UNIVERSITÀ
DI TRENTO
Department of
Information Engineering and Computer Science

Automated Reasoning and Formal Verification

Laboratory 2

Gabriele Masina
gabriele.masina@unitn.it

Università di Trento

March 12, 2025

These slides are derived from those by Giuseppe Spallitta for FM lab 2021-2024.

Automating the encoding generation

- In the last lab, encoding the last problems into DIMACS or SMT-LIB took quite a bit of time, despite only a few variables and constrains were involved.
  $\Longrightarrow$ Not feasible for larger tasks!
- We will see how to write code to automate the generation of the input file and easily read the output.
- We will use the **PySMT** library for **Python3**
  `https://pysmt.readthedocs.io/en/latest/index.html`
- If you prefer, you can use any other programming language and library you are comfortable with.

The PySMT library

- Intuitive interface to build and manipulate formulas

```
from pysmt.shortcuts import Symbol, And, Or, Not, BOOL, Solver
A, B = Symbol("A", BOOL), Symbol("B", BOOL)
f1 = Or(And(A, Not(B)), Not(A))
f2 = (A & ~B) | ~A                          # supports infix notation
```

- Common interface for several SAT/SMT solvers (MathSAT, Z3, CVC5, etc.):

```
with Solver(name="msat") as solver:
    solver.add_assertion(f1)
    if solver.solve():
        print("SAT", solver.get_model())
    else:
        print("UNSAT")
```

- Installation: `pip install pysmt && pysmt-install --msat`

Outline

Interview calendar

Exercise 2.1: logic riddle

Bill has 4 job interviews this week (Aug. 20th, 21st, 22nd, 23rd), each for a different position (copywriter, graphic design, sales rep, and social media) at a different company (Alpha Plus, Laneplex, Sancode, Streeter Inc.). Knowing that:

- The Alpha Plus interview is 2 days before the copywriter one
- The graphic design interview is after the Sancode interview
- Of the interview for the sales rep position and the Laneplex interview, one is on Aug. 23rd and the other is on Aug. 20th
- The Streeter Inc. interview is 2 days after that for Alpha Plus
- No social media interview is on Aug. 23

Match each job position to its day and company. No option in any category can be used more than once.

How people usually solve it

		position				company			
		copywriter	graphic design	sales rep	social media	Alpha Plus	Laneplex	Sanbox	Streeter Inc.
day	20th	X	X						
	21st	X		X			X		
	22nd			X		X	X		
	23rd					X		X	
company	Alpha Plus	X							
	Laneplex	X		X					
	Sanbox	X	X						
	Streeter Inc.	●	X	X	X				

Interview calendar: variables

As always, we first define the variables to describe the problem:

- c_{ij} states if in day i there is an interview for company j
- p_{ik} states if in day i there is an interview for position k
- For instance to store the status of Aug. 20th we need 8 variables, one for each company and for each role. The same applies for the other 3 days, requiring $8 \cdot 4 = 32$ variables.

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 copywriter one.

- 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.

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

Interview calendar: properties (2)

The graphic design interview is after the Sancode interview

$$(c_{0S} \wedge p_{1g}) \vee (c_{0S} \wedge p_{2g}) \vee (c_{0S} \wedge p_{3g}) \vee (c_{1S} \wedge p_{2g}) \vee (c_{1S} \wedge p_{3g}) \vee (c_{2S} \wedge p_{3g})$$

Interview calendar: properties (3)

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

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

The Streeter Inc. interview is 2 days after that for Alpha Plus

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

No social media interview is on Aug. 23

$$\neg p_{3m}$$

Interview calendar: properties (4)

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

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

Encoding *ExactlyOne*

Encoding $\mathit{ExactlyOne}(x_1, \ldots, x_n)$ is easy if you split it into two simpler conditions:

$$\mathit{ExactlyOne}(x_1 \ldots x_n) = \mathit{AtLeastOne}(x_1 \ldots x_n) \wedge \mathit{AtMostOne}(x_1 \ldots x_n)$$

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

$$\mathit{AtLeastOne}(x_1 \ldots x_n) = x_1 \vee x_2 \cdots \vee x_n$$

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

PySMT provides a shortcut `ExactlyOne(...)` to encode this. However, it is useful to know how to encode it manually!

Interview calendar: results

Now we can feed the encoding to MathSAT
$\Rightarrow$ The solver returns **SAT**, but we can write some code to quickly transform the model into a human-readable format.

Solving a Nonogram

Exercise 2.3

	2	3	4	2	2
2					
3					
3					
3 1					
1					

You have a grid of squares, which must be either filled in black or marked 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.

Your aim is to find all black squares.

Solving Nonogram: variables

As always, we first define the variables to describe the problem:

- x_{ij} states if cell in row i and column j should be black.
- We will instantiate $5 \cdot 5 = 25$ 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 others to define; we will see all of them and try to automate 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:

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

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

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

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

Solving Nonogram: properties(3)

Converting this formula into a CNF equivalent representation or writing it into a single assertion is not trivial and could be a challenging task...
$\Rightarrow$ But do not forget the existence of Tseitin's transform if you generate directly SMT-LIB files!

$$a_1 \Leftrightarrow (x_{00} \wedge x_{01} \wedge \neg x_{02} \wedge \neg x_{03} \wedge \neg x_{04})$$
$$a_2 \Leftrightarrow (\neg x_{00} \wedge x_{01} \wedge x_{02} \wedge \neg x_{03} \wedge \neg x_{04})$$
$$a_3 \Leftrightarrow (\neg x_{00} \wedge \neg x_{01} \wedge x_{02} \wedge x_{03} \wedge \neg x_{04})$$
$$a_4 \Leftrightarrow (\neg x_{00} \wedge \neg x_{01} \wedge \neg x_{02} \wedge x_{03} \wedge x_{04})$$
$$a_1 \vee a_2 \vee a_3 \vee a_4$$

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

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 to this Sudoku using a SAT solver?

Solving Sudoku: variables

As always, we first define the variables to describe the problem:

- x_{ijk} is true iff number k is placed in the cell in row i and column j.
- We will instantiate $9 \cdot 9 \cdot 9 = 729$ 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* constraint considering the right cells on the grid.

Solving Sudoku: properties (2)

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

- Each cell must contain exactly one number

But notice how the three Sudoku basic rules already ensure the uniqueness of the digit in a single cell!

Solving Sudoku: properties (3)

Lastly, let's add the constraints to indicate the digits that are already placed in the grid.

- For each digit already placed in the grid, we just assert the variable corresponding to its position.

Solving Sudoku: results

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

Outline

Working as a receptionist

Exercise 2.4: receptionist

You are a receptionist in a prestigious hotel and you are waiting for 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 behavior 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?

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.

Incremental SAT in PySMT

1. Create the core model and add it as assertions;
2. Call push to create a new level (everything before that level is still considered)
3. Add a new assertion and call `solve()`
4. If true, continue the `for` cycle adding a new level and a new assertion;
5. If false, stop the procedure.

Incremental SAT in PySMT: example

```
list_assignments = [
    [A],
    [A, B],
    [A, B, C]
]
for a in list_assignments:
    msat.push()
    msat.add_assertion(And(a))
    msat.solve()
    ...
```

Attention

You can also remove a level by using `msat.pop()`, if you want some conditions (other than the core, which is immutable) to be removed!

Working as a receptionist: modeling

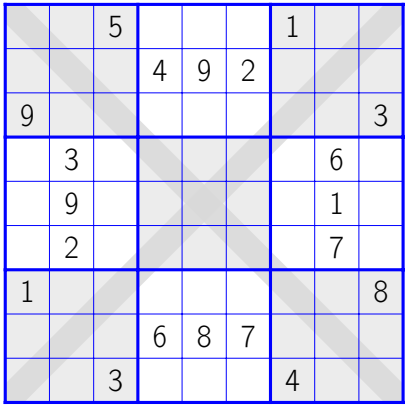
- The core of the model is easy to define: we need a variable for each pair guest-room. For each guest and each room, we define an *ExactlyOne* constraint.
- To test how many guests can be sorted without issues, we can take advantage of incremental SAT solving
  $\Rightarrow$ The preferences of the guests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

Working as a receptionist: results

- We notice that, once we add the fourth guest, there is a conflict.
- Which are the elements that cause the conflict?
  $\Rightarrow$ We can use the `get_unsat_core()` function to see the list of literals causing the conflict
  $\Rightarrow$ We must add `Solver("msat", unsat_cores_mode="all")`

Outline

Homework 2.1: diagonal sudoku



A Diagonal Sudoku has the same rules as a classic Sudoku, but the two main diagonals must also contain the numbers from 1 to 9.

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

Homeworks

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.

Homework 2.3: the n-queens problem

The n-queens problem is to place n chess queens on an $n \times n$ 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?