# Z3 Solver - SAT

## A SAT approach to solving problems

Diego Mazzieri

Alma Mater Studiorum · Università di Bologna

May 21, 2021

# Introduction

# Z3 Theorem Prover

## What is it?

- Z3 is a state-of-the-art SAT/SMT theorem prover developed at Microsoft Research and used in many real-world applications.
  - $\Rightarrow$ Software/hardware verification and testing, constraint solving, analysis of hybrid systems, geometrical problems, etc
- Recipient of the 2019 Herbrand Award, an award given for important contributions to the field of automated deduction.

## How to use it?

- The Z3 input format is an extension of the SMT-LIB 2.0 standard
- APIs for common programming languages like `Python`, `Java`, `C`, and `.Net` are officially available in order to ease implementation and favor its use as a component in the context of other tools that require solving logical formulas.

# Installation

We will use the Python interface of Z3 called Z3Py. The easiest way to install it, along with the Z3 binary, is to use Python's package manager `pip`:

```
pip install z3-solver
```

In order to check if Z3 has been installed properly, open a terminal and type:

```
z3 --version
```

The current latest version is the '4.8.X', therefore the output should be something like *Z3 version 4.8.X - 64 bit*.

# Tutorial

Before doing some exercises together, let's get familiar with Z3Py syntax. Just download the `Z3 Tutorial.ipynb` from the following link:

`https://github.com/mazzio97/CDMO-course-2021/blob/main/Z3%20Tutorial.ipynb`

# Knights & Knaves

# Description

## Description

- There is an island in which certain inhabitants, called knights, always tell the truth, and the others, called knaves, always lie. It is assumed that every inhabitant of this island is either a knight or a knave.
- Suppose an inhabitant A says: "Either I am a knave or B is a knight." What are A and B?


# Encoding

## Variables

- We should consider that both A and B can be either a knight or a knave but nothing else.
- We can represent these facts simply defining two boolean variables, one for each individual, representing if he is a knight (or alternatively a knave).

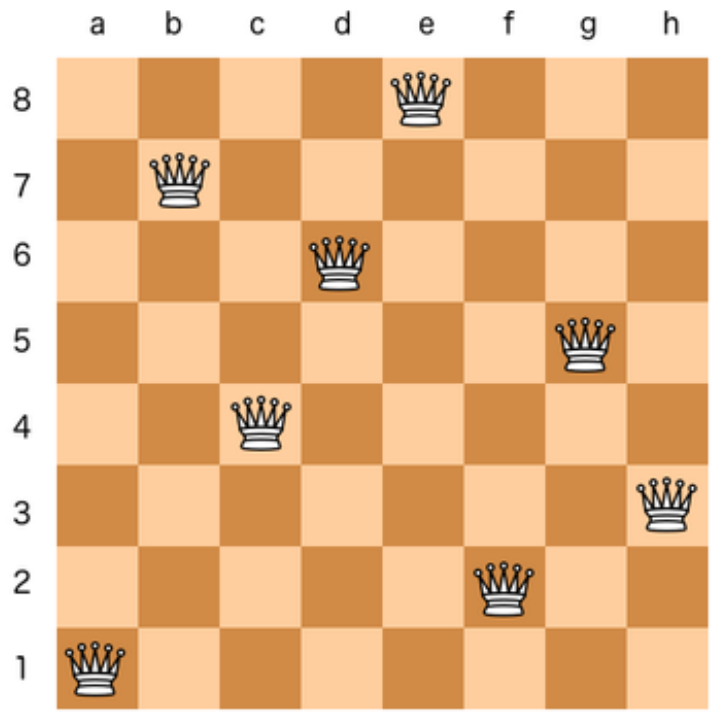
## Constraints

- From what A states, we can conclude that $A_{knave} \vee B_{knight}$.
- If A is a knight, then A's statement is true.
- If A is a knave, then A's statement is a lie.

# N-Queens

# Description

Place $N$ queens in an $NxN$ board so that no two queens can attack each other.



# Variables

Each cell in the board corresponds to a boolean variable indicating whether or not a queen is placed in that particular position.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $p_{11}$ | $p_{12}$ | $p_{13}$ | $p_{14}$ | $p_{15}$ | $p_{16}$ | $p_{17}$ | $p_{18}$ |
| $p_{21}$ | $p_{22}$ | $p_{23}$ | $p_{24}$ | $p_{25}$ | $p_{26}$ | $p_{27}$ | $p_{28}$ |
| $p_{31}$ | $p_{32}$ | $p_{33}$ | $p_{34}$ | $p_{35}$ | $p_{36}$ | $p_{37}$ | $p_{38}$ |
| $p_{41}$ | $p_{42}$ | $p_{43}$ | $p_{44}$ | $p_{45}$ | $p_{46}$ | $p_{47}$ | $p_{48}$ |
| $p_{51}$ | $p_{52}$ | $p_{53}$ | $p_{54}$ | $p_{55}$ | $p_{56}$ | $p_{57}$ | $p_{58}$ |
| $p_{61}$ | $p_{62}$ | $p_{63}$ | $p_{64}$ | $p_{65}$ | $p_{66}$ | $p_{67}$ | $p_{68}$ |
| $p_{71}$ | $p_{72}$ | $p_{73}$ | $p_{74}$ | $p_{75}$ | $p_{76}$ | $p_{77}$ | $p_{78}$ |
| $p_{81}$ | $p_{82}$ | $p_{83}$ | $p_{84}$ | $p_{85}$ | $p_{86}$ | $p_{87}$ | $p_{88}$ |

# Constraints

Column constraints are the same as the row constraints with $i$ and $j$ swapped.

- At least one queen on each row and column

$$\bigwedge_{1\le i\le n}\bigvee_{1\le j\le n} p_{ij} \quad \bigwedge_{1\le j\le n}\bigvee_{1\le i\le n} p_{ij}$$

- At most one queen on each row and column

$$\bigwedge_{1\le i\le n}\bigwedge_{0<j<k\le n} \neg(p_{ij}\wedge p_{ik}) \quad \bigwedge_{1\le j\le n}\bigwedge_{0<i<k\le n} \neg(p_{ij}\wedge p_{kj})$$

- At most one queen on each diagonal

$$\bigwedge_{1\le i<i'\le n\ j,j':i+j=i'+j'\ \vee\ i-j=i'-j'} \neg(p_{ij}\wedge p_{i'j'})$$

# Sudoku

# Description

The goal is to insert the numbers in the cells to satisfy the following conditions:

- Each row...
- Each column...
- Each 3x3 box...

... must contain the digits 1 through 9 exactly once.

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

# Variables

Each number which can be inserted in a cell of the sudoku corresponds to a boolean variable indicating if the number is located or not in that particular position.

$$v_{i,j,k} \in \{0,1\},\ i,j,k \in \{1,\ldots,9\}$$

# Constraints

- Value in each cell is valid:
  - For $i, j \in \{1, \ldots, 9\}$:

$$\sum_{k=1}^{9} v_{i,j,k} = 1$$

- Each value used exactly once in each row:
  - For $i \in \{1, \ldots, 9\}$, $k \in \{0, \ldots, 9\}$:

$$\sum_{j=1}^{9} v_{i,j,k} = 1$$

- Each value used exactly once in each column:
  - For $j \in \{1, \ldots, 9\}$, $k \in \{1, \ldots, 9\}$:

$$\sum_{i=1}^{9} v_{i,j,k} = 1$$

- Each value used exactly once in each $3 \times 3$ sub-grid:
  - For $i, j \in \{0, 1, 2\}$, $k \in \{1, \ldots, 9\}$:

$$\sum_{r=1}^{3} \sum_{s=1}^{3} v_{3i+r,3j+s,k} = 1$$