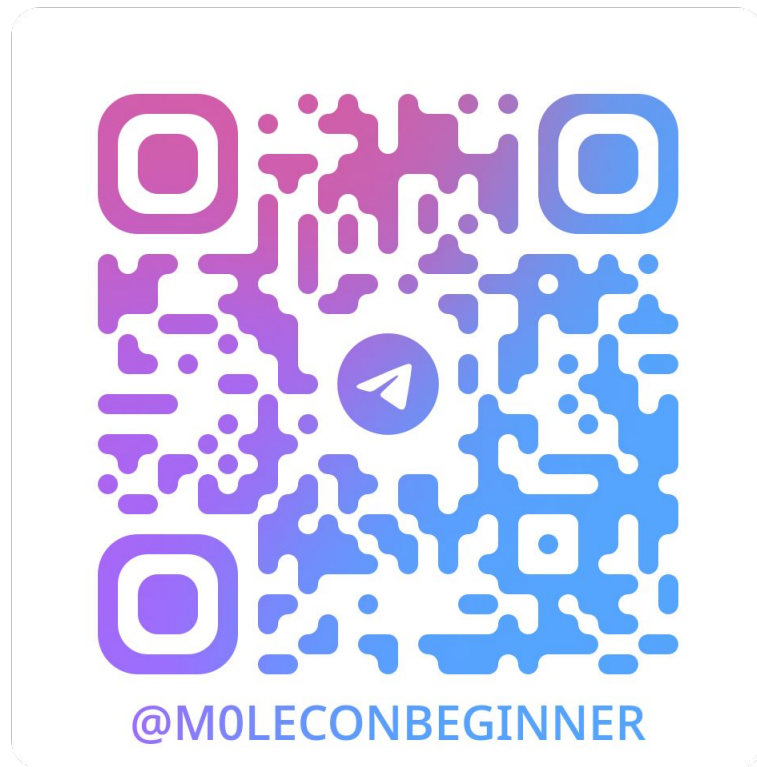


Intro to Z3 solver

Join the Telegram channel



What is Z3?

<https://github.com/Z3Prover/z3>

Theorem prover

Efficient SMT solver

MAGIC!

Symbolic logic engine

A “model”?

Symbolic variables

(x) (x)
 (x) (x)

Operations and constraints



$(x) \geq 42$

SMT solver

Problem of determining whether a mathematical formula is *satisfiable*, i.e. there **exists solutions given the relationships between the variables and the defined constraints**

Example

$x \ \& \ y \ \& \ (x \wedge z)$

x	y	z	$x \& y \& (x \wedge z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

We do security, not math. So?

- Formal testing of functional requirements (input -> output)
- Automatic static analysis looking for vulnerabilities
- Symbolic execution
- Automate finding of input to reach a specific point of the program (i.e. the “win” function)

Real example: **angr**
(true, real, magic)

<https://angr.io/>

[SMT Solvers for Software Security](#)

Z3 with Python

<https://ericpony.github.io/z3py-tutorial/>

```
pip install z3-solver
```

Recall math problems at high school?

1. Define variables
and create a “solver”
2. Add constraints from your problem
3. Solve!

```
from z3 import *

# Define vars and solver
x = BitVec('x', 1)
y = BitVec('y', 1)
z = BitVec('z', 1)
s = Solver()

# Add constraints
s.add((x & y & (x ^ z) == 1))

# Solve!
if s.check() == sat:
    m = s.model()
    print(m)
```


Main symbolic variable types and operators

- `Int`
- `Real`
- `Bool`
- `BitVec[bitsize]`

Z3 variables can then be used as any other Python variable for computations. Standard operators (+, -, &, ^, <<, ...) are supported.

Alternatively Z3 specific operator classes can be used **`And, Or, Xor, Not, ...`**

Special techniques for representing control flows and other advanced modeling e.g. **`If, Then, Sum, Distinct, ForAll, Exists...`**

Try it yourself: 8-Queens problem

Place 8 chess queens on an 8×8 chessboard so that no two queens threaten each other, i.e. max 1 queen for row, column and diagonal

Hint

Checkout the **Sum** operator!

Challenge: solving Sudoku

Hint

Checkout the **Distinct** operator!

<https://workshop.mOlecon.it/challenges#sudoku-30>

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

Challenge: easy crackme, find the hidden key

<https://workshop.m0lecon.it/challenges#crackmat-32>

1. Open the executable in a decompiler (Ghidra, IDA, Binary Ninja)
2. Rewrite the model in Z3
3. Solve

Easy right?

Challenge: Android Guessing Game

Let's recap. Can you recover the original number guess??

<https://workshop.m0lecon.it/challenges/GuessingGame-33>