**1. Introduction**

ITP 20002-02 Discrete Mathematics, Fall 2020

**Homework 1 – SudoKu-X**

Wonbin Lee, 21900543, 21900543@handong.edu

The task is solving Sudoku-X, which is a variant of the standard Sudoku puzzle. The grid of the given puzzle is 9x9. Also, the range of number that should be filled is from 1 to 9.

As the standard, sudoku-X has three constraints. ‘1. Every row has all 9 numbers’, ‘2. Every column has all 9 numbers’, ‘3. Every sub-grid has all 9 numbers.’ However, there is one more constraint for Sudoku-X, that is ‘4. Each of two diagonals has all 9 numbers.’

To apply these constraints to propositional logic to solve Sudoku-X, I introduced some propositional variables that represent to truth value whether a certain number is in a certain cell or not.

In addition, a constraint appears by introducing the propositional variable. That is “5. a cell can have exactly one number of 9 numbers.”

With the propositional variables, I conducted follow approaches to solve the puzzle during satisfying the constraints and considering how to use the propositional variables (More details of specific propositional variables and approaches are in section 2).

First, for constraint from 1 to 4, I use follow concept: the number of sizes of a row, a column, a sub-grid, a diagonal is 9, that is exactly same with the length of range of 9 numbers. It implies that what should be checked is just whether ‘there is at least one cell that has one of 9 numbers in a specific area, depending on what constraint is, for every numbers’ or not.

Second, for constraint 2, I separate it into two constraint. One is “each cell is assigned at most one number”, the other is “each cell is assigned at most one number.” On the first one, I apply this: there should be no pair of propositional variables that is true at the same time. And on the second one, I apply similar concept used constraint 1 to 4.

As well as the above constraints, I had to consider and realize the formula constructor and Z3 output interpreter because the final purpose this task is “making a program solving Sudoku-X program using proposition logic and Z3.”

Following section shows more detailed approaches.

1. **Approach**

In order to solve the problem using programming, I constructed all approaches in terms of program’s flow. The subsections are following.

* 1. Declare propositional variables.

I designed the propositional variable as p(i, j, n). ‘i’ means which row it is, ‘j’ means which column it is, and ‘n’ means one of the numbers from 1 to 9 (These definitions apply to all of this report). There are 727 (9x9x9) propositional variables, and all of them means “The cell (i, j) has the number n.” Those are propositional variables which can be either true or false.

* 1. Make the formula.

Q1. Checking the pre-assigned cells.

Since the given input can have some cells that have pre-assigned number, it is needed to check them. I use two-nested for loop to get input, a cell of 9 x 9 sudoku-x, one by one, from standard input. Also, I set the type of input value as ‘char.’ As the input is ‘?’ or one number of the number from 1 to 9, I firstly subtract ‘0’, from the input value, using ASCII code, because ASCII code ‘0’ is 48 in integer, and ‘1’ is 49 in integer, and so on. Then, check whether the output of subtraction is in the range from 1 to 9 or not, in terms of integer. If it is, I add the current position and that value to formula as propositional variable;).

Q2. Each cell is assigned with exactly one number

Q2-1. Each cell is assigned at most one number.

This constraint means, for every i and j, there should be no pair of propositional variables, satisfied at the same time. In terms of proposition logic, this can be represented by . This can be realized by conducting four-nested for loop.

Q2-2. Each cell is assigned at least one number.

This constraint means, for every i and j, there should be at least one p(i, j, n) satisfied. This can be represented by . This can be realized by conducting three-nested for loop.

Constraints from Q3 to Q6 are very similar to Q2-2.

Q3. Every row has all 9 numbers.

This constraint means, for every i and n, one of j needs to be satisfied. That is, when the i and n is fixed, one of 9 p(i, j, n) () should be satisfied, and this applies to every i and j. Fortunately, the size of each row is exactly same with the total number 9, from 1 to 9, the thing to check is whether there is at least a cell, in a row, that has a certain number of 9 numbers, for every row and number, or not. (The similar characteristic also applies to Q4, Q5, and Q6) This can be represented by . This can be realized by conducting three-nested for loop.

Q4. Every column has all 9 numbers.

This constraint means, for every j and n, one of i needs to be satisfied. Logic is similar to Q3. This can be represented by . This can be realized by conducting three-nested -for loop.

Q5. Every sub-grid has all 9 numbers.

This constraint means, for every sub-grid and n, there is at least one cell (i, j) satisfied. This can be represented by

The main idea to satisfy this is also similar to Q3, Q4, that is, there are at least a cell, in a sub-grid, that has certain number of 9 numbers for every number and every sub-grid. This can be represented by ‘r’ means the row of sub-grid, ‘s’ means the column of sub-grid. By taking 3(r or s) + (i or j), checking all the cells in the puzzle, sub-grid by sub-grid, become possible. Also, this can be realized by conducting five-nested for loop.

Q6. Each of two diagonals has all 9 numbers.

Q6-1. Diagonal from left-top to right-bottom.

This constraint means, for every number, at least one cell (i, j) in that diagonal has one of 9 numbers. In this case, i and j is same with each other. This can be represented by . This can be realized by conducting two-nested loop.

Q6-2. Diagonal from right-top to left-bottom.

This constraint means, for every number, at least one cell (i, j) in that diagonal has one of 9 numbers. This case, i starts from 1, and j starts from 9. This can be represented by . This can be realized by conducting two-nested for loop.

* 1. Pass the formula to Z3 and save the solution to a result file.

Once Z3 generate a solution from given formula, read the first line of it. If it is ‘unsat’, print about ‘unsat’ and go to the end of program. If not, start reading the left of the solution file, and save the propositional variables, which is true, to a result file.

* 1. Pass the solution to Z3 output interpreter.

Declare a 2-D (9x9) array for saving the number from result file and start reading solution file. While reading, extract i, j from value of second index of a line of solution file and third index. Then, using those i and j, fill the cell (element of array) of position (i, j) with n, which is from the number of third index in the same. Finally, if the reading is finished, print the 2-D array using two-nested for loop. That is the result of solved problem.

**3. Evaluation**

To say “the task is successful” in this assignment, the program should firstly distinguish the satisfiable inputs and the unsatisfiable inputs. Second, it should make the desirable result when the input is satisfiable.

First, I tested program with satisfiable inputs such as a case consisting of all of question marks, example input from homework1.pdf, and examples from online. I tested it with five inputs, and all results were desirable.

Second, I tested program with unsatisfiable inputs such as duplicated number in a row, column, sub-grid, and diagonals. The program coped with every case, as it printed “There is no solution for given problem!”.

Therefore, all the tasks were successfully accomplished.

**4. Discussion**

When I finally made a program and I’m sure that my program successfully accomplishes all of the tasks, I got to know the power of mathematical logic. I surprised at that I can make a wonderful program with mathematical logic and basic grammar of programming. This make me realize how important the mathematical logic is in programming.

**5. Conclusion**

I have reported how I access this problem and solve it with propositional logic and programming so far.

There were total six constraint need to be satisfied, and I realize them as a formula with certain concepts and proposition logics. Also, I made a proper Z3 output interpreter, and it worked well.

After that, I check the validity of the implemented program, and it coped with every proper case.

With this, all the tasks are successfully accomplished.