CS228 Logic for Computer Science 2020

Lecture 13: Using SAT sovler

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2020-02-08

Topic 13.1

Z3 solver

Solver basic interface

- Input : formula
- Output: sat/unsat

If satisfiable, we may ask for a satisfying assignment.

Exercise 13.1

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

Z3: SMT solver

- Written in C++
- Provides API in C++ and Python

Locally Installing Z3 (Linux)

- 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

- After the setup the following call should throw no error

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

Steps of using Z3 via the 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
```

Commentary: In python 3.0 onward one may have to write `print("sat")` instead of `print "sat"`

Get a model/assignment

In z3, assignments are called models

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

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

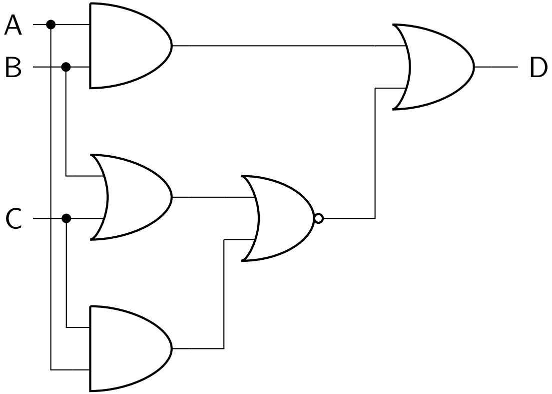
the Python variable name and the propositional variable it holds.

```
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 13.2

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 counter find it.

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.

Attendance quiz

Which of the following hold about the Z3 API?

x = Bool("x") creates Boolean variable x
phi = Or(p1, p2) creates a conjunction of two formulas
phi = And(p1, p2) creates a disjunction of two formulas
s = Solver() create an instantiation of a Z3 solver
For solver s, s.add(F) adds formula F in solver s
For solver s, s.check() runs CDCL on the conjunction of formulas added to s so far
For solver s, s.model() returns an assignment if s.check() has returned "sat"
For solver s, s.model() throws an exception if s.check() has returned "unsat"
For solver s, s.model() throws an exception if s.check() has not been called
y = Bool("x") creates Boolean variable y
phi = Or(p1, p2) creates a disjunction of two formulas
phi = And(p1, p2) creates a conjunction of two formula
s = Solver() create an instantiation of a shopping mall
For solver s, s.add(F) removes formula F in solver s
For solver s, s.check() runs DPLL on the conjunction of formulas added to s so far
For solver s, s.model() returns an assignment if s.check() has returned "unsat"
For solver s, s.model() throws an exception if s.check() has returned "sat"
For solver s, s.model() returns an assignment if s.check() has not been called

Topic 13.2

Problems

Exercise : Python programming

Exercise 13.3

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 13.4

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

Exercise 13.5

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 13.6

Prove/disprove the following theorems

- *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.*

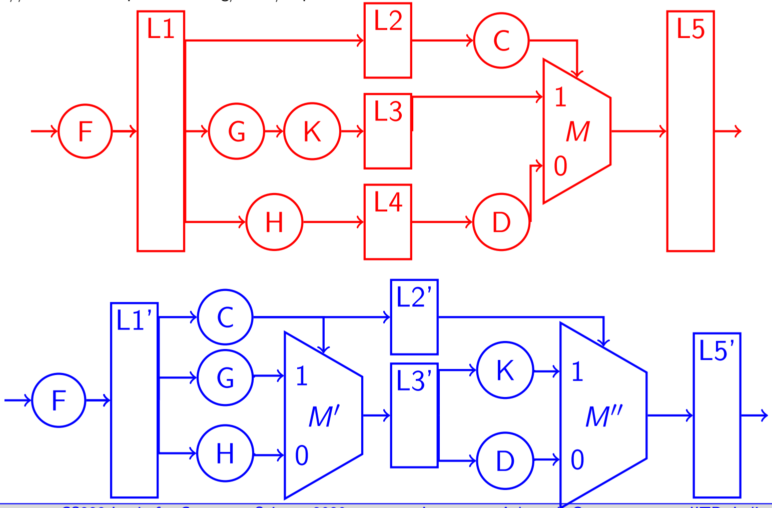
Exercise: translation validation

Exercise 13.7

Show that the following two circuits are equivalent.

Ls are latches, circles are Boolean circuts, and Ms are multiplexers.

Source: http://www.decision-procedures.org/slides/uf.pdf



Write a function: find positive variables

Exercise 13.8

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* $p \land \neg(\neg q \land p)$.

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

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

p *and* q *occur positively in* $(p \lor \neg r) \land (r \lor q)$.

Write a function: find unrelated constraints

Exercise 13.9

Consider a CNF formula F. Find the subsets of F that have disjoint set of symbols.

Examples:

$(x \vee y) \wedge (x \vee z) \wedge u$ *has two unrelated subsets* $\{(x \vee y), (x \vee z)\}$ *and* $\{u\}$

Write a function: find maximum occurring symbol

Exercise 13.10

Consider a formula F. Find the variable in F that occurs most often.

Example:

x occurs most often in $(x \vee \neg x) \wedge (y \vee \neg x)$.

Write a function: detect CNF or NNF

Exercise 13.11

Consider a formula F. Write a function that detects if F is CNF.

Exercise 13.12

Consider a formula F. Write a function that detects if F is NNF.

End of Lecture 13