Discrete Mathematics

Programming Assignment 1
Solve Puzzles with SMT Solver

Shin Hong

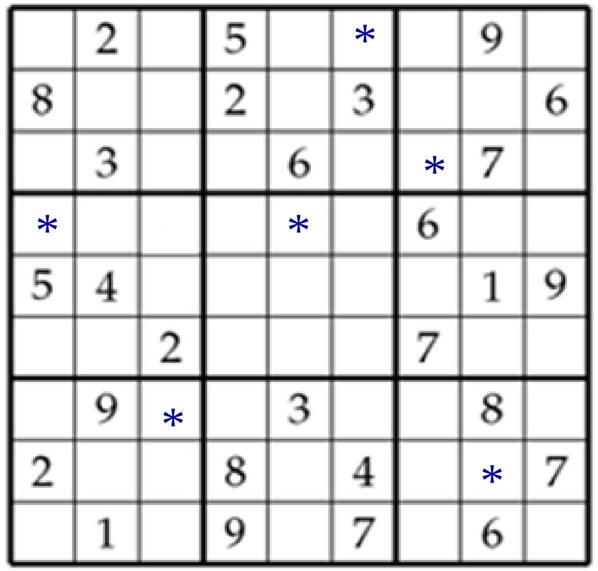
23 Sep, 2019

Assignment Overview

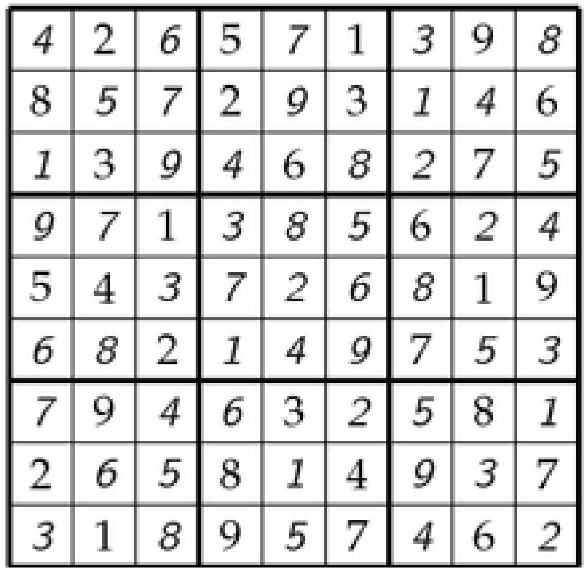
- Task: write a program for each of the following three puzzles, that automatically finds solutions using the Z3 SMT solver
    1. Sudoku*
    2. Fill-a-Pix
    3. Numbrix
- Submission (see more details at Slides 9-12)
    - deadline: 4 Oct (Fri), 11:59 PM
    - deliverables
        - three programs (one for each puzzle)
        - write up
- Team: work with your team members to make one submission

Puzzle 1. Sudoku* (1/2)

- The original 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 with one value in 1 to 9
- In Sudoku* (a variant of Sudoku), certain cells are marked with the asterisk sign (*)
  - at most 9 cells are marked the asterisk sign
- A Sudoku* 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
  - no two cells marked with Asterisk have a same number



	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	



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

Puzzle 1. Sudoku* (2/2)

- Requirement
  - Your program must use the Quantifier-free LIA logic to model this game (not propositional logic)
- Input
  - read input from the standard input
  - each line has initial settings of the 9 cells of a row
    - `?` : no specific number is assigned
    - `1`..`9` : one specific number is assigned
    - `*` : the cell is marked as Asterisk
- Output
  - print out the complete 9x9 grid to the standard output
  - or, print "No solution" if there's no solution
- Hint
  - check a Z3 primitive `(distinct ..)`

```
? 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 ?
```

<Input example>

```
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
```

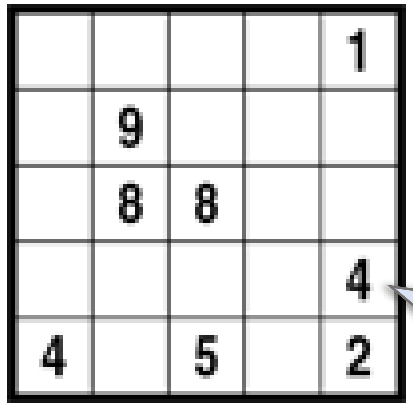
<Output example>

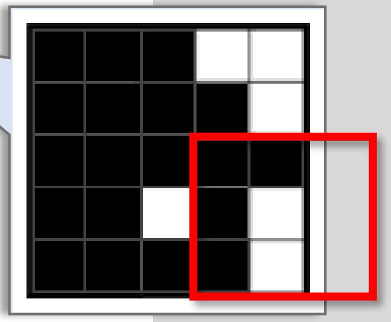
Puzzle 2. Fill-a-Pix (1/2)

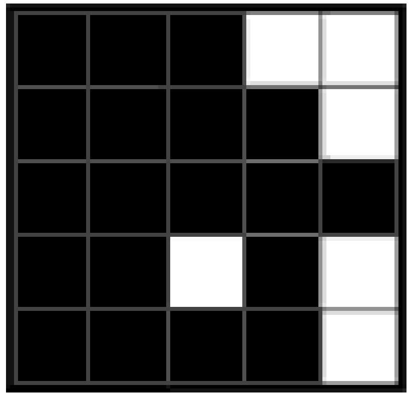
- Fill-a-Pix is to figure out whether the color of each cell of a N×M grid is White or Black, based on given clues

- Initially, the colors of all cells are unknown, and clues are placed on certain cells
  - a clue on a cell is a number between 0 and 9
  - a clue indicates the number of Black cells in the surrounding 8 cells and the cell where the clue is on

- A solution assigns each cell as Black or White
  - a game may have no solution, single solution, or multiple solutions







Puzzle 2. Fill-a-Pix (2/2)

- Requirement
  - Your program must use the Quantifier-free LIA logic to model this game (not propositional logic)
- Input
  - read input from the standard input
    - an input will be not larger than 1000x1000
  - each line has initial settings of the cells of a row
    - **`?`** : no clue is given
    - **`1`**..**`9`** : a clue is given
- Output
  - print out the colorings of the grid to the standard output
    - **`1`** : Black
    - **`0`** : White
  - print out "No solution" if there is no solution
  - if there are multiple solutions, print them up to 5

```
? ? ? ? ?
? 9 ? ? ?
? 8 8 ? ?
? ? ? ? 4
4 ? 5 ? 2
```

<Input example>

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

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


. . .
```

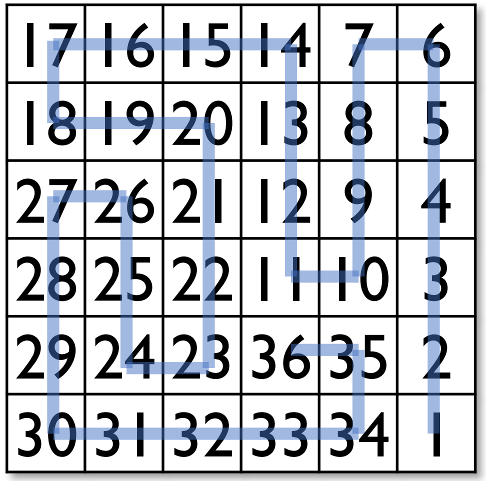
for there are more solutions

<Output example>

Puzzle 3. Numbrix (1/2)

- A Numbrix puzzle consists of a N×M grid where some numbers between 1 and N×M are placed on some cells
- The goal of a game is to place the remaining numbers between 1 and N×M on the grid such that two numbers x and $x + 1$ are placed at vertically or horizontally adjacent cells always ($1 \leq x <$ N×M)
  - as a result, a sequence of 1 to N×M spans whole grid by moving veritically and horizontally

		20	13		
	26			9	
	25			10	
		23	36		



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

Puzzle 3. Numbrix (2/2)

- Requirement
  - Your program must use the Quantifier-free LIA logic to model this game (not propositional logic)
- Input
  - read input from the standad input
  - each line has initial settings of the cells of a row
    - **?** means that no value is yet assigned
  - the given grid will be not larger than 100 x 100
- Output
  - print out the completed grid to the standard output
  - print out "No solution" if there is no solution

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

<Input example>

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

<Output example>

Program Structure

- Write each program as a C program running on UNIX/LINUX
  - Each program receives input from the standard input and produces the ouput to the standard output
  - Tests will be conducted on Peace
- Each program (per puzzle) must be built as a single executable
  - Programs can execute Z3 in a middle of execution through `popen` (see an example of `nqueen-LIA.c`)
- You must submit build scripts and README together with source code files
  - buid script: Bash script, Makefile, Ant, Maven, etc.
  - REAME: instruction/manual on how to build and run your program

Submission

- Deadline: 4 Oct (Fri), 11:59 PM
  - no late submission will be accepted
  - one submission per team

- Each team should submit 3 programs and one write-up (report) on the program designs and results
  - Programs: source code files, build script and README
  - Write up: must not exceed 6 pages (single-sided A4)

- Submit all deliverables via Hisnet homework submission repository

Evalution Criteria

- Write up (60 points)
  - Description (45 points)
    - check whether you found all constraints of a solution
    - check whether each constraint is correctly represented as a logic formula
    - check whether you demonstrate the correctness of your programs in a convincing way (e.g., by tests)
    - check whether all descriptions are clear and consistent
  - Discussion (15 points)
    - detailed analysis of results, interesting observations, lessons learned, suggestions, new ideas, etc.

- Tests (40 points)
  - Run each program with several inputs to see whether the results are correct

Notes

- Right after the deadline, an individual homework related to PA 1 will be given
  - HW1 will be on the same line of PA 1
  - You will have 5 days
- Right after the deadline, peer evaluation of your team members will follow
- Your submissions may be open to the class and public
- You can request to get a Peace account at http://peace.handong.edu:8000/register
- If you have a question, write a post in Piazza; TAs and I will not answer to any email regarding PA 1.
- Help desk by TAs
  1. Sep 25 (Wed), 8-9 PM @ Coding Space
  2. Sep 30 (Mon), 8-9 PM @ Coding Space
  3. Oct 2 (Wed), 8-9 PM @ Coding Space