Bachelor Thesis

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

extending JavaSMTs solver backend

Goal

- integrate new solver
- test
- document
- evaluate

Example: Why more Solvers?

CPAchecker 2019-11-14 05:52:12 CET k1-cvc4			CPAchecker 2019-11-14 05:52:12 CET k1-boolector		
status	cputime (s)	memory (MB)	status	cputime (s)	memory (MB)
Show all	Min:Max	Min:Max	correct	Min:Max	Min:Max
false	522	696	false	11.4	407
TIMEOUT	902	732	false	10.3	318
false	467	695	false	13.7	389
TIMEOUT	902	803	false	9.99	270
TIMEOUT	902	852	false	11.9	398
TIMEOUT	902	692	false	13.2	250
TIMEOUT	902	925	false	13.4	387
TIMEOUT	902	799	false	10.2	319
TIMEOUT	902	1080	false	16.8	550

Preliminaries: SMT

Satisfiability Modulo Theories (SMT)

- SMT is a decision problem → extension of SAT
  SAT UNKNOWN UNSAT
- uses multiple theories
  Bitvector *Integer* *Float* *Array* etc.
- first-order logic with equality

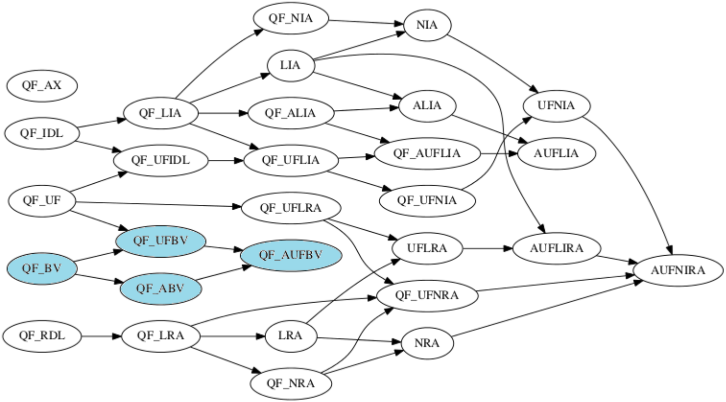
SMT Example:

1		3
		4
	8	

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

Preliminaries: Boolector code

```
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

```
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

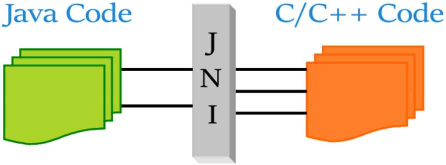
- little runtime overhead
- features of solvers useable
- individual settings of solvers customizable
- type-safety

	Boolector	Z3	MathSAT5	CVC4	Princess	SMTInterpol	Yices
JavaSMT	yes	yes	yes	yes	yes	yes	soon
ScalaSMT	no	yes	yes	yes	no	yes	yes
MetaSMT	yes	yes	no	yes	no	no	no
PySMT	yes	yes	yes	yes	no	no	yes

Implementation: JNI Wrapper

- wrapper created by SWIG[1]
- refined by hand
- added custom methods
- compiled as shared library

↯ MiniSat cannot be compiled into a shared library



[1] Simplified Wrapper and Interface Generator

Implementation: JavaSMT

Good

- LogManager
- automatic memory cleanup
- assumption-solving
- all native options accessable
- 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 → no visitor[2]

Incomplete Model

- no visitor → cannot access all assignments → no toList()[3]

No Bitvector Quantifier

- quantifier need a seperate variable in Boolector

  boolector_param() instead of boolector_var()
- no visitor → 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

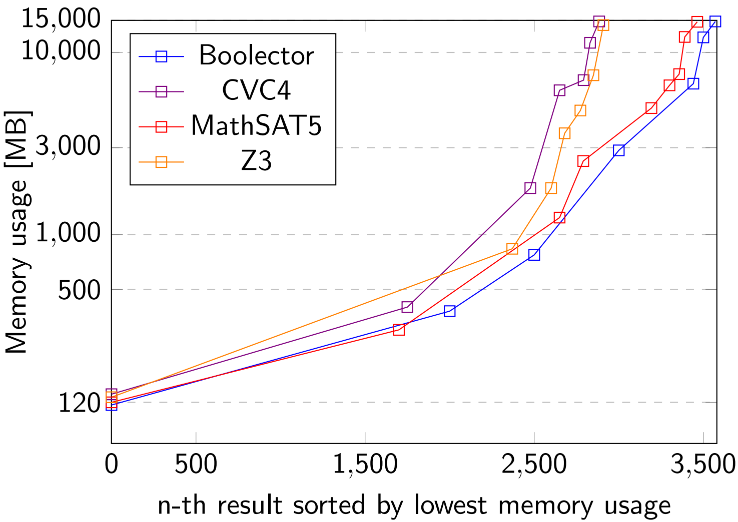
Options because of Restrictions

- encoding as bitvector
  floats $\rightarrow$ *integers* $\rightarrow$ *bitvectors*
- eager creation of formula encoding
- no pointer aliasing
- no output

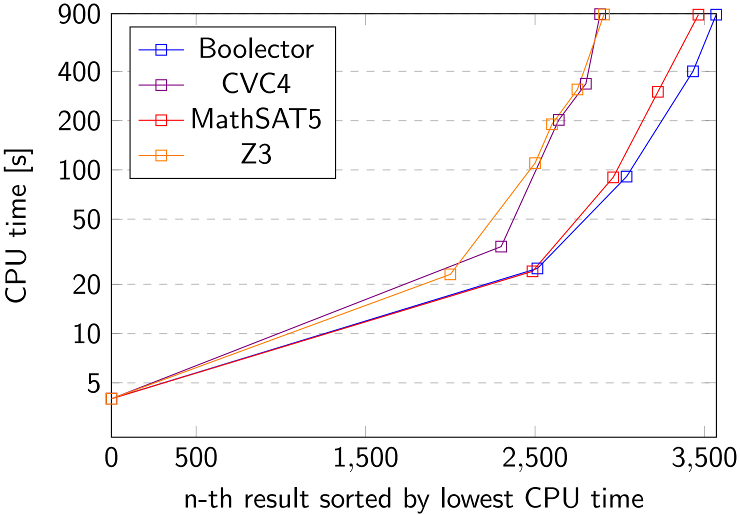
Evaluation: Results

k	Solver	Correct Results			Incorrect Results			Unknown
		total	true	false	total	true	false	
1	Boolector	729	415	314	70	0	70	4476
	CVC4	726	**419**	307	72	0	72	4525
	MathSAT5	**738**	**419**	**319**	**73**	0	**73**	**4533**
	Z3	728	**419**	309	64	0	64	4510
10	Boolector	**1587**	**720**	867	**246**	**120**	**126**	1738
	CVC4	1252	540	712	118	1	117	1515
	MathSAT5	1446	553	**893**	127	1	**126**	**1892**
	Z3	1300	543	757	119	1	118	1490

Evaluation: Memory Usage (k=10)
(All Correct + Incorrect + Unknown Results)



Evaluation: CPUtime (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 → missing JavaSMT features

But

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

⇒ **Incomplete but working implementation**
⇒ **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

$\Rightarrow$ 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/