restrictions to the classical puzzle:

1. any two cells separated by a *Knight's move* or *King's move* cannot contain the same number;
2. any two orthogonally adjacent cells cannot contain consecutive digits.

The *Knight's move* restriction means that if we have a number in a cell then all cells that can be reached by a Knight's move cannot contain that number. For instance, in the situation depicted on the left, the cells marked with ♣ cannot contain the number 3 because of the Knight's move.

The *King's move* restriction means that if we have a number in a cell then all cells that can be reached by a King's move cannot contain that number. For instance, in the situation depicted in the middle, the cells marked with ♢ cannot contain the number 3 due to the King's move.

The restriction on *orthogonally adjacent cells* means that the cells left, right, above and below of a cell cannot contain a number consecutive to the number in that cell. For instance, in the situation depicted on the right, the cells marked with ♡ cannot contain the numbers 2 or 4 due to this restriction. Observe that 1 and 9 only have one consecutive number.



Capitalizing on the encoding of classic Sudoku, encode the problem of determining the solution for this variant of a Sudoku puzzle as a SAT problem. Based on this encoding, and using Z3, define, the following function in Pyhton:

1. `sudoku_var(P)` – function that receives a Sudoku puzzle `P` and determines the solution according to the new restrictions, if there is one.

Use `sudoku_var` to solve the following puzzle.

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

2 Sums

Let $R \subseteq \mathbb{N}$ be a set of natural numbers and let $t \in \mathbb{N}$ be a natural number. Consider the problem of deciding if there is a subset S of R such that the sum of its elements is exactly t, that is, if there is $S \subseteq R$ such that

$$\sum_{s \in S} = t.$$

For instance, given $R = \{1, 2, 3, 4\}$ and $t = 6$ we have that $S = \{1, 2, 3\}$ is a solution for the problem because $S \subseteq R$ and $1 + 2 + 3 = 6$. This is not the only solution as $S = \{2, 4\}$ also meets the requirements.

Encode this problem as a SAT problem. Based on this encoding, and using Z3, define the following function in Python:

1. `sums(R,t)` – function that given a set of natural numbers `R` and a natural number `t`, determines if there is a subset set `S` of `R` such that the sum of its elements is `t` and, if so, returns `S`. Otherwise, returns `false`.

A set of natural numbers is represented by a list of natural numbers.

3 Submission

The project is to be submitted in the *Fenix* platform by a member of the group (after the group has registered). The submission should consist of a single compressed folder containing:

- a report on the project, that should include the encoding of each problem as a SAT problem, and execution examples;
- one or more Jupyter notebooks with the implementation all requested functions;
- the slides for the oral presentation (a draft that can be improved after submission).

Submission deadline: **November 5, 2021, at 23:59.**