Formal Method Mod. 1 (Automated Reasoning)

Laboratory 2

Giuseppe Spallitta
giuseppe.spallitta@unitn.it

Università degli studi di Trento

March 24, 2021

Automating the encoding generation

- During the last lecture we spent a lot of time encoding the last problem, despite the low number of variables and constraints involved
  $\Rightarrow$ For harder tasks we would spend tons of hours simply in defining the DIMACS file!
- From now on we will write some code to automate the generation of the input file and easily read the output in a human friendly style.
- In the following examples I will use Python 3.x for its simplicity and readability, but feel free to use any programming languages you are confident with (e.g. C, C++, Java and others).

Outline

1. Advanced SAT solving
   - Logic puzzles
   - Solving Sudoku
   - Nonogram
2. SAT incrementality and UNSAT core extraction
3. Homeworks

Interview calendar

UNIVERSITÀ DEGLI STUDI DI TRENTO

Exercise 2.1: logic riddle

Bill has a series of job interviews this week (August 20th, 21st, 22nd and 23rd), each for a different type of position (copywriter, graphic design, sales rep and social media) at a different company (Alpha Plus, Laneplex, Sancode, Streeter Inc.). Using only the clues below, match each job position to its company, and determine the day for each interview and the town it will be held in. No option in any category can be used more than once.

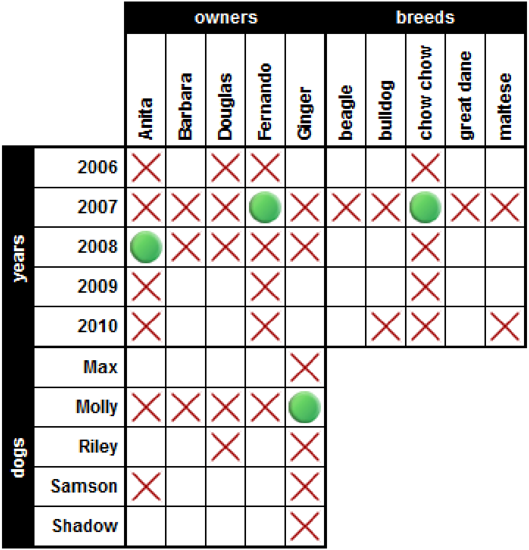
Interview calendar (cont.d)

Exercise 2.1: logic riddle (cont.d)

Some clues are given to us to determine each assignment:

- The Alpha Plus interview is 2 days before the meeting for the copywriter position.
- The meeting for the graphic design position is sometime after the Sancode interview.
- Of the interview for the sales rep position and the Laneplex interview, one is on August 23rd and the other is on August 20th.
- The Streeter Inc. interview is 2 days after the Alpha Plus interview.

How people usually solve it



Interview calendar: variables

As always, we first define the variables that efficiently describe the problem:

- x_{ij} states if day i ($i \in${20th,21st,22nd,23rd}) refers to a specific property, such as the company and the position available.
- For instance to store the status of August 20th we need 8 variables, one for each company and for each role. The same applies for the other 3 days, requiring $8 * 4 = 32$ variables.
- We must define a simple function to map each variable into an indexed variable and store this mapping while generating the constraints!

Now we can encode the clues stated by the problem, one by one:

Interview calendar: properties (1)

The Alpha Plus interview is 2 days before the meeting for the copywriter position.

- This means that either the Alpha Plus interview is on the 20th or the 21st of August and then the copywriter interview respectively on the 22nd or the 23rd.

$$(x_{0A} \wedge x_{2c}) \vee (x_{1A} \wedge x_{3c})$$

It is not in CNF form, but we can automate the process creating a dedicated function...
$\Rightarrow$ Maybe we can recycle it for future problems :)

Interview calendar: properties (2)

The meeting for the graphic design position is sometime after the Sancode interview.

- Given an interval of 4 days, we must exclude that the 20th of August is associated to the graphic design position, that the 23rd of August is associated to the Sancode interview and that 21st and 22nd are respectively the graphic design and Sancode interviews:

  $$\neg x_{0g} \wedge \neg x_{3S} \wedge \neg(x_{1g} \wedge x_{2S})$$

  It can be easily converted into CNF form, we will obtain 3 clauses.

Interview calendar: properties (3)

Of the interview for the sales rep position and the Laneplex interview, one is on August 23rd and the other is on August 20th.

$$(x_{0s} \wedge x_{3L}) \vee (x_{0L} \wedge x_{3s})$$

The Streeter Inc. interview is 2 days after the Alpha Plus interview.

$$(x_{0A} \wedge x_{2I}) \vee (x_{1A} \wedge x_{3I})$$

The two clues can be encoded using the same function used for the first clue passing different inputs.

Interview calendar: properties (4)

Do not forget to consider some hidden conditions to avoid the generation of non-valid assignments. In particular me must encode the following properties:

- Each day must be associated to exactly one company
- Each day must be associated to exactly one position
- Each company must be associated to exactly one day
- Each position must be associated to exactly one day

Encoding *ExactlyOne*

Encoding $ExactlyOne(x_1, ..., x_n)$ is easy if you split it into two simpler conditions:

$$ExactlyOne(x_1...x_n) = AtLeastOne(x_1...x_n) \wedge AtMostOne(x_1...x_n)$$

The latter conditions can be formalized (and consequently we can implement the respective function to automate the definition of the clauses) into the formulas:

$$AtLeastOne(x_1...x_n) = x_1 \vee x_2... \vee x_n$$

$$AtMostOne(x_1...x_n) = \{\neg x_i \vee \neg x_j \mid 0 \leq i < j \leq n\}$$

Interview calendar: results

Now we can fed the encoding into Glucose
$\Rightarrow$ The solver returns **SAT**, but probably we can write some code to quickly read the valid assignment in a human-readable format.

Solving an hard Sudoku

Exercise 2.2

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

Can we find the solution of this Sudoku using a SAT solver?

Solving Sudoku: variables

As always, we first define the variables that efficiently describe the problem:

- x_{ijk} states if number k is placed in the cell in row i and column j.
- We will instantiate $9 * 9 * 9 = 729$ different variables.

Solving Sudoku: properties (1)

The three basic rules to solve a Sudoku are the following:

- Each column contains all of the digits from 1 to 9
- Each row contains all of the digits from 1 to 9
- Each of the nine 3×3 subgrids contains all of the digits from 1 to 9

Defining each rule is not difficult: for each digit we encode an *ExactlyOne* constraints considering the right cells on the grid.

Solving Sudoku: properties (2)

From our previous experience, there is an hidden rule that we could encode:

- Each cell must contains exactly one number

Actually this constraint is unnecessary since they are logically implied by the previous rules, but adding them would lead to the same solution and a slightly higher computational time. constraints

Solving Sudoku: properties (3)

Lastly, we must add some constraints to indicate the digits which are already placed in the grid.

- For each digit already placed in the grid one of the 729 variables will be surely **TRUE**.
- The conjunction of all these true variables represents the specific configuration of the Sudoku we want to solve.

Solving Sudoku: results

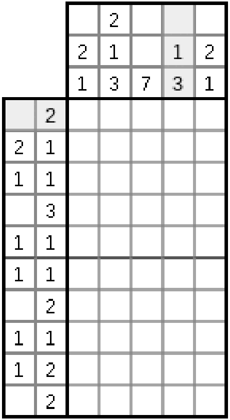
Now we can fed the encoding into Glucose
$\Rightarrow$ The solver returns **SAT**, so a solution exists. A brief manipulation of the output could help us in visualizing the result.

Solving a Nonogram

Exercise 2.3

	2	3	4	2	2
2					
3					
3					
3 1					
1					

Your aim in these puzzles is to colour the whole grid in to black and white squares or mark with X. Beside each row of the grid are listed the lengths of the runs of black squares on that row. Above each column are listed the lengths of the runs of black squares in that column. Can we find the solution of this Nonogram?



2 1
2 1 3
7
1 3
2 1
2
2 1
1 1
3
1 1
1 1
2
1 1
1 2
2

Solving Nonogram: variables

UNIVERSITÀ DEGLI STUDI DI TRENTO

As always, we first define the variables that efficiently describe the problem:

- x_{ij} states if cell in row i and column j should be black.
- We will instantiate $5 * 5 = 25$ different variables.

Solving Nonogram: properties(1)

For each row and each column we must define a constraint considering the valid position of black cells. Some will be easier than the others to define; we will see all of them and try to automate the most of the process.

- For instance, row 4 is straightforward: there is a single valid color assignment for each satisfying the constraint, so we can easily encode the conjunction of this trivial and unique assignment.
- Row 5 is also trivial: exactly one of the cells should be black.

Solving Nonogram: properties(2)

Let's define a constraint for a non-trivial row, the first one.

- If two subsequent cells must be black, there are 4 valid assignments to consider:

$$
\begin{aligned}
&(x_{00} \wedge x_{01} \wedge \neg x_{02} \wedge \neg x_{03} \wedge \neg x_{04}) \\
\vee&(\neg x_{00} \wedge x_{01} \wedge x_{02} \wedge \neg x_{03} \wedge \neg x_{04}) \\
\vee&(\neg x_{00} \wedge \neg x_{01} \wedge x_{02} \wedge x_{03} \wedge \neg x_{04}) \\
\vee&(\neg x_{00} \wedge \neg x_{01} \wedge \neg x_{02} \wedge x_{03} \wedge x_{04})
\end{aligned}
$$

Solving Nonogram: properties(3)

Converting this formula into a CNF equivalent representation is not trivial and writing the code to define this translation could be a challenging task...

$\Rightarrow$ But do not forget the existence of Tseitin's transform!

$$a_1 \Leftrightarrow (x_{00} \wedge x_{01} \wedge \neg x_{02} \wedge \neg x_{03} \wedge \neg x_{04})$$

$$a_2 \Leftrightarrow (\neg x_{00} \wedge x_{01} \wedge x_{02} \wedge \neg x_{03} \wedge \neg x_{04})$$

$$a_3 \Leftrightarrow (\neg x_{00} \wedge \neg x_{01} \wedge x_{02} \wedge x_{03} \wedge \neg x_{04})$$

$$a_4 \Leftrightarrow (\neg x_{00} \wedge \neg x_{01} \wedge \neg x_{02} \wedge x_{03} \wedge x_{04})$$

$$a_1 \vee a_2 \vee a_3 \vee a_4$$

Solving Nonogram: properties(4)

- The idea behind the other rows and columns is similar, so we can provide a sort of generalized algorithm to manage the remaining constraints.
- In this case there are no "hidden" constraints to configure, the only clues to solve the puzzle are the the numbers near the grid.

Solving Nonogram: results

Now we can fed the encoding into Glucose
$\Rightarrow$ The solver returns **SAT**, so a solution exists. A brief manipulation of the output could help us in visualizing the result; given the simplicity of the chosen nomenclature we can also quickly understand the solution scanning the result.

Outline

1. Advanced SAT solving
2. SAT incrementality and UNSAT core extraction
3. Homeworks

Working as a receptionist

Exercise 2.4: receptionist

You are a receptionist in a prestigious hotel and you are waiting 5 new guests. There are 5 available rooms, but you don't know their preferences about the room they want to book until the last moment:

- Guest A would like to choose room 1 or 2.
- Guest B would like to choose a room with an even number.
- Guest C would like the first room.
- Guest D has the same behaviour as user B.
- Guest E would like one of the external rooms.

Supposing the guests come one after the other, is there a moment where it is not possible to help every guest? How many guests can be sorted without problems?

Working as a receptionist: modeling

- The core of the model is easy to define: we need a variable for each pair guest-room. For each guest and each room we define an *ExactlyOne* constraint.
- To test how many guests can be sorted without issues, we can take advantage of incremental SAT solving
  $\Rightarrow$ The preferences of the guests can be encoded adding selection variables and we can progressively add the various constraints

Why incremental SAT?

What are the advantages of employing incremental SAT?

- SAT formula can be modified and solved again, while reusing information from previous solving steps!
- Really useful when the core model is not trivial
- Effective when applied in "Planning as SAT" problems.

Working as a receptionist: results

- Automating the process we notice that, once we add the fourth guest, there is a conflict.
- Which are the elements that causes the conflict?
  $\Rightarrow$ We can use the $get\_core()$ function to see the list of literals causing the conflict

Outline

UNIVERSITÀ DEGLI STUDI DI TRENTO

1. Advanced SAT solving
2. SAT incrementality and UNSAT core extraction
3. Homeworks

Homeworks

Homework 2.1: diagonal sudoku

Diagonal Sudoku

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

Rajesh Kumar @ www.FunWithPuzzles.com

Can we find the solution of this Diagonal Sudoku using a SAT solver?

Homeworks

UNIVERSITÀ DEGLI STUDI DI TRENTO

Homework 2.2: puzzle baron

Adapt the code written for exercise 2.1 to solve similar puzzles from the following website: `https://logic.puzzlebaron.com/`.

- To obtain an exercises almost identical to the one shown in class select the "3*4 grid" and the "challenging" difficulty.
- Adapt the code to manage a "3*5" grid, if you can.

Homeworks

UNIVERSITÀ DEGLI STUDI DI TRENTO

Homework 2.3: the n-queens problem

The n-queens problem is to place n chess queens on an $n * n$ chessboard such that no two queens are mutually attacking (i.e., in the same row, column, or diagonal).

- Solve the n-queens problem with $n = 8$.
- Is the solution obtained unique?