University of Royal Holloway CS5960-2020

Prof. Sara Bernardini, Dr. Santiago Franco, Dr. Ali Akbari, Dr. Ferdian Jovan

Lab 4.

Solutions due Tuesday, **Dec 8, 23:59** uploaded to Moodle. See submission instructions below.

This exercise sheet is accompanied with some source files, which we provide through a separate `zip` archive. You can download this archive from the Moodle link for this lab.

Exercise 5: CSP Modeling in Z3. ()

This sheet is about modeling a Sudoku puzzle (see below) as CSP, and solving it using a CSP solver. To do so, we will be using the Z3 theorem prover.[1][2] To run Z3, you have to create a text file in Z3's input language, which you can then pass as command line argument to the Z3 executable. The Z3 binary is available in https://github.com/Z3Prover/z3/releases/download/z3-4.8.7/z3-4.8.7-x64-win.zip. Once downloaded and extracted to your folder of choice (Y drive for the lab) it can be run by typing `z3 filename` (as long as you are in the right folder). Printing additional information about the solving process can be done through adding the "-st" option, e.g., `z3 -st path/to/csp.z3`. The result of Z3 (`sat` or `unsat`) will be printed to the console.
An overview of the, for this sheet, relevant Z3 language fragments is shown in Table 1. As the input language of this solver allows to represent strictly more general problems than CSPs, **you must not use any statement that is not shown in this table**. An example file (the Coloring Australia example from the lecture) is available in the zip file.

Encode the following generalized Sudoku puzzle as CSP using the Z3 language fragment depicted in Table 1. The goal of this puzzle is to fill the empty cells in the board with numbers from 1 to 9 complying with the constraints listed below. We use $\langle i, j\rangle$ to denote the value of the cell with $x = i$ and $y = j$.

1. Typical Sudoku constraints:
   (a) Numbers cannot be repeated in any row, column, or 3x3 square
   (b) Cells whose values are already specified must be assigned to the respective values.

[1] `https://github.com/Z3Prover/z3`

[2] An introduction can be found in `http://rise4fun.com/z3/tutorial/guide`.

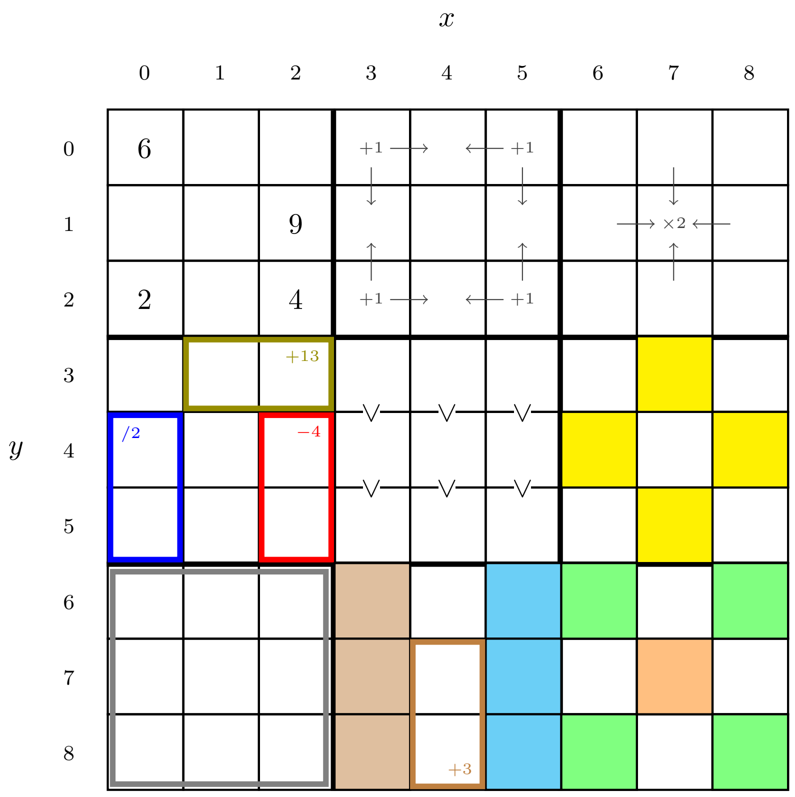

Figure 1: Illustration of the generalized Sudoku puzzle.

2. Top middle 3x3 square: For every corner cell of this square, one of the horizontally or vertically adjacent cells must equal the value plus 1. For example, for the cell $\langle 3, 0\rangle$, if $\langle 3, 0\rangle = 4$, then either $\langle 3, 1\rangle = 5$ or $\langle 4, 0\rangle = 5$.

3. Top right 3x3 square: The sum of the indicated cells must be equal to twice the value of the center cell. In other words: $\langle 7, 0\rangle + \langle 6, 1\rangle + \langle 7, 2\rangle + \langle 8, 1\rangle = 2 \times \langle 7, 1\rangle$

4. Middle left square: The numbers must comply with the arithmetic expressions drawn in the figure:

   (a) $\langle 1, 3\rangle + \langle 2, 3\rangle = 13$

   (b) $\langle 0, 5\rangle / \langle 0, 4\rangle = 2$

   (c) $\langle 2, 5\rangle - \langle 2, 4\rangle = 4$

5. Center square: Numbers must comply with the inequalities. More specifically:

   (a) $\langle 3, 3\rangle > \langle 3, 4\rangle > \langle 3, 5\rangle$

   (b) $\langle 4,3\rangle > \langle 4,4\rangle > \langle 4,5\rangle$
   (c) $\langle 5,3\rangle > \langle 5,4\rangle > \langle 5,5\rangle$

6. Middle right square: Exactly one of the yellow cells must contain a value smaller than 5, the other yellow cells must contain a value greater or equal to 5.

7. Bottom left square: The sum of all rows in this square must be equal, i.e., $\langle 0,6\rangle + \langle 1,6\rangle + \langle 2,6\rangle = \langle 0,7\rangle + \langle 1,7\rangle + \langle 2,7\rangle = \langle 0,8\rangle + \langle 1,8\rangle + \langle 2,8\rangle$.

8. Bottom middle square:

   (a) Multiplying the sums of the two indicated columns gives an odd number: $(\langle 3,6\rangle + \langle 3,7\rangle + \langle 3,8\rangle) \times (\langle 5,6\rangle + \langle 5,7\rangle + \langle 5,8\rangle)$ must be odd.

   (b) The numbers must comply with the arithmetic expressions drawn in the figure: $\langle 4,7\rangle + \langle 4,8\rangle = 3$.

9. Bottom right square: The values of the green cells must be either all odd or all even. Moreover, if the green cells contain odd numbers, then the orange cell must contain an even number. If the green cells contain even numbers, then the orange cell must contain an odd number.

Your task is to:

(a) Implement the constraints into the template file `sudoku.z3` that we provide. There, we already included the basic definition of the Sudoku board, and print the values of all cells before Z3 terminates. Please, implement your constraints where the "; TODO constraints for X." indicates you and do **not** delete comments "BEGIN-CONSTRAINTS-X" or "END-CONSTRAINTS-X". You may delete the TODO comments.

    You can refer to the value of the cell $\langle i,j\rangle$ through `(Board x`*i* `y`*j*`)` (as exemplified in the `get-value` statements).

(b) Run Z3 on your file. Save the output in a file called "sudoku.log". Note: If Z3 does not find any solution (prints `unsat`), one of the constraints is not implemented correctly. If you have not modeled all constraints, Z3 might take a long time to find a solution. If you have modeled all constraints and Z3 still takes extremely long (> 1 minute depending on your hardware) until it terminates, it is very likely that you have a bug in your model.

Submission Instructions

Solutions need to be packaged into a `.zip` file and uploaded to Moodle. The file should be named as:

`CS5960-FirstName-LastName-StudentID-lab3`

The `.zip` file needs to contain the following files

- a PDF file with the solutions to the theoretical questions of this sheet.
- the python file with your implementation of the solution.

We will deduct 3 points if you do not adhere to these rules!

Statement	Description
General	
`(check-sat)`	Checks whether the CSP defined up to this point is satisfiable.
`(declare-const var Int)`	Declares a new variable with name `var`.
`(assert E)`	Adds boolean expression `E` as constraint.
`(get-value (E))`	Prints the value of `E`, where `E` can be an arbitrary expression such as constant, variable, function, or mathematical or boolean combination thereof (must occur after `(check-sat)`).
`(get-model)`	Prints all variable assignments (must occur after `(check-sat)`).
`(echo "message")`	Prints `message` to the console.
`; This is a comment`	Commenting.
Mathematical Expressions	
c	Constants $c \in \mathbb{Z}$.
`var`	Evaluates to the value of variable `var`.
`(Board x1 y2)`	Evaluates to the value of the cell with coordinates $\langle 1, 2 \rangle$
`(`$\circ$ E_1 `...` E_n`)`	Evaluates to $E_1 \circ E_2 \circ \cdots \circ E_n$, where $\circ$ can be any of `+`, `-`, and `*`.
Boolean Expressions	
`true`	Constant for true.
`false`	Constant for false.
`(not` E`)`	Negation of the boolean expression E.
`(and` E_1 `...` E_n`)`	Conjunction over the boolean expressions E_1 to E_n.
`(or` E_1 `...` E_n`)`	Disjunction over the boolean expressions E_1 to E_n.
`(`$\circ$ E_1 `...` E_n`)`	Is true iff for the evaluation of the expressions E_1 to E_n, it holds that $E_1 \circ E_2$ and $E_2 \circ E_3, \ldots,$ and $E_{n-1} \circ E_n$, where $\circ$ can be any of `<`, `<=`, `>`, `>=`, and `=`.
`(distinct` E_1 `...` E_n`)`	Is true iff every expression E_1 to E_n evaluates to a different value.

Table 1: Z3 input language fragment you may use for the CSP modeling exercises.