ITP20002-01 Discrete Mathematics

# Programming Assignment 1

28 Sep 2018

# Puzzle Solvers Using SMT-solver

- Construct three programs that automatically find solutions of the following three kinds of puzzles, respectively

  1. Sudoku
  2. Kakurasu
  3. 3-In-A-Row

- Settings
  - Due date: **11:59 PM, 8 Oct (Mon)**
  - Teamwork with your first team members
  - Evaluation criteria
    - Report: 60%
    - Program test: 40%

# Using SMT-solver Z3

- Software repository: https://github.com/Z3Prover/z3
- Tutorial: https://rise4fun.com/z3/tutorial
  - Ex. propositional logic $(p \rightarrow q) \wedge (q \rightarrow \neg p) \wedge \neg(p \vee q)$

```
; prop.txt
(declare-const p Bool)
(declare-const q Bool)
(assert (and (=> p q)
             (=> q (not p))
             (not (or p q))))
(check-sat)
(get-model)
```

```
$ z3 prop.txt
sat
(model
  (define-fun q () Bool
    false)
  (define-fun p () Bool
    false)
)
```

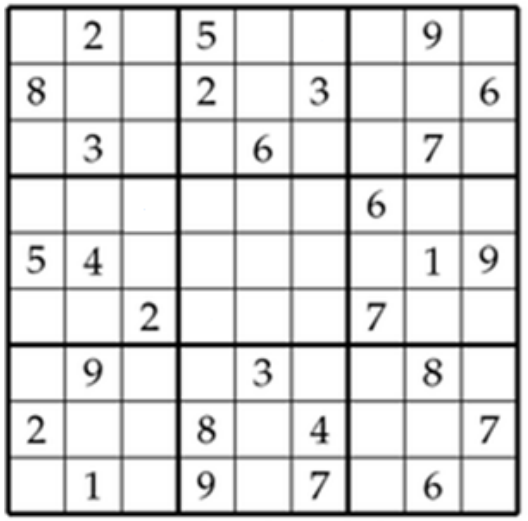
  - Ex. Linear-integer arithmetic $\exists a \exists b \exists c\ (a > b + 2\ \wedge\ a = 2c + 10\ \wedge\ b + c \leq 1000)$

```
; lia.txt
(declare-const a Int)
(declare-const b Int)
(declare-const c Int)
(assert (> a (+ b 2)))
(assert (= a (+ (* 2 c) 10)))
(assert (<= (+ c b) 1000))
(check-sat)
(get-model)
```

```
$ z3 lia.txt
sat
(model
  (define-fun b () Int
    0)
  (define-fun c () Int
    0)
  (define-fun a () Int
    10)
)
```

# Puzzle 1. Sudoku

- A Sudoku puzzle has a 9 x 9 grid with nine 3 x 3 subgrids (i.e., blocks)
  - each cell has a number in 1 to 9
  - certain cells are assigned to certain values
- The puzzle is solved by assigning a 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



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

- Example

input.txt

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

Output

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



- Hint: `(assert (distinct ..))` of Z3

# Puzzle 2. Kakurasu

- There is a 8x8 grid each of whose cell 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 indicies 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
- Write a program that finds a 8x8 grid with 'O' and 'X' that solves the given puzzle
- Example

input.txt

```
14 18 30 33 29 24 6 1
13 15 9 15 18 27 19 12
```

Solution

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | o | x | o | o | x | o | x | o | 14 |
| 2 | o | o | x | x | x | x | o | o | 18 |
| 3 | o | x | x | x | o | x | x | x | 30 |
| 4 | o | o | x | x | x | x | x | x | 33 |
| 5 | x | x | o | o | x | x | x | x | 29 |
| 6 | o | x | o | x | x | x | x | o | 24 |
| 7 | o | o | o | o | o | x | o | o | 6 |
| 8 | x | o | o | o | o | o | o | o | 1 |
| | 13 | 15 | 9 | 15 | 18 | 27 | 19 | 12 | |

# Puzzle 3. 3-In-A-Row

- Fill up a 8x8 grid with 'X' and 'O' to fulfill the following requirements
  - As input, the initial arrangements are given 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
  - There is no case that three consecutive cells in a row have all 'O' or all 'X'
  - There is no case that three consecutive cells in a column have all 'O' or all 'X'
- Write a program that finds a such 8x8 grid filled with 'X' and 'O'
- Example

input.txt

```
? ? ? ? ? ? ? ?
o o ? ? x o ? ?
? ? ? o ? ? ? ?
x ? x ? ? x ? ?
? ? ? ? ? x ? ?
o ? x ? o ? ? ?
? x x ? x x ? ?
? ? ? ? ? ? ? ?
```

Solution

```
x x o x o x o o
o o x o x o x x
o x o o x o x x
x o x x o x o o
x x o o x x o o
o o x x o o x x
o x x o x x o o
x o o x o o x x
```

# Program Structure

- A puzzle solver should have three components:
  A. a module that reads an input from `input.txt` and then generates a corresponding formula as `formula.txt`
  B. a module that runs Z3 to solve `formula.txt` and then receives the result
  C. a module that interprets the Z3 output and print out the solution to `output.txt`

- For each puzzle, implement the whole procedure as one executable

# Submission

- Each team should submit 3 program source-code files and a report on the program designs and results
  - Source code: either Github (recommended) or Hisnet
  - Report: Hisnet

- Report
  - the report must contain the followings:
    - description on how each program models a puzzle as a logical formula
    - description on the program designs
    - instruction on how to build and execute your programs (i.e., manual)
    - demonstration that each of your program works correctly
  - there will be extra points if the report contains interesting discussions
  - the report must not exceed 6 pages (single-sided A4)