ITP 20002 Discrete Mathematics, Fall 2020

Homework 1. Solving Logic Puzzles Using SAT Solver

Shin Hong

1. Overview

In this homework, for two grid-based puzzles *Sudoku-X* and *Easy-As-ABC*, you are asked to construct two C programs each of which transforms a given puzzle instance into a corresponding propositional logic formula and finds a solution by solving the formula with a SAT solver Z3[1].

Sudoku-X, a variant of the standard Sudoku puzzle, is to fill out each cell of a $N \times N$ grid such that every row, every column, every subgrid and each of the two diagonals has all N numbers (see Section 2.1). Easy-AS-ABC is another grid-based puzzle where a player is asked to fill some of empty cells with one of five letters 'A', 'B', 'C', 'D', and 'E' such that each of the five letters appears exactly once in every row and column, while using the first/last letter of a column/row as specified in a puzzle (see Section 2.2).

You must construct two programs, one for Sudoku-X and the other for Easy-As-ABC. Given puzzle instance input, your program must generate a corresponding propositional formula following the Z3 input grammar, and then pass this generated formula to Z3, and finally translate the Z3 result into a puzzle solution.

The submission deadline is **11:00 PM, 30 September** (Wed). You must submit reports together with programming results.

2. Problem Puzzles: Sudoku-X and Easy-As-ABC

2.1. Sudoku-X

Like its original version, Sudoku-X has a 9×9 grid which consists of nine 3×3 subgrids. Each cell can be assigned with one integer between one and nine such that each of the nine integers must appear exactly once at every row, every column and every subgrid. In addition, Sudoku-X requires that the nine cells of each diagonal of the 9×9 grid (i.e., one from top-left to bottom-right and one from top-right to bottom-left) must contain all nine integers.

A puzzle instance is given as a 9×9 grid with some pre-assigned cells. A player must assign an integer value to every unassigned cell such that the grid satisfies all constraints of Sudoku-X. Figure 1 is an example of a Sudoku-X puzzle instance (left) and its solution (right) [2]. You can find that each of the two diagonals (marked with shadow) are assigned to contain all nine integers.

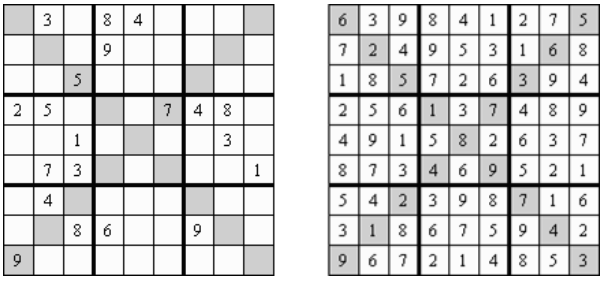

Figure 1. Sudoku-X Example

2.2. Easy-As-ABC

Easy-As-ABC consists of a $n \times n$ grid with letters annotated at the beginning/end points of some rows and columns. Each cell in a grid may or may not be assigned with one of five letters 'A', 'B', 'C', 'D', and 'E'. A player is asked to assign letters to some of the cells such that the five letters appear exactly once at every row and every column, and all constraints of given letter annotations are satisfied. An annotation given at the beginning point of a row (or column) constraints that the first non-empty cell in that row (or column) must contain the annotated letter. Similarly, an annotation given at the end of a row (or column) asserts that the last non-empty cell in that row (or column) must be the annotated letter. Figure 2 shows an example of a Easy-As-ABC puzzle instance (left) and one of its solution (right)[3]. You can find that every letter annotated in a row or column is the closest letter in the corresponding row or column.

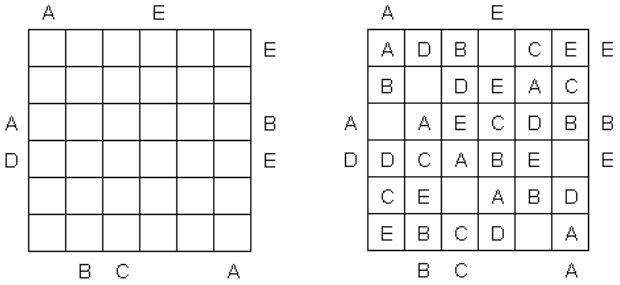

Figure 2. Easy-As-ABC Example

3. Tasks

You are asked to model a puzzle instance as a propositional logic formula such that an assignment that satisfies the formula represents a solution of the puzzle instance (the formula should be unsatisfiable if a puzzle instance has no solution). Based on this modelling, you are asked to construct a program that uses a SAT solver Z3 to find a solution of a puzzle instance given as input. More specifically, for each of Sudoku-X and Easy-As-ABC, you need to accomplish the following tasks:

- Step 1. devise a scheme to construct the corresponding propositional logic formula for a given puzzle instance
- Step 2. construct a puzzle solving program. The program must have the following components: (a) formula constructor, and (b) Z3 output interpreter
- Step 3. validate your program
- Step 4. write a report

The following subsections give details of these four steps.

3.1. Step 1. Puzzle modeling.

You must devise a scheme for constructing a propositional logic formula from a given puzzle instance. Your scheme must define what kinds of propositional variables should be introduced for modelling a puzzle instance and how to translate all posed constraints into propositional formula over these propositional variables. You must describe this scheme in a report clearly and systematically. You must show how the satisfiability of a translated formula is related with a puzzle solution.

[1] The latest version of Z3 can be found at https://github.com/Z3Prover/z3

[2] This example is taken from http://cross-plus-a.com/sudoku.htm

[3] This example is taken from http://cross-plus-a.com/puzzles.htm

3.2. Step 2. Program construction.

Based on your puzzle modeling, you should construct a C program that returns a solution for a received puzzle instance. Your program must have the following two components:

- Formula constructor
  Given puzzle instance, this module represents the corresponding constraints as a Z3 input string. The resulting string should be passed to Z3 by the main driver.
- Z3 output interpreter
  Given propositional constraints for a puzzle instance, Z3 returns a solution (either a satisfying assignment or a determination that the formula is unsatisfiable) as a formatted text. This module translates this Z3 output into a puzzle solution.

Your program must receive a puzzle instance input via the standard input (see more details in Section 4) and then passes the given puzzle instance to the formula constructor module. Once the Z3 input string is generated, the main driver executes Z3 and then passes the Z3 output to the Z3 output interpreter. Finally, the puzzle solution must be printed out to the standard output.

3.3. Step 3. Validation.

You must present in your report the process and the result of how you have confirmed that your program is working correctly. You must show how you test your programs with different input cases.

3.4. Step 4. Report writing

You must write two reports, one on your solution of Sudoku-X and the other on your solution of Easy-As-ABC. Your writing must include the following aspects of your solution in detail:

(1) your idea and details on modeling a puzzle with a propositional logic
(2) structure of your puzzle-solving program, and challenges in implementation
(3) validation results
(4) open discussion, for instances, lessons learned, observations, further investigation, suggestions of possible extension or improvements.

Evaluation will be primary based on your report (not your implementation), thus, you must try best to deliver all your results with best presentations in the reports.

4. Requirements

4.1. Input format

An input of Sudoku-X consists of nine lines each of which has nine values separated by one or multiple whitespaces. A value can be one of the nine integers (from '1' to '9') representing preassigned values and '?' indicating an unassigned cell. The following is an input text for the puzzle instance of Figure 1:

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

An input of Easy-As-ABC consists of 5 lines of text. The first line contains an integer representing the grid size (i.e., n). This value should be one between 5 and 10. The second line has n values indicating annotations given to the beginning (top) of the columns. A value is one of the five letters ('A' to 'E') or '_' indicating empty annotation. Values are separated by one or multiple whitespaces in a line. The third line contains the annotations given to the end (bottom) of columns. The fourth and the fifth lines contain the annotations for the beginning (left) and the end (right) of rows respectively, where the first value corresponds to the first row (top) and the last value to the last row (bottom). The following text represents the input for Figure 2:

```
6
A _ _ E _ _
_ B C _ _ A
_ _ A D _ _
E _ B E _ _
```

You can assume that always a valid input is given to your program.

4.2. Program

A puzzle solving program must be written as a single C source code file. Write a README if your program requires specific compilation and/or linking options or other non-trivial requirements. Your program must be compatible with GCC 5.4.0 or a higher version. As your program will be tested on the peace server[4], it is recommended to check if your program is working well on it.

Your program must receive input via the standard input and print the output to the standard output. A solution must be printed out in a human-readable form. Your program must print that there is no solution if it is so.

4.3. Report

You must write two reports, one for Sudoku-X and the other for Easy-As-ABC. You must use a given report template. Each of your reports must not exceed 2 pages in the given template. All contents in reports must be written in English (none of your Korean sentences will be counted in evaluation). A report must be submitted as a PDF file.

5. Submission

You should submit an archive (e.g., tar, zip) of your results to Hisnet by 11:00 PM, 30 September (Wed). The submission deadline is strict, and no late submission will be accepted. Your submission must include (1) a PDF file of report, and (2) a C source code file for each of the two puzzle problems.

[4] peace.handong.edu