Programming Assignment 1

- Class : Discrete mathematics
- Professor : Hong Shin
- Team Num : 9
- Name : Chaeun Lim, Isaac Park, Gicheol Kim, Sukjin Kim, Jiho Choi

1. How each program models a puzzle as a logical formula

1.1 Sudoku

Brief description about Sudoku

A Sudoku puzzle has a 9 x 9 grid with 3 x 3 subgrids. Each subgrids fulfills the following requirements.

- ✓ Each cell has a number in 1 to 9
- ✓ Certain cells are assigned to certain values. These cannot be changed.

The puzzle is solved by assigning a proper number to each cell such that

- ✓ Every row contains each of 1 to 9
- ✓ Every column contains each of 1 to 9
- ✓ Every block contains each of 1 to 9

Transformation puzzle to code

To transformation puzzle to code, we need the following.

- ✓ Generating 9 x 9 grid
- ✓ Assign proper value in each cells so that can distinguish certain cells that already assigned values
- ✓ Making value of every column distinct
- ✓ Making value of every row distinct
- ✓ Making value of every subgrid distinct

Description about Code

```
//Generating 9x9 grid
for (y = 1 ; y <= 9 ; y++)
    for (x = 1 ; x <= 9 ; x++)
        fprintf(stream, "(declare-const a%d%d Int)\n", y, x) ;
```

These declares 81 constants of Int type which means 9 x 9 grid.

```
//Assign proper value
for (y = 1 ; y <= 9 ; y++)
    for (x = 1 ; x <= 9 ; x++){
        int mark = from_input[index ++];

        if(mark == 0) fprintf(stream, "(assert (and (<= 1 a%d%d) (<= a%d%d 9)))\n", y, x, y,
            x) ;
        else fprintf(stream, "(assert (= a%d%d %d))\n", y, x, mark);
    }
```

These receive values of the array that has input. If the value is not 0 which means that the value of this cell has already assigned, Z3 assigns the value. If the value is 0, Z3 assigns integer between 1 and 9.

```
 for (y = 1 ; y <= 9 ; y++){  //every column disticnt
     fprintf(stream, "(assert(distinct");
     for (x = 1 ; x <= 9 ; x++){
         fprintf(stream, " a%d%d",y, x);
     }
     fprintf(stream, "))\n");
 }

 for (x = 1 ; x <= 9 ; x++){    //every row distinct
     fprintf(stream, "(assert(distinct");
     for (y = 1 ; y <= 9 ; y++){
         fprintf(stream, " a%d%d",y, x);
     }
     fprintf(stream, "))\n");
 }
```

The cases that have duplicate integer in each row should not be permitted. Same applies to column.

```
fprintf(stream, "(assert(distinct");
for (y = 1 ; y <= 3; y++){//every square disticnt
    for (x = 1 ; x <= 3 ; x++)
        fprintf(stream, " a%d%d",y, x);
}  fprintf(stream, "))\n");

fprintf(stream, "(assert(distinct");
for (y = 1 ; y <= 3; y++){//every square disticnt
    for (x = 4 ; x <= 6 ; x++)
        fprintf(stream, " a%d%d",y, x);
}  fprintf(stream, "))\n");
```

The cases that have duplicate integer in every subgrid should not be permitted.

1.2 Kakurasu

Brief description about Kakurasu

kakurasu is a puzzle that fills up 8 x 8 grid with ‘X’ and ‘O’ by fulfilling the following requirements.

- ✓ At first, Cells should be marked with either 'O' or 'X'
- ✓ A puzzle gives an integer number for each row and each column. The number for a column should be the same as the sum of column indices of the cells marked with 'X' in the column
  - symmetrically, the number for a row means the sum of row indices of the 'X' cells in the row
- ✓ finds a 8x8 grid with 'O' and 'X' that solves the given puzzle

Transformation puzzle to code

To transform puzzle to code, we need the following.

- ✓ Generate 8 x 8 grid A grid and B grid
- ✓ Assign proper value in each cell, In case A assign the row (column index) sum instruction
- ✓ Assign proper value in each cell, In case B assign the Column (row index)sum instruction
- ✓ A and B grid cells can have only two numbers. make the condition to do select only two numbers then, Check the value that satisfy two grids at the same time. When you add two grids in A grid and B grid, you should get 0 or sum of each row and column.
- ✓ Select the one of grid A or B, and if the value is 0 put that cell 'O' else 'X'

Description about Code

```
//declare-const a and b
for (y = 1 ; y <= 8 ; y++)
    for (x = 1 ; x <= 8 ; x++)
        printf("(declare-const a%d%d Int)\n", y, x) ;

for (y = 1 ; y <= 8 ; y++)
    for (x = 1 ; x <= 8 ; x++)
        printf("(declare-const b%d%d Int)\n", y, x) ;
```

First. I made two 8x8 grid because sum of row and sum of clolumn are different according to the index. So I made A grid (sum of row) B grid(sum of column)

```
//(assert (or (= axy 0) (= axy (x)or(y))))

for (y = 1 ; y <= 8 ; y++)
    for (x = 1 ; x <= 8 ; x++)
        printf("(assert (or (= a%d%d %d) (= %d a%d%d)))\n", y, x,x,0,y,x) ;

for (y = 1 ; y <= 8 ; y++)
    for (x = 1 ; x <= 8 ; x++)
        printf("(assert (or (= b%d%d %d) (= %d b%d%d)))\n", y, x,y,0,y,x) ;
```

In A grid, each Row has a value that can be 0 or each column index value. I created a command that limits the value
In B grid, each Row has a value that can be 0 or each row index value, I created a command that limits the value

```
//assert row sum

for (y = 1 ; y <= 8 ; y++){
        printf("(assert (= (+") ;
    for (x = 1 ; x <= 8 ; x++)
        printf(" a%d%d ", y, x) ;
    printf(")%d))\n",b[y]);
}
//assert column sum

for (x = 1 ; x <= 8 ; x++){
    printf("(assert (= (+") ;
    for (y = 1 ; y <= 8 ; y++)
        printf(" b%d%d ", y, x) ;
    printf(")%d))\n",b[z++]);
}
```

Create a command that adds the values in each row of the grid to the values from input.txt when they are added In A grid.
Create a command that adds the values in each column of the grid to the values from input.txt when they are added In B grid.

```
//(assert (or (= (+ a11 b11) 0) (= (+ a11 b11) x+y))))

for (y = 1 ; y <= 8 ; y++)
    for (x = 1 ; x <= 8 ; x++)
        printf("(assert (or (= (+ a%d%d b%d%d) 0) (= (+ a%d%d b%d%d) %d)))\n", y, x,y,x,y,x,y,x,x+y) ;
```

Finally, we have created a command that finds values that satisfy two grids at the same time. When you add two grids in A grid and B grid, you should get 0 or sum of each row and column.

1.3 3-In-A-Row

Brief description about 3-In-A-Row

3-In-A-Row is a puzzle that fills up 8 x 8 grid with 'X' and 'O' by fulfilling the following requirements.

- ✓ At first, the initial arrangements partially filled are offered and these cannot be changed.
- ✓ In each row, the number of 'X' and the number of 'O' should be the same.
- ✓ In each column, the number of 'X' and the number of 'O' should be the same.
- ✓ Case that 3 consecutive cells in a row have all 'O' of all 'X' is not permitted.
- ✓ Case that 3 consecutive cells in a column have all 'O' of all 'X' is not permitted.

Transformation puzzle to code

To transform puzzle to code, we need the following.

- ✓ Generating 8 x 8 grid
- ✓ Assigning proper value in each cell so that can distinguish 'O' and 'X'
- ✓ Making the number of 'O' and the number of 'X' be the same for each row
- ✓ Making the number of 'O' and the number of 'X' be the same for each column
- ✓ Removing cases that have three consecutive 'O' or 'X' in a row
- ✓ Removing cases that have three consecutive 'O' or 'X' in a column

Description about Code

```
//Generating 8 X 8 grid
for (y = 1 ; y <= 8 ; y++)
  for (x = 1 ; x <= 8 ; x++)
   fprintf(stream, "(declare-const a%d%d Int)\n", y, x) ;
```

These declares 64 constants of Int type which mean 8 x 8 grid.

```
//Assigning values[0 or 1 for ?(denoted by 0), 1 for O(denoted by 1), 0 for X(denoted by 2)]
for (y = 1 ; y <= 8 ; y++)
  for (x = 1 ; x <= 8 ; x++){
    int mark = from_input[index++];

    if(mark == 0) fprintf(stream, "(assert (and (<= a%d%d 1) (<= 0 a%d%d)))\n", y, x, y, x) ;
    //if(mark == 1) fprintf(stream, "(assert (= a%d%d 1))\n", y, x, y, x) ;
    if(mark == 1) fprintf(stream, "(assert (= a%d%d 1))\n", y, x) ;
    //if(mark == 2) fprintf(stream, "(assert (= a%d%d 0))\n", y, x, y, x) ;
    if(mark == 2) fprintf(stream, "(assert (= a%d%d 0))\n", y, x) ;
  }
```

These receive values of the array that has input. If the value is 0 which denoted '?', Z3 assigns 0 or 1. If the value is 1 which denoted 'O', Z3 assigns 1. If the value is 2 which denoted 'X', Z3 assigns 0.

```
//Fomula that makes # of O and # of X same for each row
for (x = 1 ; x <= 8 ; x++){
  fprintf(stream, "(assert(= (+ ") ;
    for (y = 1 ; y <= 8 ; y++)
    fprintf(stream, "a%d%d ", y, x) ;
  fprintf(stream, ") 4))\n") ;
}
```

In this case, each 'O' and 'X' is 1 and 0. If the number of 'O' and the number of 'X' is same in a row which has 8 cells, sum of the cells in a row must be 4. The same applies to column.

```
//Formula that removes cases that have three consecutive 'O' or 'X' in a row
for(y = 1; y <= 6;y ++){
  for (x = 1 ; x <= 8 ; x++){
    fprintf(stream, "(assert (and (< (+ ");
      for (i = 0 ; i < 3 ; i++){
        fprintf(stream, "a%d%d ", y + i, x) ;
      }
    fprintf(stream, ") 3) ") ;
    fprintf(stream, "(> (+ ");
      for (i = 0 ; i < 3 ; i++){
          fprintf(stream, "a%d%d ", y + i, x) ;
        }
    fprintf(stream, ") 0)))\n") ;
  }
}
```

In this case, each ‘O’ and ‘X’ is 1 and 0. In the case of having three consecutive ‘O’ or ‘X’ in a row, sum of three adjacent cells must be 3 or 1. These avoid those cases. Same applies to column.

2. Program Designs

Design

There are 4 steps in the program design. First, the program will be given input.txt file by argument. Then module ‘Load’ will encode it to store in integer array due to logical use. Second, module ‘Construct’ takes the array data and constructs ‘formula’ for extracting satisfiable model into Z3 language. Third, after formula is completed, module ‘Solver’ takes the formula through file, compute it, if the puzzle is satisfiable, then it produces ‘result’ as a text file. Lastly, module ‘Convert’ decodes the result into words on grid layer which is familiar to user.

For additional function, there are module ‘Output’ and ‘Test’, module ‘Output’ is saving the result from module ‘Convert’ as a file. Module ‘Test’ is printing values in array (input data or solution data) on an interpreter.

Load

void LoadData(int* data)

Module ‘LoadData’ is the first module that works when this program starts. When user runs this program with argument for ‘input.txt’, then this module converts data into integer array. For using the data as a logical value, every character must be translated into an integer. For Sudoku puzzle, ‘input.txt’ has some integer values from 0 to 9 and also has ‘?’. 0 and ‘?’ are same sign for ‘unknown’. So LoadData function will encode both data as a 0.

Construct

void ConstructSolution(int* data)

The second is module ‘Construct’. This module is the most important core module in this program. The module makes formula with data array. The formula will be constructed into Z3 SMT language. Then it saves the formula as a ‘formula.txt’

Solver

void Z3Solver(int* solution)

Third, module ‘Solver’ calls Z3 program to solve the puzzle with a formula which is generated by ‘Construct.’ The module will throw ‘formula.txt’ file to Z3 program as an argument, and order Z3 to create result as a file in a local folder.

Convert

void ConvertData(int* solution)

The last module ‘Convert’ is converting result produced by module ‘Solver’ into a familiar style for the user. The result from Z3 has some pattern that indicates what each cell has. So that the module reconstructs the logical result into words.

Output

void MakeOutput(int* solution)

Output module works for making ‘output.txt’ file with converted solutions. When module ‘Convert’ is done with decoding the data into words, then this module stores it in a local folder. The data in the file is constructed in grid style.

Test

void TestPrint(int* data)

void TestOutputKaku(int* data)
void TestOutput(int* solution)

Test module works for showing what the data or solution has on user interpreter. TestPrint() function shows the data linearly, and TestOutputKaku() and TestOutput() shows the data on a grid.

3. How to build and execute programs (i.e. manual)

You can get source codes through this site. (https://github.com/PASTANERD/DM_)
There are 3 folders for each puzzle (sudoku, kakurasu, and 3-in-a-row). Each folder has program source code and 'input.txt' for puzzle example.

Process for executing program
Every instruction for executing this program is based on OSX and GCC.

1. Running Terminal.
2. Go to local folder where source code is located.
3. Compile source code. (gcc sudoku_solver.c) (or you can change the name by putting -o)
4. Run exec program (./a.out input.txt) (or ./(program) ---(program) is the name of program you make)
5. If the process is ended, then you can see formula, result and output file

Additional instruction about product files

- formula.txt file is another source code for Z3
- result.txt file is result of the puzzle in Z3 language
- output.txt file is converted

Constraints

You must check the ingredients of input.txt are proper to puzzle.
This program is verified in below conditions

OSX: Target: x86_64-apple-darwin17.4.0
GCC: 4.2.1
Apple LLVM version 9.0.0 (clang-900.0.39.2)

OSX: Target: x86_64-apple-darwin17.7.0
GCC: 4.2.1
Apple LLVM version 10.0.0 (clang-1000.11.45.2)

This program is programed on OSX. We can't assure that this program will work on another OS like Windows or Linux. Also, we didn't test on any compilers but GCC. Please take this into consideration when testing the programs.

4. Demonstration that each of program works correctly

Figure 1. Demonstration of Sudoku_solver Program

Figure 2. Demonstration of Kakurasu_solver Program

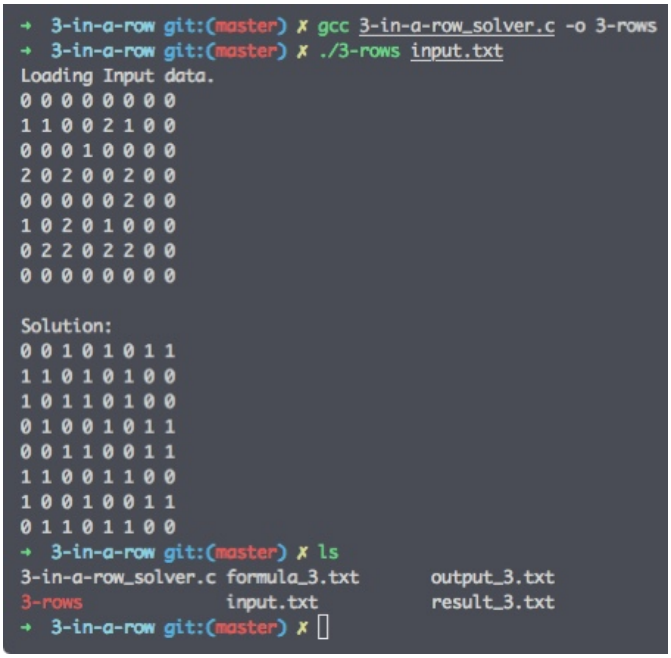

Figure 3. Demonstration of 3-In-A-Row_solver Program