Integration of the SMT Solver Boolector in the Framework JavaSMT and Evaluation in CPAchecker

Daniel Baier

Software and Computational Systems Lab
LMU Munich

27.11.2019
Motivation & Goal

Motivation

extending JavaSMTs solver backend

Goal

- integrate new solver
- test
- document
- evaluate
Example: Why more Solvers?

<table>
<thead>
<tr>
<th>status</th>
<th>cputime (s)</th>
<th>memory (MB)</th>
<th>status</th>
<th>cputime (s)</th>
<th>memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>522</td>
<td>696</td>
<td>false</td>
<td>11.4</td>
<td>407</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>902</td>
<td>732</td>
<td>false</td>
<td>10.3</td>
<td>318</td>
</tr>
<tr>
<td>false</td>
<td>467</td>
<td>695</td>
<td>false</td>
<td>13.7</td>
<td>389</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>902</td>
<td>803</td>
<td>false</td>
<td>9.99</td>
<td>270</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>902</td>
<td>852</td>
<td>false</td>
<td>11.9</td>
<td>398</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>902</td>
<td>692</td>
<td>false</td>
<td>13.2</td>
<td>250</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>902</td>
<td>925</td>
<td>false</td>
<td>13.4</td>
<td>387</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>902</td>
<td>799</td>
<td>false</td>
<td>10.2</td>
<td>319</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>902</td>
<td>1080</td>
<td>false</td>
<td>16.8</td>
<td>550</td>
</tr>
</tbody>
</table>
Preliminaries: SMT

Satisfiability Modulo Theories (SMT)
- SMT is a decision problem \(\rightarrow\) extension of SAT
  - SAT  UNKNOWN  UNSAT
- uses multiple theories
  - Bitvector  Integer  Float  Array  etc.
- first-order logic with equality

SMT Example:

\[
\begin{array}{|c|c|}
\hline
1 & 3 \\
\hline
& 4 \\
\hline
& 8 \\
\hline
\end{array}
\]
SMT-LIB
The Satisfiability Modulo Theories Library

- provide communication standard
- SMT-Competition
- provides benchmarking library
Preliminaries: Boolector

Bitvector specialized SMT solver

- C and Python API
- BV, QF_BV, QF_UFBV, QF_ABV and QF_AUFBV
- assumption solving
- good results in past competitions
- 4 SAT solvers
  - CaDiCaL  Lingeling  PicoSAT  MiniSat
- MIT license
Boolector code

```c
BoolectorNode *x, *y, *z, *add, *eq;

Btor *btor = boolector_new();  // New instance

x = boolector_var(btor, 8, "x");
y = boolector_var(btor, 8, "y");
z = boolector_zero(btor, "z");  // z = 0

add = boolector_add(btor, x, y);  // x + y
eq = boolector_eq(btor, add, z);  // x + y = z

boolector_assert(btor, eq);  // Assert eq

boolector_sat(btor);               // SAT check
```
Preliminaries: Boolector code

```c
BoolectorNode *add, *eq;

Btor *btor = boolector_new ();

add = boolector_add (btor,
    boolector_var (btor, 8, "x"),
    boolector_var (btor, 8, "y"));  // x + y

eq = boolector_eq (btor, add,
    boolector_zero (btor, "z"));    // x + y = z

boolector_assert (btor, eq);     // Assert

boolector_sat (btor);            // SAT check

→ How to get variable assignments?
```
Preliminaries: JavaSMT

Common API layer for various SMT solvers
- little runtime overhead
- features of solvers useable
- individual settings of solvers customizable
- type-safety
Preliminaries: JavaSMT

Common API layer for various SMT solvers
- little runtime overhead
- features of solvers useable
- individual settings of solvers customizable
- type-safety

<table>
<thead>
<tr>
<th></th>
<th>Boolector</th>
<th>Z3</th>
<th>MathSAT5</th>
<th>CVC4</th>
<th>Princess</th>
<th>SMTInterpol</th>
<th>Yices</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaSMT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>soon</td>
</tr>
<tr>
<td>ScalaSMT</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>MetaSMT</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>PySMT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Implementation: JNI Wrapper

- wrapper created by SWIG\(^1\)
- refined by hand
- added custom methods
- compiled as shared library

\footnote{Simplified Wrapper and Interface Generator}

\(\text{_MINI\textsc{Sat} cannot be compiled into a shared library}\)
Implementation: JavaSMT

Good

- LogManager
- automatic memory cleanup
- assumption-solving
- all native options accessible
- variables cache

Problems with Boolector

- bitvectors width 1 $\rightarrow$ booleans
- incremental mode for stack $\rightarrow$ Cadical not usable
- no ShutdownManager
- no parsing
Implementation: JavaSMT

More Problems with Boolector
- cannot access all variables $\rightarrow$ no visitor$^2$

Incomplete Model
- no visitor $\rightarrow$ cannot access all assignments $\rightarrow$ no toList()$^3$

No Bitvector Quantifier
- quantifier need a separate variable in Boolector
  - boolector_param() instead of boolector_var()
- no visitor $\rightarrow$ not able to change variables for quantifier

$^2$Class used to access all variables, constants and formulas
$^3$Lists all assignments of all variables
Implementation: Tests

Boolector Unique Characteristics

- no bitvectors width 1
- no integer theorie → requireIntegers()

Many Tests Need

- parsing
- visitor
- toList()
→ a lot of exceptions at the moment
Evaluation: Setup

Evaluation in CPAchecker with Benchexec

Contrary to SV-COMP, results don’t matter in this context, only the performance of the solvers is measured

- slightly modified SV-COMP standards
  - 900s timelimit 15GB memory 2 CPU cores
- subset of SV-benchmarks
- bounded model checking (BMC)
- 1 run with 1 and 10 loop iterations each
- via VerifierCloud on Apollon cluster
  - Intel Xeon E3-1230 v5 @ 3.40GHz with 33GB memory
- only bitvector capable solvers
  - Boolector CVC4 MathSAT5 Z3
Evaluation: Setup

Due to Boolector only having bitvectors, no complete model and no visitor, the CPAchecker had to be modified and needs some additional options to run properly.

Options because of Restrictions

- encoding as bitvector
  \[ \text{floats} \rightarrow \text{integers} \rightarrow \text{bitvectors} \]
- eager creation of formula encoding
- no pointer aliasing
- no output
## Evaluation: Results

<table>
<thead>
<tr>
<th>k</th>
<th>Solver</th>
<th>Correct Results</th>
<th>Incorrect Results</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>Boolector</td>
<td>729</td>
<td>415</td>
<td>314</td>
</tr>
<tr>
<td></td>
<td>CVC4</td>
<td>726</td>
<td>419</td>
<td>307</td>
</tr>
<tr>
<td></td>
<td>MathSAT5</td>
<td>738</td>
<td>419</td>
<td>319</td>
</tr>
<tr>
<td></td>
<td>Z3</td>
<td>728</td>
<td>419</td>
<td>309</td>
</tr>
<tr>
<td>10</td>
<td>Boolector</td>
<td>1587</td>
<td>720</td>
<td>867</td>
</tr>
<tr>
<td></td>
<td>CVC4</td>
<td>1252</td>
<td>540</td>
<td>712</td>
</tr>
<tr>
<td></td>
<td>MathSAT5</td>
<td>1446</td>
<td>553</td>
<td>893</td>
</tr>
<tr>
<td></td>
<td>Z3</td>
<td>1300</td>
<td>543</td>
<td>757</td>
</tr>
</tbody>
</table>
Evaluation: Memory Usage ($k=10$)
(All Correct + Incorrect + Unknown Results)

$n$-th result sorted by lowest memory usage

<table>
<thead>
<tr>
<th>Tool</th>
<th>Memory Usage [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolector</td>
<td>120</td>
</tr>
<tr>
<td>CVC4</td>
<td>250</td>
</tr>
<tr>
<td>MathSAT5</td>
<td>300</td>
</tr>
<tr>
<td>Z3</td>
<td>500</td>
</tr>
</tbody>
</table>

Daniel Baier (SoSy-Lab)
Evaluation: CPU time (k=10)
(All Correct + Incorrect + Unknown Results)
Conclusion

Boolector was integrated into JavaSMT

Summary

- 2 of 4 SAT solvers available
- no quantifier
- missing Boolector methods → missing JavaSMT features

But

- working implementation
- working tests
- good results compared to other SMT solvers
Conclusion

Boolector was integrated into JavaSMT

Summary

- 2 of 4 SAT solvers available
- no quantifier
- missing Boolector methods $\rightarrow$ missing JavaSMT features

But

- working implementation
- working tests
- good results compared to other SMT solvers

$\Rightarrow$ Incomplete but working implementation
$\Rightarrow$ Good results in evaluation
Future Work

Good News: Missing Methods Promised to be Added

Whats Next?
- integrate visitor
- integrate toList()
- integrate quantifier

More Good News: Updated Version of Boolector soon

⇒ Re-evaluate
Sources

electrofriends.com/articles/jni/jni-part1-java-native-interface
github.com/Boolector/boolector
github.com/Z3Prover/z3
mathsat.fbk.eu/index.html
github.com/CVC4/CVC4
www.philipp.ruemmer.org/princess.shtml
github.com/ultimate-pa/smtinterpol
github.com/SRI-CSL/yices2
github.com/sosy-lab/java-smt
github.com/regb/scala-smtlib
github.com/agra-uni-bremen/metaSMT
github.com/pysmt/pysmt
github.com/arminbiere/cadical
Sources

github.com/niklasso/minisat
fmv.jku.at/picosat/
github.com/arminbiere/lingeling
github.com/sosy-lab/sv-benchmarks/
github.com/sosy-lab/benchexec
sv-comp.sosy-lab.org/2020/
svn.sosy-lab.org/software/cpachecker/