**<Discrete Mathematics Programming Assignment 1>**

[Team 4]

21700097 김민주, 21700525 이수아, 21500394 양수진, 21500601 임아론, 21500273 박수현

1. Introduction
2. Program design

The PuzzleSolver is written in Java language. We also designed it to use the apache cli library to make it easier for users to run programs.

Input : input.txt in ./PuzzleSolver

Output : formula.txt and output.txt in ./PuzzleSolver

Use -I input option to save the name of input txt file. Note that the input file must be saved in the folder where the executable file is located.

eg . your computer path/ Puzzle\_Solver/ (puzzle and your os type)/ build/ distributions/ PuzzleSolver/ bin/ (your input.txt)

Follow the manual, you can get the out.txt file.

Caution! The formula.txt and output.txt files are created each time when you run the program. So be careful not to overwrite previous information with new information.

The initial PuzzleSolver program was made using the c language. However, because of the following reasons, we implemented PuzzleSolver using java.

First, java uses Java Virtual Machine, So it can be run on any OS.

Second, java provides many convenient libraries that developers can use.

Our team were hard at parsing in c, but it was easy in Java. There are a lot of things to consider when parsing string in c. We have to compare the characters of the line one by one to find the part to parse, or We have to think about the type of variable type to parse line and the type of variable to copy. However, this complicated problem can be solved at once through a contains in string method and a regular expression.

Third, java has more stability of the code than c. We designed the PuzzleSolver using the excellent editor Eclipse. Also, java is more safe in the danger of pointers when using c language.

Fourth, Java has an apache-cli that makes it easy to use input command options.

So, Users are easy to understand what commands to put through the h option. We can also reduce the error when user insert the input, since each option will not cause any problems if the order of the options changes.

Finally, java has better access to github than c. The open source repository based on git is github. Therefore, It is optimized and convenient for collaboration and distributing programs.

Based on these reasons, we have redesigned the PuzzleSolver using the java language instead of C language.

1. Manual

This PuzzleSolver use the apache cli library.

1. Open terminal or PowerShell or cmd. And please type the following instruction.

git clone

1. Go to ‘Puzzle\_Solver’ folder.
2. Running Program in Mac
3. Sudoku

cd Sudoku\_Mac; gradle distZip; cd build; cd distributions; unzip PuzzleSolver.zip; cd PuzzleSolver; cd bin; cp ../../../../input.txt input.txt; cp ../../../../z3 z3; ./PuzzleSolver -i input.txt

1. Kakurasu

cd Kakurasu\_Mac; gradle distZip; cd build; cd distributions; unzip PuzzleSolver.zip; cd PuzzleSolver; cd bin; cp ../../../../input.txt input.txt; cp ../../../../z3 z3; ./PuzzleSolver -i input.txt

1. 3-in-row

cd 3InARow\_Mac; gradle distZip; cd build; cd distributions; unzip PuzzleSolver.zip; cd PuzzleSolver; cd bin; cp ../../../../input.txt input.txt; cp ../../../../z3 z3; ./PuzzleSolver -i input.txt

1. Running Program in Window
2. Sudoku

cd Sudoku\_Win; gradle distZip; cd build; cd distributions; unzip PuzzleSolver.zip; cd PuzzleSolver; cd bin; cp ../../../../input.txt input.txt; cp ../../../../z3.exe z3.exe; .\PuzzleSolver.bat -i input.txt

1. Kakurasu

cd Kakurasu\_Win; gradle distZip; cd build; cd distributions; unzip PuzzleSolver.zip; cd PuzzleSolver; cd bin; cp ../../../../input.txt input.txt; cp ../../../../z3.exe z3.exe; .\PuzzleSolver.bat -i input.txt

1. 3-in-row

cd 3InARow\_Win; gradle distZip; cd build; cd distributions; unzip PuzzleSolver.zip; cd PuzzleSolver; cd bin; cp ../../../../input.txt input.txt; cp ../../../../z3.exe z3.exe; .\PuzzleSolver.bat -i input.txt

1. Program Model
2. Logic of program
3. Sudoku
4. Logic
5. Each number is ranged 1~9
6. Define the number that already assigned to have that number.
7. Make sure that each row has a different number that is, the numbers from 1 to 9 must be all on one line.
8. Make sure that each column has a different number that is, the numbers from 1 to 9 must be all on one line.
9. Make sure that each 3X3 room has a different number that is, the numbers from 1 to 9 must be all on one room.

In conclusion, if you satisfy all this conditions, you can solve the problem of sudoku.

1. The role of z3 and function
2. Define all the array from a11 to a99 has integer type through ‘define’
3. Add the condition that each of number is different through ‘distinct’
4. Add all the conditions through ‘assert’ function.
5. Find the numbers that are satisfied all these conditions through ‘check’ function.
6. Kakurasu
7. Logic
8. Suppose X is the number 1, and O is the number 0. Therefore, the number of digits is 1 or 0. The reason why X is 1 and o is zeroed is that if O is left to zero, the position of O is not added but added to the position of X because it automatically becomes zero by any number.
9. Based on one line (number of columns where X exists \* 1)+(number of columns where O exists \* 0), this formula applies to 8 O or X present in a row.
10. Apply this to all rows, then the sums will be numbers on the right side of the puzzle.
11. Based on one line (number of rows where X exists \* 1)+(number of rows where O exists \* 0), this formula applies to 8 O or X present in a row.
12. Apply this to all columns, then the sums will be numbers on the numbers at the bottom of the puzzle.

Meeting all these conditions will solve the kakurasu problem.

1. The role of z3 and function.
2. Declare 8X8 variables.
3. Assert that all variables are zero or one.
4. Based on the column, assert that the sum of the values of the variable multiplied by the row number of each variable are the numbers of the right side of the puzzle.
5. Base on the row., assert that the sum of the values of the variable multiplied by the column number of each variable are the numbers of the bottom side of the puzzle.
6. 3-in-row
7. Logic
8. Assume that each cell is a[row][column] to make it easier to call an 8X8 array of 3-in-rows.
9. Assume that each box contains two values, O and X, and that values that have not yet been assigned values are filled with symbols.
10. To easily count the number of each O and X, temporarily assign a value of 1 to the box containing O and a value of 0 for X.
11. Since the numbers of O and X in all rows and columns must be the same, each row and column should contain 4 O and 4 X.
12. According to rule 4, set each box should have a value that makes each row and column have a value 4 that is the sum of the assigned numbers.
13. O or X cannot appear more than 3 times in any direction. That is, the sum of the numbers assigned to the three consecutive rows and columns should be 1 or 2.
14. According to rule 6, begin with a[1][1] and set the sum of the numbers assigned to each of the three neighboring cells to be 1 or 2 sequentially.
15. When all the columns of the array are filled with numbers, 0 is substituted for X and 1 is replaced by O.

The 3-in-row puzzle is solved in this process

1. The role of z3
2. Use Declare-const to specify that all numbers assigned to each field have an int type.
3. Find all arrays of 0 or 1 values that satisfy all of the above logic and print one of them.
4. If the value of the array satisfying all of the above logic does not exist, it is marked as unsatisfied.
5. Demonstration of all program work.

Sudoku’s output kakurasu

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| O | X | O | O | X | O | X | O |
| O | O | X | X | X | X | O | O |
| O | X | X | X | O | X | X | X |
| O | O | X | X | X | X | X | X |
| X | X | O | O | X | X | X | X |
| O | X | O | X | X | X | X | O |
| O | O | O | O | O | X | O | O |
| X | O | O | O | O | O | O | O |

-> Satisfy all rules of sudoku -> Satisfy all rules of kakurasu

3-in-row

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| O | O | X | O | X | O | X | X |
| X | X | O | X | O | X | O | O |
| X | O | X | X | O | X | O | O |
| O | X | O | O | X | O | X | X |
| O | O | X | X | O | O | X | X |
| X | X | O | O | X | X | O | O |
| X | O | O | X | O | O | X | X |
| O | X | X | O | X | X | O | O |

-> Satisfy all rules of 3-in-row

1. Interesting Discussion
2. in sudoku

Before making program that solves sudoku, We thought how we have solved sudoku. Checking where is the number, write down tiny number in the blank. But this method is massy, so we tried to find new way to solve. This way use concept of Set, SubSet, Cartesia Product. First make two sets. One set contains rows that there is no number N, the other set contains contain columns that there is no number N. And calculate Cartesia Product A, B. So we can get new set about where number N can be being(need more rule). Next, delete some elements that other number is being already. And check number N is already being or not in sections(sudoku has 9 sections). If number N is already being, delete that element in same section. Else if only one element is being in section, write down number N in the element. But if there are two element in one section, check next number. Repeat this way until all sudoku is done. When we solve sudoku, we can remind concept of Set. Set is being in our life even we didn’t know that.

1. Deep Neural Network and SMT-solver

The way SMT-solver and Deep Neural Network solve problems is completely different. SMT-solver gives us whether an interpolation that meets a given boolean format exists and tells us the combination that is the solution to the problem. In Deep Learning, the Neural Network deals with data, finds patterns in the data, and find solution to problems. We have come up with opinions that this use of SMT-solver may compensate for the lack of theoretical support for Deep Learning and increase accuracy. To see if the two can join together, we looked at each of the problems that were being applied. Deep Learning focuses on complex and real-world courses such as automatic voice recognition, image recognition, and natural language processing. SMT-solver handles problems such as proving the correctness of programs, software testing based on symbolic execution that can be accessed by converting problems into logic format. We found that the two deal with a problem of opposite character. There was a case in which SMT-solver was applied when solving XOR in Image Recognition, but it was inefficient. It's hard to put guide in training process as when training like putting assert in the middle of a neural network. SMT-solver is difficult to use for training, but it can be useful for verifying Deep natural networks.