Comparison between SAT and SMT as a Software Analysis Engine

Presented by Yunho Kim
Provable Software Laboratory
CS Dept. KAIST
Comparison between SAT and SMT as a Software Analysis Engine

Introduction

Q: Which is better for a software analysis engine?

SAT & SMT

Bounded Model Checking

Test case Generation

Predicate Abstraction
Comparison between SAT and SMT as a Software Analysis Engine

- Arithmetic
- Bit-vectors
- Arrays
- ...

SMT
SAT
Theories

Yunho Kim@ pswlab
Arithmetic

\[ x + 2 = y \Rightarrow f(select(store(a, x, 3), y - 2)) = f(y - x + 1) \]

Array

Uninterpreted Function
C Bounded Model Checking

Yunho Kim@pswlab

Comparison between SAT and SMT as a Software Analysis Engine

1 int main(){
2   int a[2], i, x;
3   if(x==0)
4       a[i] = 0;
5   else
6       a[i+2]=1;
7   assert(a[i+1]==1);
8 }

1 guard\_1 == (x\_0 == 0)
2 a\_1 == (a\_0 WITH [i\_0:=0])
3 a\_2 == a\_0
4 a\_3 == (a\_2 WITH [2+i\_0:=1])
5 a\_4 == (guard\_1 ? a\_1 : a\_3)
6 t\_1 == (a\_4[1+i\_0]==1)

(¬x\_4 ∨ ¬x\_36) ∧
(¬x\_5 ∨ ¬x\_36) ∧
...
(¬x\_134 ∨ ¬x\_135) ∧
(x\_38 ∨ ¬x\_135) ∧
(x\_134 ∨ ¬x\_38 ∨ x\_135) ∨
...

Program

Claim

VC Generator

Constraints

CNF Generator

SAT Solver

CNF

SAT
(C.E. exists)

UNSAT
(no C.E. found)
Encoding Approach (1/2)

- Bit-level encoding
  - Every bit is represented individually

- Word-level encoding
  - E.g., unbounded integers

- Bit-vector encoding
  - Captures true semantics of hardware and software
  - Has more structures for abstraction than with bits
Comparison between SAT and SMT as a Software Analysis Engine

**Encoding Approach (2/2)**

- **Bit-level encoding**
  \[
  \neg x_4 \lor \neg x_{36} \land \\
  \neg x_5 \lor \neg x_{36} \land \\
  \ldots \\
  \neg x_{134} \lor \neg x_{135} \land \\
  x_{38} \lor \neg x_{135} \land \\
  \ldots 
  \]

- **Word-level encoding**
  \[\text{guard}_1 \iff x_0 = 0 \land \\
  a_1 = \text{store}(a, i_0, 0) \land \\
  \ldots \\
  (T_1 \iff \text{select}(a_4, 1+i_0)=1)\]

- **Bit-vector encoding**
  \[\text{guard}_1 \iff x_0 = \text{bv0}[32] \land \\
  a_1 = \text{store}(a, i_0, \text{bv0}[32]) \land \\
  \ldots \\
  (T_1 \iff \text{select}(a_4, (\text{bvadd \ bv1}[32] \ i_0)) \land \\
  =\text{bv1}[32])\]
Benchmark

- We benchmark three examples to compare each encoding scheme
  - Insertion sort (20LOC, 2-level loop)
  - Binary search (54LOC, 1-level loop)
  - Multi-sector read function in the proprietary flash device driver (157LOC, 4-level loop)

- We use four state-of-the-art SAT and SMT solvers
  - MiniSAT 1.14(integrated with CBMC)
  - Z3 (2008 SMT-competition winner of linear arithmetic category)
  - Yices
  - Boolector (2008 SMT-competition winner of bit-vector category)
Comparison between SAT and SMT as a Software Analysis Engine

Yunho Kim@ pswlab

Comparison of solving time
Insertion sort and binary search

<table>
<thead>
<tr>
<th></th>
<th>Sort_5</th>
<th>Sort_7</th>
<th>Search_5</th>
<th>Search_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-level MiniSAT</td>
<td>17.4</td>
<td>53.1</td>
<td>8.8</td>
<td>23.5</td>
</tr>
<tr>
<td>Word-level Yices</td>
<td>2.5</td>
<td>5.5</td>
<td>1.8</td>
<td>4.5</td>
</tr>
<tr>
<td>Word-level Z3</td>
<td>0.8</td>
<td>1.1</td>
<td>2.9</td>
<td>3.6</td>
</tr>
<tr>
<td>Bit-vector Yices</td>
<td>9.6</td>
<td>14.9</td>
<td>4.2</td>
<td>12.4</td>
</tr>
<tr>
<td>Bit-vector Z3</td>
<td>6.5</td>
<td>19.7</td>
<td>8.2</td>
<td>11.4</td>
</tr>
<tr>
<td>Bit-vector Boolector</td>
<td>3.8</td>
<td>17.5</td>
<td>6.6</td>
<td>13.5</td>
</tr>
</tbody>
</table>
Comparison of solving time
MSR in flash device driver

<table>
<thead>
<tr>
<th></th>
<th>MSR_4_5</th>
<th>MSR_4_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-level MiniSAT</td>
<td>8.6</td>
<td>15.8</td>
</tr>
<tr>
<td>Word-level Yices</td>
<td>85.1</td>
<td>359.7</td>
</tr>
<tr>
<td>Word-level Z3</td>
<td>16.1</td>
<td>76.3</td>
</tr>
<tr>
<td>Bit-vector Yices</td>
<td>293.4</td>
<td>270.4</td>
</tr>
<tr>
<td>Bit-vector Z3</td>
<td>141.4</td>
<td>486.4</td>
</tr>
<tr>
<td>Bit-vector Boolector</td>
<td>1196.8</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

- SAT & SMT is a hot issue in software verification

- Constraint encoding scheme can vary solving performance dramatically

- We need to investigate more efficient encoding method to improve scalability