Automated Reasoning 2018

Lecture 9: Inside the solver

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2018-08-14

Topic 9.1

Supporting tools

Install tools

Let us install a few tools.

- linux
- emacs
- git
- python
- g++
- make
- gdb
- 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

```
git clone https://github.com/Z3Prover/z3.git
rm -rf z3/buildd
mkdir -p z3/buildd
cd z3/buildd
cmake -G "Unix Makefiles" -DCMAKE_BUILD_TYPE=Debug ../
make
```

Get the input test files from the course website

emacs and gdb

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

Here, we will use emacs25 and gdb on linux.

Please choose any other IDE and debugger combination. You should be able to load the breakpoints file.

Using gdb in emacs

- Start emacs from the folder of the binary
- M-x gdb (will give you a prompt to type the binary name)
- Some commands in gdb
  - r [program arguments] // runs the binary
  - b [program location] // inserts a breakpoint
  - s // steps in the program
  - n // steps over the function calls
  - c // continue to next breakpoint
  - fin // finish current function
  - p [code] // executes the code

Topic 9.2

Engineering for CDCL(QF_EUF)

Engineering CDCL(QF_EUF)

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

Running Z3 in emacs gdb

- Download the following file *test.smt2* from the course site

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

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

(check-sat)
```

Exercise 9.1

Run z3 with the above input using the following command
(gdb) r path/to/input

Start of solving: setup_and_check

- Entry point of smt solver
- Some simplification are applied to the input before reaching here

Look at z3/src/smt/smt_context.cpp:3346

Exercise 9.2

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

b. (gdb) p display(std::cerr)

Simplification

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

Look at

- src/smt/asserted_formulas.cpp:226
- src/smt/asserted_formulas.cpp:242

Internalize

- Every atom gets a Boolean variable
- Every term and gets a enode (nodes for union-find)

Look at z3/src/smt/smt_internalizer.cpp:193

Exercise 9.3

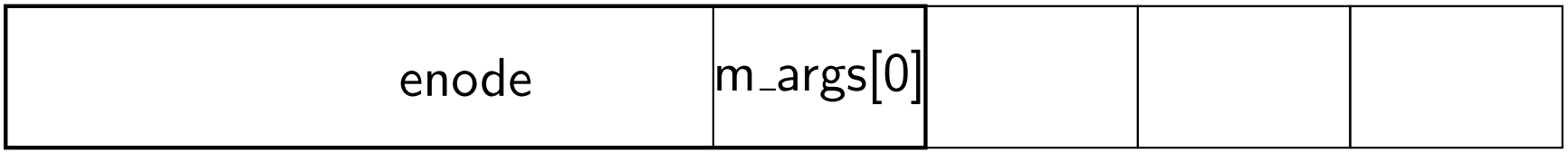
a. Find the code where enode is allocated

b. Find the enode declaration and identify fields for storing pointer to the children terms

Allocating enode

We need variable-sized enode to store pointer to arguments

- Keep a pointer at the end
- Allocate extra space for the argument pointers
- Access the rest of the space via array access



Look at

- z3/src/smt/smt_enode.h:121
- z3/src/smt/smt_enode.h:158
- z3/src/smt/smt_enode.cpp:68

After internalize

Insert a breakpoint at z3/src/smt/smt_context.cpp:3361

Exercise 9.4

Look at the current state of the solver

- *Identify the number of declared enodes and their types*
- *Identify boolean encoder variables and their corresponding terms*

CDCL - search

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

Look at at z3/src/smt/smt_context.cpp:3511

Exercise 9.5

Place a breakpoint there and find the place

a. for Boolean propagation

b. where variable is decided

c. for push in the theory of equality

Boolean propagation

- Uses watched literals for unit propagation

Look at at z3/src/smt/smt_context.cpp:1726

Exercise 9.6

- *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 9.7

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 9.8

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 9.9

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 9.3

Problems

Problem

Exercise 9.10 (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 debugging

End of Lecture 9