ITP20002-02 Discrete Mathematics

# Programming Assignment 1

29 Sep 2017

# PA1. SAT-based Sudoku Puzzle Solver

- Task: construct a program that automatically finds a solution of a given 9 x 9 Sudoku puzzle
  1. represent the problem as a propositional formula
  2. use an off-the-shelf SAT solver to solve the formula
  3. convert the SAT solver result to a Sudoku solution

- Setting
  - Due date: **11:59PM, 12 Oct (Thu)**
  - Teamwork: 3 persons per team
  - Evaluation
    - report: 40%
    - test: 30%
    - online exam: 30% (schedule: TBA)

# Using Off-The-Shelf SAT Solvers

- SAT solvers (constraint solvers)
  - Z3 (MS Windows) https://z3.codeplex.com/releases
  - MiniSAT http://minisat.se/MiniSat.html

- Input: DIMACS format: a structural representation of a CNF formula
  - E.g., solve $(x \vee y) \wedge (\neg z) \wedge (\neg x \vee \neg y)$ using Z3
    - Input file

```
p cnf 3 3
1 2 0
-3 0
-1 -2 0
```

to use numbers 1 to 4 as propositional variables

to give 3 clauses

    - Operation

```
$ z3 –dimacs formula.txt
sat
1 -2 -3
```

# Sudoku Puzzle

- 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 so that every row, every column, and every of a block contains each of the nine numbers.
- Modeling
  - Proposition $p(i, j, n)$ is true if and only if the cell at row $i$ and column $j$ has a value $n$

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

$$\bigwedge_{i=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{j=1}^{9} p(i, j, n)$$

$$\bigwedge_{j=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{i=1}^{9} p(i, j, n)$$

$$\bigwedge_{r=0}^{2}\bigwedge_{s=0}^{2}\bigwedge_{n=1}^{9}\bigvee_{i=1}^{3}\bigvee_{j=1}^{3} p(3r+i, 3s+j, n)$$

$$\bigwedge_{i=1}^{9}\bigwedge_{j=1}^{9}\bigwedge_{n=1}^{8}\bigwedge_{m=n+1}^{9} \big(p(i,j,n) \rightarrow \neg p(i,j,m)\big) \wedge \big(p(i,j,m) \rightarrow \neg p(i,j,m)\big)$$

# Extended Sudoku Puzzle

- In an extended Sudoku puzzle, some cells may be marked with Asterias (i.e., *) to assert that these must be assigned with the same number in the solution
  - at most 4 Asterias can appear in a problem

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 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 | |

Unsolved Sudoku

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 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 |

Solved Sudoku

# Hint - 3x3 Sudoku Puzzle

| | | |
|---|---|---|
| 1 | | 2 |
| | | 1 |
| | 1 | |

| | | |
|---|---|---|
| 1 | 3 | 2 |
| 3 | 2 | 1 |
| 2 | 1 | 3 |

- Setting
  - Let a natural number *X* represent a proposition $p_X$
  - $p_X$ is true iff the cell at the ( (X-1) / 9 + 1)-th row and the ( ((X-1)/3)%3 + 1)-th column has value ( (X-1) % 3 + 1)
- Constraint encoding
  - Every row must contain each number at least once
  - DIMACS

$$\bigwedge_{i=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{j=1}^{9} p(i,j,n)$$

```
1 4 7 0          # i=1,n=1,j={1,2,3}
2 5 8 0          # i=1,n=2,j={1,2,3}
3 6 9 0          # i=1,n=3,j={1,2,3}
10 13 16 0       # i=2,n=1,j={1,2,3}
11 14 17 0       # i=2,n=2,j={1,2,3}
12 15 18 0       # i=2,n=3,j={1,2,3}
……
```

# Input

- Given as text file `input.txt`
- The input file has 9 lines each of which contains 9 words
- Each word is either zero (unassigned), or one of 1 to 9 (assigned), or Asterias (marked as star).
- Example. `input.txt`

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

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 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

- Print out the solution to `output.txt` if there is any solution
- Print out "`no solution`" if there is no solution.
- Example. `output.txt`

```
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
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 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 |

# Program Structure

- The Sudoku solver should have three components:
  A. a module that reads an input from `input.txt` and then generates a DIMACS formula file `formula.txt` as output
  B. a module that executes a SAT solver to solve `formula.txt` and then receives the SAT solver output
  C. a module that interprets the SAT solver results and print out the solution to `output.txt`

- Implement the whole procedure as one executable file
  - For an inevitable case, you can implement the modules A and C as separate programs. This implementation should have a penalty in evaluation.

# Submission

- Each team should submit both program source code and a report on the program design and result
  - Source code: either Github (recommended) or Hisnet
  - Report: Hisnet

- Report
  - the report must contain the followings:
    - description on the program design
    - instruction on how to build and execute your programs (i.e., manual)
    - demonstration that shows your program works properly
  - there will be extra points if the report contains interesting discussions
  - the report must not exceed 5 pages (single-sided, A4)

# Team

- Claim:
  send me an email by tonight

| | | | | |
|---|---|---|---|---|
| Team1 | 김정환 | 박천명 | 최호윤 | |
| Team2 | 고은성 | 김빛나 | 한현수 | |
| Team3 | 김명철 | 최시령 | 김세인 | |
| Team4 | 서지수 | 정겨운 | 김영표 | |
| Team5 | 이기혁 | 허규진 | 김종성 | |
| Team6 | 윤성결 | 이주혜 | 김윤정 | |
| Team7 | 정재훈 | 이용호 | 박유경 | |
| Team8 | 정명윤 | 김도현 | 김준영 | |
| Team9 | 장준용 | 장태양 | 윤지영 | |
| Team10 | 김시민 | 이정은 | 정현섭 | |
| Team11 | 이광현 | 안동민 | 김소은 | |
| Team12 | 최선웅 | 김진향 | 한현민 | |
| Team13 | 오승준 | 전영민 | 이준섭 | |
| Team14 | 최유진 | 김성민 | 이주영 | |
| Team15 | 김형준 | 김아현 | 하재경 | |
| Team16 | 신현웅 | 한보경 | 김효림 | |
| Team17 | 박희성 | 강준민 | 김하은 | |
| Team18 | 김지웅 | 김현지 | 이충현 | |
| Team19 | 서성혁 | 하민지 | 오재영 | 한정섭 |