**1. Introduction**

ITP 20002-02 Discrete Mathematics, Fall 2020

**Homework 1 – Easy As ABC**

Lee Wonbin, 21900543, [21900543@handong.edu](mailto:21900543@handong.edu).

The task is solving Easy-As-ABC. The given input consists of ss gird ( with letters (‘A’, ‘B’, ‘C’, ‘D’, ‘E’) annotated at the beginning/end points of some rows and columns. Filling cells properly with a letter with satisfying given and found constraints is the key to solve this problem.

There are total three constraints from given constraints of the problem, to be satisfied to solve the problem. ‘1. The five letters must appear exactly once at every row.”, “The five letters must appear exactly once at every column.” “3. Every given letter annotated in a row or column must be closest letter in the corresponding row or column.”

In addition to above constraints, there is one more constraint added by introducing propositional variables that represent to truth value whether a certain letter is in a certain cell or not. It is, “4. Every none-empty cell, that has a letter, is assigned with exactly one number.”

I used complementary-constraint approaches to satisfy constraint 1, 2, making specific cells always false – strategy to 3, and the concept of “There should be no pair satisfied at the same time’ to 4 (see more details in section 2).

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 Easy- program using proposition logic and Z3.”

Following section shows more detailed approaches.

**2. Approach**

In order to solve the problem using programming, I constructed all approaches in terms of program’s flow, and the structure is like this:

2.1. Declare propositional variables.

2.2. Make the formula.

- Checking annotations.

- Every none-empty cell should have exactly one letter.

- The five letters must appear exactly once at every row and column.

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

* 1. Pass the solution to Z3 output interpreter.

The above subsections have organic relationships and flow within a program. The following paragraphs will talk about each of the subsections.

* 1. Declare propositional variables.

I designed the propositional variable as p(i, j, l). ‘i’ means which row it is, ‘j’ means which column it is, and ‘l’ means one of the letters from ‘A’ to ‘E’ (These definitions apply to all of this report). There are 405 (9x9x5) propositional variables, and all of them means “The cell (i, j) has the letter a.” Those are propositional variables which can be either true or false.

* 1. Make the formula.

Q1. Checking the annotations.

Since the format of given input is first the size (s) of gird and the annotations of top – bottom – right – left (each is a new line), I checked the annotations of each area in turn. First, while reading the four area representing annotations, I checked if the cell I'm currently reading is (annotation) letter or not. Second, if it so, I used “making specific cells always false” strategy, as I said, to assert that the annotated letter is the closet letter from the annotation. That strategy is making every propositional variables of every cell before (or after) a certain cell that has annotated letter as false, for every possible cell that can has the annotated letter. Using the following three characteristics, “the number of letters is 5”, “the possible size of gird is from 5 to 10”, and “the number of left letters (not annotated at that row or cell) is 4”, help to realize the strategy. Also, it is important to divide the case – top, bottom, right, left – and access them properly. Lastly, in terms of code, there are five-nested loop. The two outer loops are for reading the given inputs, and the three inner loops are for making all possible propositional variables, in the specific cells before (or after) the cell having annotated letter, false properly.

Q2. Every none-empty cell should have exactly one letter.

This means, the every none-empty cell is assigned ‘at most’ one letter. For this, assumption that every cell could be assigned is needed because distinguishing empty cell and none-empty cell will be conducted at Q3 and Q4. Because of that, the thing turned to easy. For every row and column, there should be no pair satisfied at the same time. This is the logic. This can be represented by The concept of ASCII code is used in here, ‘m=n+1.’ Also, this can be realized by conducting four-nested for loop.

Q3. The five letters must appear exactly once at every row.

As I said at the introduction, I used complementary-constraint approach on this. In more detail, I separated this constraint into two: ‘1. Every row has all 5 letters” and “2. Every letter must appear only once at every row” (This may be similar to the original). The first one can be satisfied by using this logic: “for every row and letters, there are at least one cell that has the letter.” The second one can be satisfied by using this logic: “for every row and letters, there should be no pair of propositional variables that is true at the same time”. Through combining those two constraints, it is possible to satisfy the Q2 perfectly. (This approach also applies to Q3.)

The first one can be represented by in terms of proposition logic. The second one can be represented by These can be realized by conducting three-nested for loop and four-nested for loop for each one.

Q4. The five letters must appear exactly once at every column.

The logic of this is exactly same with the Q2. Q2 is enough to explain this. Therefore, I’ll just show the proposition logic for each of separated constraint.

The first one can be represented by in terms of proposition logic. The second one can be represented by These can be realized by conducting three-nested for loop and four-nested for loop for each one.

* 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 (ss) array for saving the letter from result file and initialize all of the elements of that array as ‘\_’. And start reading solution file. While reading, extract i, j from a line (propositional variable) of solution file. Then, using those i and j, fill the cell (element of array) of position (i, j) with n, which is from the last index in a line. 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 the program with unsatisfiable inputs, I made, such as “duplicated annotation on same column or same row”, “duplicated annotation on same row and column”, and “crossed(duplicated) annotation over (top or bottom) and (right or left).” I conducted it with all possible size, from 5 to 10, and the program coped with every case, as it printed “There is no solution for given problem!”.

Second, I tested the program with satisfiable inputs such as various size of cases that consisting of all of none-annotations, example input from homework1.pdf, and examples from online. I conducted it with nine inputs, and all results were desirable.

Therefore, all the tasks were successfully accomplished.

**4. Discussion**

Fortunately, the solution of this task is using the proposition logics that I have learned from class, especially the N-Queen problem. By applying similar proposition logic to another puzzle problem, I found that proposition logic was fundamental to solving puzzles, and I was grateful that abstract thinking could be arranged into proposition logic.

**5. Conclusion**

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

There were four constraints, each for annotated letters, cells, rows, and columns. I approached them with various strategy, such as “making all possible propositional variable false”, complementary-constrain approach, and so on. Through them, I could construct desired formula and 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 were successfully accomplished.