Discrete Mathematics 02

# Grop6

21800147 Youyoung Kim 21500706 Sungmin Jee
21600062 Miso Kim 21500388 Jane An
21400636 Yechan Lim 21800294 Suah Park

Puzzle 1. Sudoku*

In Sudoku*, there is a variant of Sudoku which has a 9x9 grid with nine 3x3 sub-grids, each cell that has a number in 1 to 9 and certain cells that contain asterisk sign (*) with unique one value in 1 to 9.

[Constraint and Logic]

1. Each cell has a number in 1 to 9

$$\bigwedge_{i=1}^{9}\bigwedge_{j=1}^{9} 1 \le a_{i,j} \le 9$$

2. Every row contains every number.

$$\bigwedge_{i=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{j=1}^{9} a_{i,j,n}$$

3. Every column contains every number.

$$\bigwedge_{j=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{i=1}^{9} a_{i,j,n}$$

4. Each 3x3 box contains every number.

$$\bigwedge_{i=0}^{2}\bigwedge_{j=0}^{2}\bigwedge_{n=1}^{9}\bigvee_{h=1}^{3}\bigvee_{k=1}^{3} a_{3i+h,3j+k,n}$$

5. Asterisk(*)

a. at most 9 asterisk sign
b. no two cells marked with Asterisk have a same number

$$\bigvee_{i=1}^{9}\bigvee_{j=1}^{9} p_{i,j}\text{ , at most } 9\ p_{i,j} \text{ has asterisk sign,}$$
$$\text{each } p_{i,j} \text{ has distinct value between } 1 \text{ to } 9$$

[Understanding about 'distinct' of Z3]

What we found about a function of Z3 'distinct' mean is when the elements of the final domain are enumerated as distributed components, 'distinct' serves to ensure that the factors contained in them do not have overlapping values.

[Processing]

I. List all conditions that satisfy Sudoku.

II. Create logic using Conjunction and disjunction.

III. Make a C language code in Quantifier-Free LIA form.

IV. Create c program that outputs the languages applicable to SMT Solver Z3.

V. Receive a standard input file and execute Z3 and output the solved Sudoku result as standard output.

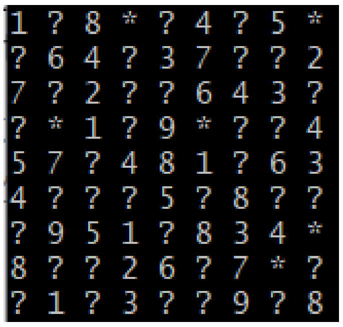
[Discussion]

In the process of creating Z3 formula, when we create in c language, we must consider the order in which the characters are read from the input file. In contrast, we found that z3 processes constraints without priority and produces results.

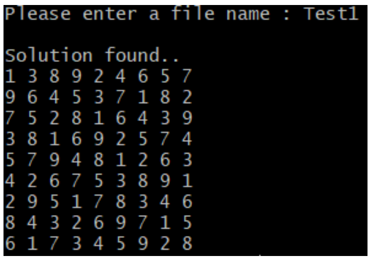
We could have received the input as a file, or we could have input directly through the keyboard. But we chose the way we received the file because we thought it would be more convenient to run the file directly when it was given However, I think it is quick to modify the input input through the keyboard, and actually it will be practical when someone wants to solve any problem with Sudoku.

[Program Testing]

1. Test1 – Correct input

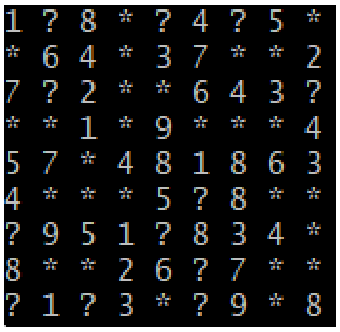


```
1 ? 8 * ? 4 ? 5 *
? 6 4 ? 3 7 ? ? 2
7 ? 2 ? ? 6 4 3 ?
? * 1 ? 9 * ? ? 4
5 7 ? 4 8 1 ? 6 3
4 ? ? ? 5 ? 8 ? ?
? 9 5 1 ? 8 3 4 *
8 ? ? 2 6 ? 7 * ?
? 1 ? 3 ? ? 9 ? 8
```

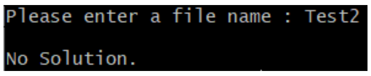


```
Please enter a file name : Test1

Solution found..
1 3 8 9 2 4 6 5 7
9 6 4 5 3 7 1 8 2
7 5 2 8 1 6 4 3 9
3 8 1 6 9 2 5 7 4
5 7 9 4 8 1 2 6 3
4 2 6 7 5 3 8 9 1
2 9 5 1 7 8 3 4 6
8 4 3 2 6 9 7 1 5
6 1 7 3 4 5 9 2 8
```

2. Test2 – Too many asterisk sign (more than 9)

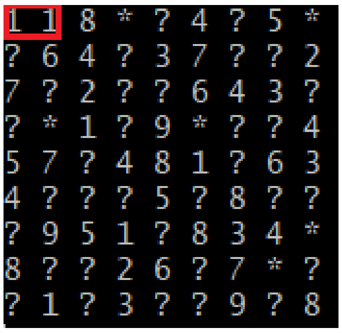


```
1 ? 8 * ? 4 ? 5 *
* 6 4 * 3 7 * * 2
7 ? 2 * * 6 4 3 ?
* * 1 * 9 * * * 4
5 7 * 4 8 1 8 6 3
4 * * * 5 ? 8 * *
? 9 5 1 ? 8 3 4 *
8 * * 2 6 ? 7 * *
? 1 ? 3 * ? 9 * 8
```

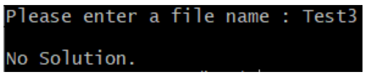


```
Please enter a file name : Test2

No Solution.
```

3. Test3 – The same number in one row

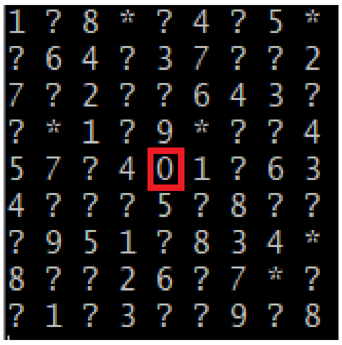


```
1 1 8 * ? 4 ? 5 *
? 6 4 ? 3 7 ? ? 2
7 ? 2 ? ? 6 4 3 ?
? * 1 ? 9 * ? ? 4
5 7 ? 4 8 1 ? 6 3
4 ? ? ? 5 ? 8 ? ?
? 9 5 1 ? 8 3 4 *
8 ? ? 2 6 ? 7 * ?
? 1 ? 3 ? ? 9 ? 8
```

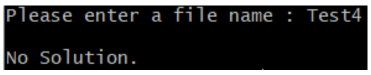


```
Please enter a file name : Test3

No Solution.
```

4. Test4 – Zero in a sudoku

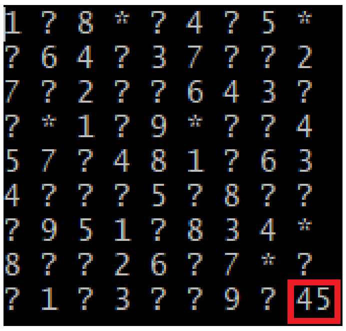


```
1 ? 8 * ? 4 ? 5 *
? 6 4 ? 3 7 ? ? 2
7 ? 2 ? ? 6 4 3 ?
? * 1 ? 9 * ? ? 4
5 7 ? 4 0 1 ? 6 3
4 ? ? ? 5 ? 8 ? ?
? 9 5 1 ? 8 3 4 *
8 ? ? 2 6 ? 7 * ?
? 1 ? 3 ? ? 9 ? 8
```

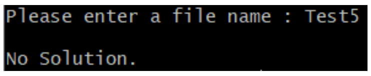


```
Please enter a file name : Test4

No Solution.
```

5. Test5 – Two numbers in one cell



```
1 ? 8 * ? 4 ? 5 *
? 6 4 ? 3 7 ? ? 2
7 ? 2 ? ? 6 4 3 ?
? * 1 ? 9 * ? ? 4
5 7 ? 4 8 1 ? 6 3
4 ? ? ? 5 ? 8 ? ?
? 9 5 1 ? 8 3 4 *
8 ? ? 2 6 ? 7 * ?
? 1 ? 3 ? ? 9 ? 45
```



```
Please enter a file name : Test5

No Solution.
```

Puzzle 2. Fill -a -Pix

Fill -a -Pix has random numbers between 1-9 of N*M grids. Of the total nine cells, including eight cells surrounding that clue number and one with clue number, the sum of the cells in black must be the same.

[Constraints]

1. The number (value) of each cell must be between 0 and 9.
2. The value of N and M shall not exceed 1000.
3. All cells must be white or black. (The program defines white as 0 and black as 1.)
4. If a solution does not exist, it can be printed as "No Solution" and up to five solutions can be printed.
5. Each cell is $a_{ij} \Rightarrow \quad 0 \leq i \leq M-1$ and $0 \leq j \leq N-1$

[Logic Formula]

1. Declare each cell "Int"
   - ➔ X = $\{a_{00}, a_{01}, \dots, a_{m-1n-1}\}$
   - ➔ Declare const X Int
2. Assert $\forall a_{ij} \in X \ (0 \leq a_{ij} \leq 1)$
3. The sum of values around the clue number including cells in the clue number must be the same as the clue number.

$$a_{i-1j-1} + a_{i-1j} + a_{i-1j+1} + \cdots + a_{i+1j-1} + a_{i+1j} + a_{i+1j+1} = \sum_{i-1}^{i+1}\sum_{j-1}^{j+1} a_{ij} = clue\ number$$

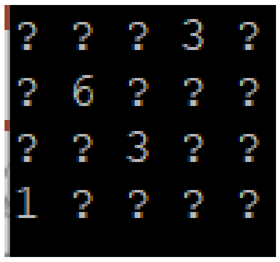
4. The surrounding cells speaking in 'c logic' must meet the conditions of

$$j-1 \geq 0, \qquad j+1 \leq N-1$$

5. No solution : print no solution if no answer meets the given conditions.
6. Multiple solutions : If one solution is available, look for another solution is available. Find values which are not '?'. Change the values of the nine cells around the value, including the input value, one by one, to see if another solution is available. If a cell is used as a_{ij}, the condition $0 \leq i \leq M-1$ and $0 \leq j \leq N-1$ must be met.

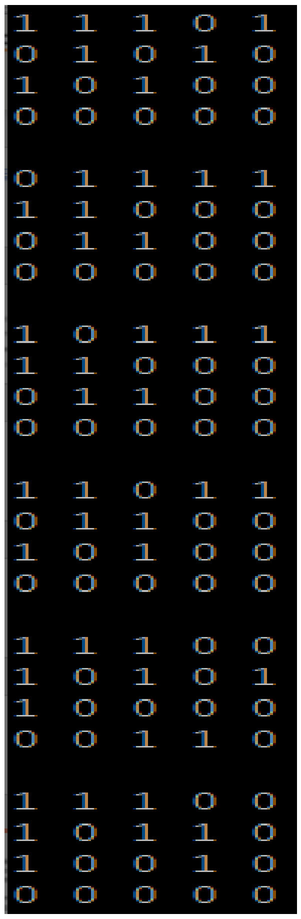
[Program Testing]

1. If there are more than five solutions, print only five solutions.
   a. Input



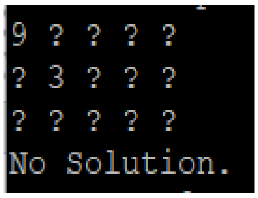
```
? ? ? 3 ?
? 6 ? ? ?
? ? 3 ? ?
1 ? ? ? ?
```

   b. Output



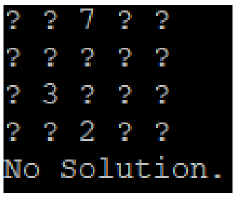
```
1 1 1 0 1
0 1 0 1 0
1 0 1 0 0
0 0 0 0 0

0 1 1 1 1
1 1 0 0 0
0 1 1 0 0
0 0 0 0 0

1 0 1 1 1
1 1 0 0 0
0 1 1 0 0
0 0 0 0 0

1 1 0 1 1
0 1 1 0 0
1 0 1 0 0
0 0 0 0 0

1 1 1 0 0
1 0 1 0 1
1 0 0 0 0
0 0 1 1 0

1 1 1 0 0
1 0 1 1 0
1 0 0 1 0
0 0 0 0 0
```

2. No solution if clue number in corner is more than 5.



```
9 ? ? ? ?
? 3 ? ? ?
? ? ? ? ?
No Solution.
```

3. No solution when clue number of sides other than edge is greater than 7

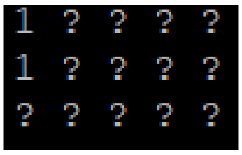


```
? ? 7 ? ?
? ? ? ? ?
? 3 ? ? ?
? ? 2 ? ?
No Solution.
```

[Discussion]

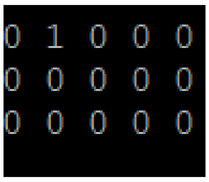
1. We found two problems in the result.

   a. When looking for a solution, find the input number and change the nine values around it. This may result in overlapping values around different input. If input is entered as shown below, find the first input value, 1, and change the value of the surrounding cells, including 1. First, since the value of a_{00} is zero, it replaces a_{00} with one to find solutions. With this sequence, the second and sixth solutions have the same values.



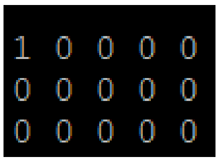
```
1 ? ? ? ?
1 ? ? ? ?
? ? ? ? ?
```

<input>



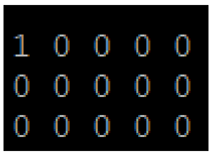
```
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
```

<solution1>



```
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

<solution2>



```
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

<solution6>

   b. If the input value is high or the process of finding solutions is prolonged, multiple solutions are not found. To find a solution, it needs to use the fopen and popen several times, but if the process is prolonged, the memory quota is thought to be exceeded.

2. What was interesting about solving this problem was "popen". As I said earlier, popen has been used repeatedly. If it had not been for the use of popen, it would have required an effort to input every single one directly. However, by using popen, I was able to save the trouble by taking care of myself in the program.

3. At first, we couldn't figure out how to weave the code, so we started to write the code in the main function without any function. We were working on the code continuously, and it was only in the main function. Because we did not use any other function, it was difficult to correct the code or check the wrong part.

Puzzle 3. Numbrix

Numbrix is a type of logic puzzle. In this puzzle, a rectangular grid of N x M is given. This puzzle's purpose is to fill in the blanks in order. The puzzle only runs vertically or horizontally. Diagonal paths are not allowed. When the puzzle is completed, the grid becomes a line in order of sequential number. Start number is 1, and the number goes to N x M.

[Constraint and Logic]

1. Numbrix consists of N x M grid.
   (N and M both less than 100)
   $numbrix[N][M]\ (when, N \leq 100, M \leq 100)$

2. Every cell value is between 1(first value) and N x M(last value).

$$\bigwedge_{y=1}^{N}\bigwedge_{x=1}^{M} 1 \leq a_{yx} \leq \mathrm{N\,x\,M}$$

3. Every cell is integer.

$$\bigwedge_{y=1}^{N}\bigwedge_{x=1}^{M} a_{yx} = [\mathrm{z}]\ (z\ is\ an\ number)$$

4. Every cell value must be different.

$$\bigwedge_{y=1}^{N}\bigwedge_{x=1}^{M}\bigwedge_{w=1}^{N}\bigwedge_{u=1}^{M} a_{yx} \neq a_{wu}\ (when, y \neq w\ and\ x \neq u)$$

5. All cells, except the cell that has first value, have own -1 value on the left, right, up, or down sides.

$$\bigwedge_{y=1}^{N}\bigwedge_{x=1}^{M}(a_{yx} \neq 1)$$
$$\bigwedge\{(a_{y+1x} = a_{yx} - 1)\bigvee(a_{y-1x}a_{yx} - 1)$$
$$\bigvee(a_{yx+1} = a_{yx} - 1)\ \bigvee(a_{yx-1} = a_{yx} - 1)\}$$

6. All cells, except the cell that has first value, have own -1 value on the left, right, up, or down sides.

$$\bigwedge_{y=1}^{N}\bigwedge_{x=1}^{M}(a_{yx} \neq NM)$$
$$\bigwedge\{(a_{y+1x} = a_{yx} + 1)\ \bigvee(a_{y-1x} = a_{yx} + 1)$$
$$\bigvee(a_{yx+1} = a_{yx} + 1)\bigvee(a_{yx-1} = a_{yx} + 1)\}$$

7. Cell that has first value has 2 on the left, right, up, or down sides.

$$\bigvee_{y=1}^{N}\bigvee_{x=1}^{M}(a_{yx} = 1)$$
$$\bigwedge\{(a_{y+1x} = 2)\bigvee(a_{y-1x} = 2)$$
$$\bigvee(a_{yx+1} = 2)\bigvee(a_{yx-1} = 2)\}$$

8. Cell that has last value has (last value – 1) on the left, right, up, or down sides.

$$\bigvee_{y=1}^{N}\bigvee_{x=1}^{M}(a_{yx} = NM)$$
$$\bigwedge\{(a_{y+1x} = NM - 1)\bigvee(a_{y-1x} = NM - 1)$$
$$\bigvee(a_{yx+1} = NM - 1)\bigvee(a_{yx-1} = NxM1)$$

[Processing]

I. List all conditions that satisfy Numbrix.
II. Create logic using Quantifier-free Logic.
III. Create Example input.txt File, and test our logic with it.
IV. Make a C language code in Quantifier-Free LIA form.
V. Create c program that outputs the languages applicable to SMT Solver Z3.
VI. Receive a standard input file and execute Z3 and output the solved Numbrix puzzle result as standard output.

[Discussion]

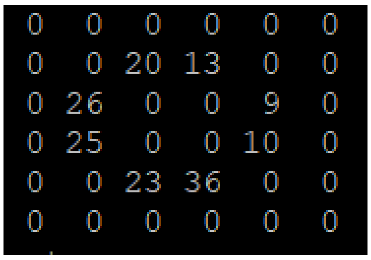
It failed to solve this Numbrix puzzle with z3 solution. We think the reason is probably due to misconfiguration of the logic. Specially, we were difficult to consider number One, and last number (N*M) in Numbrix Puzzle. Because this is bothering us, when thinking any logic, by making constraints. So we cannot solve this problem, and submit our imperfect result.
However, the thing that was interesting to do this assignment was by far the z3. Despite listing only a set of conditions, it was curious to know the intermediate results and the reason for the error, solving the problem on its own.

In addition, through an unfamiliar vi editor, I could learn to edit and execute .c files and think about problem-solving methods such as Puzzle with Quantifier-Free Logic. Although I have used only 6 x 6 grids of map, if I can make a proper formula, I would like to try to solve the maximum size of 100 by 100 Puzzle.

[Program Testing]

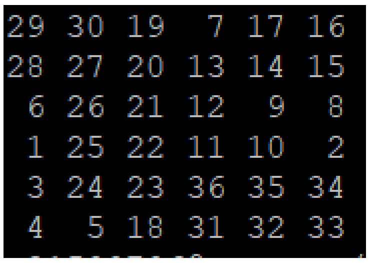
1. Test1 – Correct input

(Incorrect answer)



```
 0  0  0  0  0  0
 0  0 20 13  0  0
 0 26  0  0  9  0
 0 25  0  0 10  0
 0  0 23 36  0  0
 0  0  0  0  0  0
```

<input.txt>

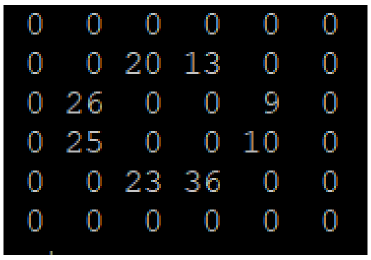


```
29 30 19  7 17 16
28 27 20 13 14 15
 6 26 21 12  9  8
 1 25 22 11 10  2
 3 24 23 36 35 34
 4  5 18 31 32 33
```

<output1.txt>

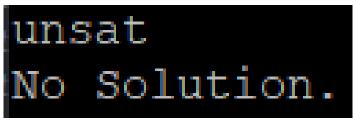
2. Test2 – Code change 1

(Logic collision)



```
 0  0  0  0  0  0
 0  0 20 13  0  0
 0 26  0  0  9  0
 0 25  0  0 10  0
 0  0 23 36  0  0
 0  0  0  0  0  0
```

<input.txt>

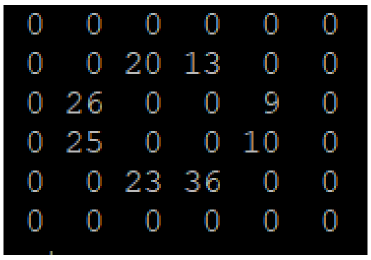


```
unsat
No Solution.
```

<output.txt>

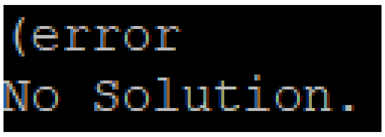
3. Test3 – Code change 2

(Formula Error)



```
 0  0  0  0  0  0
 0  0 20 13  0  0
 0 26  0  0  9  0
 0 25  0  0 10  0
 0  0 23 36  0  0
 0  0  0  0  0  0
```

<input.txt>

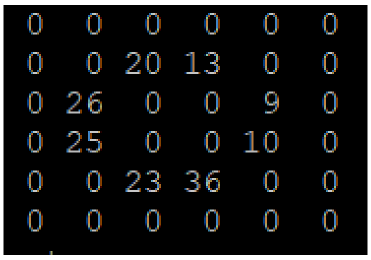


```
(error
No Solution.
```

<output.txt>

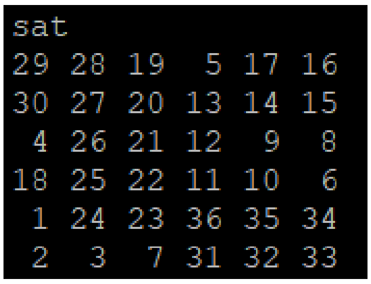
4. Test4 – Code change 3

(Logic collision)



```
 0  0  0  0  0  0
 0  0 20 13  0  0
 0 26  0  0  9  0
 0 25  0  0 10  0
 0  0 23 36  0  0
 0  0  0  0  0  0
```

<input.txt>



```
sat
29 28 19  5 17 16
30 27 20 13 14 15
 4 26 21 12  9  8
18 25 22 11 10  6
 1 24 23 36 35 34
 2  3  7 31 32 33
```

<Output.txt>