Inside the solver

Ashutosh Gupta

IITB, India

Compile date: 2022-11-20

Topic 1.1

Supporting tools

Tools

Let us install a few tools.

- linux
- emacs
- git
- python3
- g++
- make
- gdb
- Eclipse for C++ (Eclipse installer asks for the choice of installation.)
- z3 source `https://github.com/Z3Prover/z3`
    - compile in debug and trace mode

Z3 in debug mode

Please follow these commands to compile z3 in debug mode

```
cd ~/eclipse-workspace
wget https://github.com/Z3Prover/z3/archive/z3-4.8.7.tar.gz
tar -xvzf z3-4.8.7.tar.gz
mv z3-z3-4.8.7 z3
git clone https://github.com/Z3Prover/z3.git
rm -rf z3-build
mkdir -p z3-build
cd z3-build
cmake -G "Eclipse CDT4 - Unix Makefiles" -DCMAKE_BUILD_TYPE=Debug ../z3
make
```

Get the input test files from the course website and put in `~/eclipse-workspace/z3-build`

IDE and gdb

We need to learn to use an IDE and a debugger before start looking inside Z3

Here, we will use Eclipse and gdb on linux.

You may choose any other IDE and debugger combination.

Setting Eclipse CDT for Z3

- Start Eclipse and choose folder "`eclipse-workspace`" in the home directory as workspace
- File → Import → C/C++ → Existing code as Makefile Project
  - Project Name = z3-build
  - Existing Code location : choose z3-build folder from file navigation
- Build all using Ctrl-B
- Run → Run configurations...
  - Project = z3-build
  - C/C++ Application = choose z3 executable inside z3-build
  - Arguments = class.smt2

Debugging using Eclipse

Some commands in gdb

- F11 runs the program in debug mode
  - Program will stop at the main entry point
- Inserting breakpoints
  - Navigate to source in left pane Project-explorer → Source directory
  - By double clicking on left of source line number
- F5 // steps in the program
- F6 // steps over the function calls
- F7 // finish current function
- F8 // continue to next breakpoint
- p [code] // executes the code

Topic 1.2

Engineering for CDCL(T)

Engineering CDCL(T)

Now we will look into the internals of Z3.

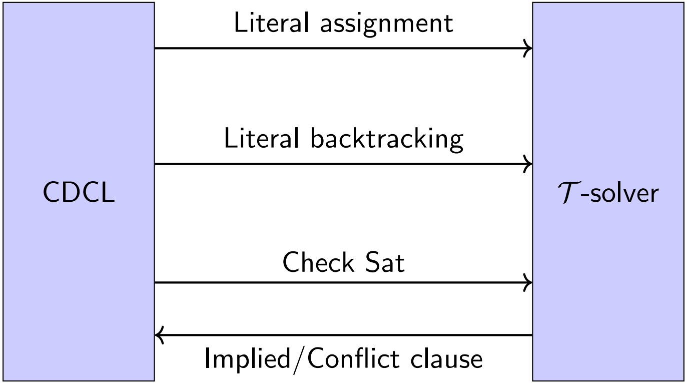
Key ideas to learn in implementation design

- term management in Z3
- tactics layer
- CDCL implementation
- base theory(QF_EUF) implementation

Key ideas to learn in software design

- be comfortable with large code base
- memory management
- customized library support

CDCL(T) architecture



Running Z3 in emacs gdb

- Consider the following SMT problem in *class.smt2*

```
(declare-sort U 0)
(declare-fun f (U) U)
(declare-const a U)

(assert (or  (= (f(f a)) a)    (= (f(f(f(f a)))) a)  )   )
(assert (or  (= (f(f(f a))) a) (= (f a)  (f(f a)) )  )   )
(assert (not (= (f a) a) ) )

(check-sat)
```

Exercise 1.1

a. *Is the above sat?*

b. *Run z3 with the above input by pressing F11 and then F8*

Start of solving: setup_and_check

- setup_and_check is entry the point of SMT solver
- Some simplification are applied to the input before reaching here

Look at [Source directory]/src/smt/smt_context.cpp:3366

Exercise 1.2

a. place a breakpoint there and rerun the binary till the breakpoint.

b. Go to debugger console in the bottom pane and give command

```
p display(std::cerr)
```

c. Observe callstack, explain the 20+ long call stack before start solving

Simplification

Before solving, the input goes via a series of simplifications.

Look at

- src/smt/asserted_formulas.cpp:236

Exercise 1.3

a. Place a breakpoint there and F8 till the breakpoint.
c. Get out of function using F7 until you reach smt_context.cpp:3373
[Be careful you may overshoot!]

Internalize

- Every atom gets a Boolean variable
- Every term and gets a enode (nodes for equality reasoning)

Exercise 1.4

a. Go to debugger console in the bottom pane and give command

```
p display(std::cerr)
```

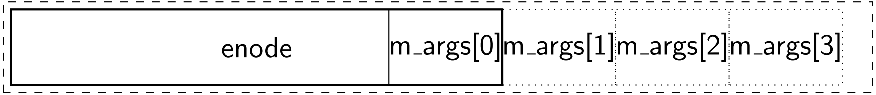
b. Identify the number of declared enodes and their types

c. Identify boolean encoder variables and their corresponding terms

Allocating enode

We need variable-sized enode to store pointer to arguments

1. Allocated a big chunk of memory
2. Declare initial part to be enode
3. Keep a pointer at the end of enode
4. Allocate extra space for the argument pointers
5. Access the rest of the space in the chunk via array access



For details look at

- src/smt/smt_enode.h:121
- src/smt/smt_enode.h:158
- src/smt/smt_enode.cpp:68

CDCL - search

- propagates
- decides
- pushes to the theory - base theory is implemented within the file

Look at at z3/src/smt/smt_context.cpp:3658

Exercise 1.5

a. Place a breakpoint there and go there

a. find the function for Boolean propagation

b. find the function for variable decisions

c. find the function for push in the theory of equality

Boolean propagation

- Uses watched literals for unit propagation

Look at
z3/src/smt/smt_context.cpp:1707
z3/src/smt/smt_context.cpp:342

Exercise 1.6

- *Find where watched literal is implemented*
- *What are the special cases depending on the type of the clauses?*
- *Where propagated atoms are passed to equality engine?*

Decision

- Maintains a priority queue

Look at at z3/src/smt/smt_context.cpp:1817

Exercise 1.7

a. Find which data structure contains the the priority queue?

b. How priority is managed?

Equality propagation

- Equivalence classes are stored as circular linked lists over endoes
- Parents of classes are “exogenously” stored at the root
- Congruence table is used to find quick matches

Look at at z3/src/smt/smt_context.cpp:492

Exercise 1.8

find the place where

a. classes are merged

b. congruence on the parents are applied

Congruence

- Iterates over parents of the looser root
- Copies the parents to the winner
- Identifies new congruences and propagate them

Look at at z3/src/smt/smt_context.cpp:675

Exercise 1.9

a. find the place where the new congruences are identified

b. Explain the mechanism

Clause learning

- Clause learning learns clauses from the conflict
- Adds new clauses in the

Look at at z3/src/smt/smt_context.cpp:3650

- Where conflicts are resolved?
- What is the state after the conflict analysis?

Topic 1.3

Problems

Problem

Exercise 1.10 (2.5 points)

Read Z3 code for maintenance of parent relation in EUF solving.
Write a short explanation of optimizations implemented to achieve efficient operations on the parent relation.
Files of interest are:

- *z3/src/smt/smt_enode.cpp // class for nodes in union-find*
- *z3/src/smt/smt_enode.h*
- *z3/src/smt/smt_cg_table.cpp // class for parent relation*
- *z3/src/smt/smt_cg_table.h*
- *z3/src/smt/smt_context.cpp // class for smt solver*
- *z3/src/smt/smt_context.h*
- *z3/src/smt/smt_internalizer.cpp:902*
- *z3/src/smt/smt_internalizer.cpp:968*

SMT solving begins at z3/src/smt/smt_context.cpp:3100
context object in smt_context.h has many display functions. Use them to print current state during

End of Lecture 1