

# CS228 Logic for Computer Science 2022

## Lecture 12: Encoding into reasoning problems

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2022-01-23

# Topic 12.1

## Z3 solver

# Solver basic interface

- ▶ Input : formula
- ▶ Output: sat/unsat

If satisfiable, we may ask for a satisfying assignment.

## Exercise 12.1

*What can we ask from a solver in case of unsatisfiability?*

## Z3: SMT solver

- ▶ Written in C++
- ▶ Provides API in C++ and Python
- ▶ We will initially use python interface for quick ramp up
- ▶ Later classes we will switch to C++ interface

## Installing Z3 (Ubuntu-18.04)

```
$sudo apt-get install z3 python3-z3
```

Not tested on 20.04

**Commentary:** You may also try `$pip3 install python3-z3`

## Locally Installing a version of Z3 (Linux)

Let us install z3-4.7.1. You may choose another version.

- ▶ Download

```
https://github.com/Z3Prover/z3/releases/download/z3-4.7.1/z3-4.7.1-x64-ubuntu-16.04.zip
```

- ▶ Unzip the file in some folder. Say

```
/path/z3-4.7.1-x64-ubuntu-16.04/
```

- ▶ Update the following environment variables

```
$export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/z3-4.7.1-x64-ubuntu-16.04/bin  
$export PYTHONPATH=$PYTHONPATH:/path/z3-4.7.1-x64-ubuntu-16.04/bin/python
```

- ▶ After the setup the following call should throw no error

```
$python3 /path/z3-4.7.1-x64-ubuntu-16.04/bin/python/example.py
```

## Topic 12.2

### Using solver

## Steps of using Z3 via python interface

```
from z3 import *           # load z3 library

p1 = Bool("p1")            # declare a Boolean variable
p2 = Bool("p2")

phi = Or( p1, p2 )         # construct the formula
print(phi)                 # printing the formula

s = Solver()               # allocate solver
s.add( phi )               # add formula to the solver
r = s.check()              # check satisfiability
if r == sat:
    print("sat")
else:
    print("unsat")         # save the script test.py
                           # run \python3 test.py
```



## Get a model

```
r = s.check()
if r == sat:
    m = s.model()      # read model
    print(m)           # print model
else:
    print("unsat")
```

### Exercise 12.2

What happens if we run `m = s.model()` in the unsat case?

## Solve and print model

```
from z3 import *

# packaging solving and model printing
def solve( phi ):
    s = Solver()
    s.add( phi )
    r = s.check()
    if r == sat:
        m = s.model()
        print(m)
    else:
        print("unsat")

# we will use this function in later slides
```

# Pointer and variable

There is a distinction between the Python variable name and the propositional variable it holds.

```
from z3 import *      # load z3 library

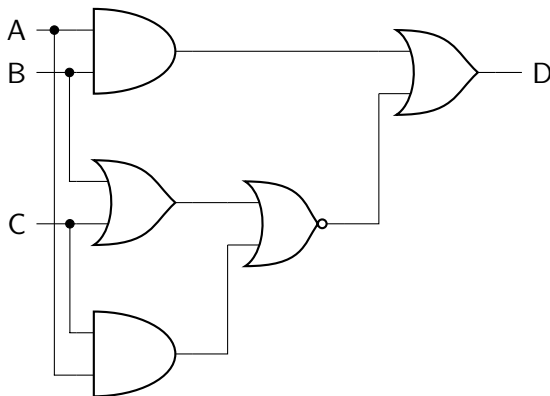
x = Bool("y")          # creates Propositional variable y

z = x                  # python pointer z also holds variable y
```

## Exercise: encoding Boolean circuit

### Exercise 12.3

Using Z3, find the input values of  $A$ ,  $B$ , and  $C$  such that output  $D$  is 1.



We know you can do it! Please do not shout the answer. Please make computer find it.

## Topic 12.3

### Solver engineering

## Design of solvers: context vs. solver

Any complex software usually has a context object.

The context consists of a **formula store** containing the constructed formulas.

Z3 Python interface **instantiates a default context**. Therefore, we do not see it explicitly.

A `Solver` is a solving instance. There can be **multiple solvers** in a context.

The `Solver` solves only the added formula.

## Formula handling

```
a = Bool('a')
b = Bool('b')
ab = And( a, b )

# accessing sub-formulas
print(ab.arg(0))
print(ab.arg(1))

# accessing the symbol at the head
ab_decl = ab.decl()
name = ab_decl.name()
if name == "and":
    print("Found an And")
```

## Topic 12.4

### Problems



## Exercise : Python programming

### Exercise 12.4

Write a Python program that generates a random graph in a file `edges.txt` for  $n$  nodes and  $m$  edges, which are given as command line options.

Please store edges in `edges.txt` as the following sequence of tuples

10,12

30,50

...

### Exercise 12.5

Write a program that reads a directed graph from `edges.txt` and finds the number of **strongly connected components** in the graph

### Exercise 12.6

Write a program that reads a directed graph from `edges.txt` and finds the cliques of size  $k$ , which is given as a command line option.

# Proving theorems

## Exercise 12.7

*Prove/disprove the following theorems using a solver*

- ▶ *Sky is blue. Space is black. Therefore sky and space are blue or black.*
- ▶ *Hammer and chainsaw are professional tools. Professional tools and vehicles are rugged. Therefore, hammers are rugged.*

## Write a function: find positive variables

### Exercise 12.8

*Find the set of Boolean variables that occur only positively in a propositional logic formula.*

*An occurrence of a variable is positive if there are even number of negations from the occurrence to the root of the formula.*

*Examples:*

*Only  $q$  occurs positively in  $p \wedge \neg(\neg q \wedge p)$ .*

*$p$  occurs positively in  $\neg\neg p$ .*

*$p$  does not occur positively in  $\neg p$ .*

*$p$  and  $q$  occur positively in  $(p \vee \neg r) \wedge (r \vee q)$ .*

End of Lecture 12