1 Sum-Sudoku

Consider the game of simplified (n, m) Sum-Sudoku. The game consists of an $n \times n$ square-grid. Each entry in the grid has to be filled using digits between 1 and m (both inclusive). The usual rules of Sudoku apply where a digit must appear only once in each row and each column. And in this simplified version of the game, we will ignore the "box"-constraint that Sudoku usually has. In addition, Sum-Sudoku requires that each row and each column sum to a specified value.

	8	8	12
8			
10			
10			

(a) (3,5) Sum-Sudoku puzzle.

	8	8	12
8	1	4	3
10	5	1	4
10	2	3	5

(b) (3, 5) Sum-Sudoku puzzle with solution.

Figure 2: Sum Sudoku

Figure 2a shows a (3, 5) Sum-Sudoku puzzle. A valid solution requires that the rows sum to 8, 8 and 10 respectively, while the columns should sum to 8, 8 and 12 respectively. Figure 2b shows its solution.

In this problem, we will use the Z3 SMT solver's Java API to investigate properties of Sum-Sudoku puzzles. Skeleton code for Scala that demonstrates use of the API is provided. You will implement specific functions in this skeleton code.

(a) Formulate an SMT instance that finds a solution to Sum-Sudoku puzzles. Assignment: In solution.txt:

- Describe your encoding and list the constraints in it. Write what theories are used in your formulation.
- Encode your formulation using the Z3 API by completing the implemen- tation of the function valid in the file Sumsudoku.scala.

(b) A Sum-Sudoku puzzle may not have a unique solution. Formulate a (quantified) SMT query that finds an assignment to the row and column sums such that the resulting puzzle has a unique solution. You should assume that the grid itself is empty and only the row and column sums are specified in the puzzle that you will be generating.

Assignment: In your solution:

- Describe this formulation and list its constraints.
- Encode your formulation using the Z3 API by completing the implemen- tation of create_puzzle in the file Createsudoku.scala.
- How scalable is your solution ? Play around with the values of m and n and discuss your findings.

(c) A more scalable method of generating Sum-Sudoku puzzles with unique solu- tions would be to start with a fully-filled out puzzle and repeatedly remove entries as long as the resulting puzzle has a unique solution.

Assignment: In your solution:

- Describe the algorithm for this formulation and list the constraints that are checked in each iteration.
- Encode your formulation using the Z3 API by completing the implemen- tation of make_puzzle_unique in the file Makeuniquesolution.scala.