JavaSMT 5.0 in CPAchecker

Daniel Baier

September 9, 2024

LMU Munich, SoSy-Lab

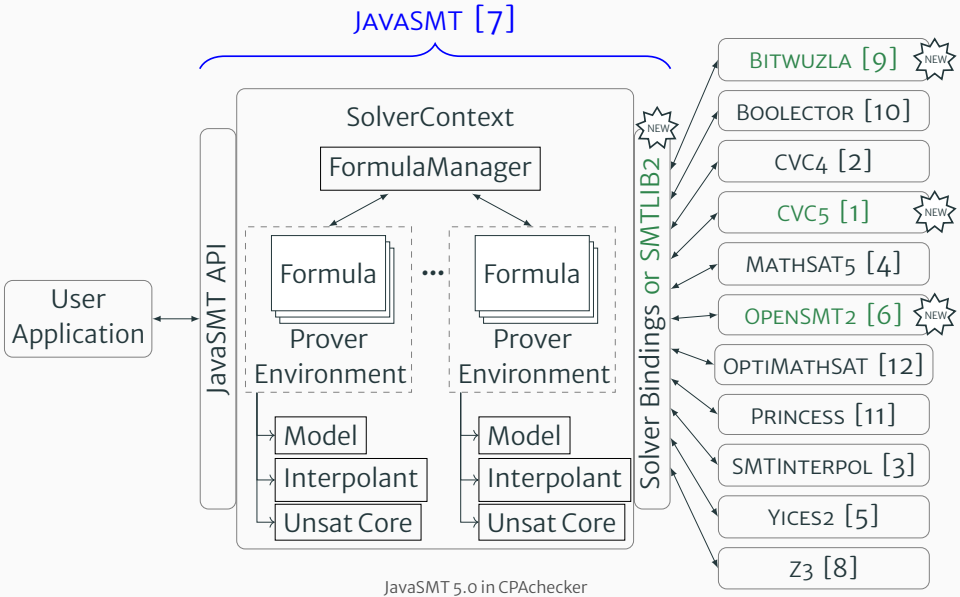
What is JavaSMT?

What is JavaSMT?

- Common API layer for SMT solvers
- SMT backend of CPAchecker
- Low overhead and automatic memory cleanup
- Java based and usable on Unix, Windows* and MacOS*
- Visit us here: `https://github.com/sosy-lab/java-smt`

* Depending on the solver.

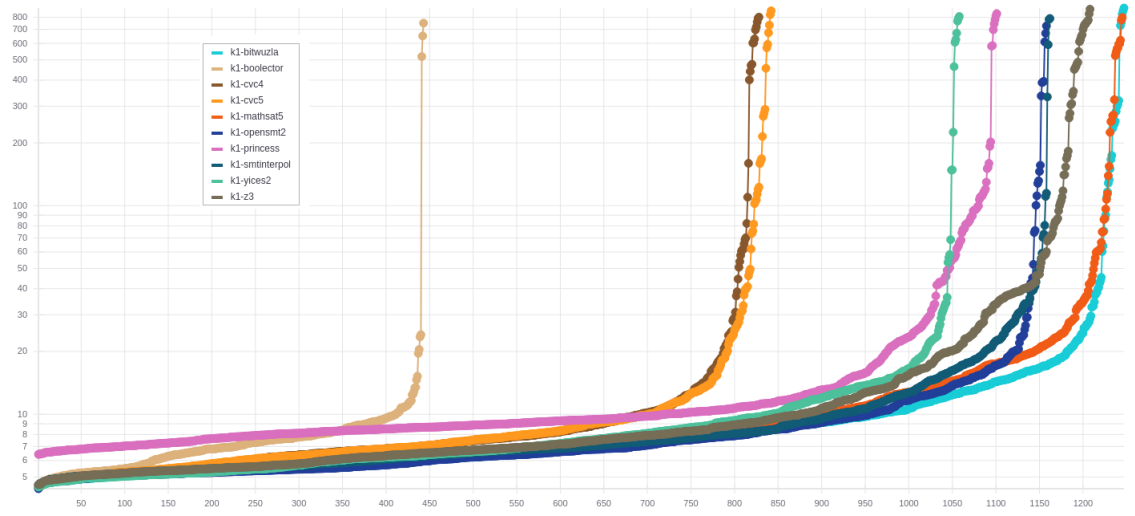
JavaSMT: New SMT Solvers



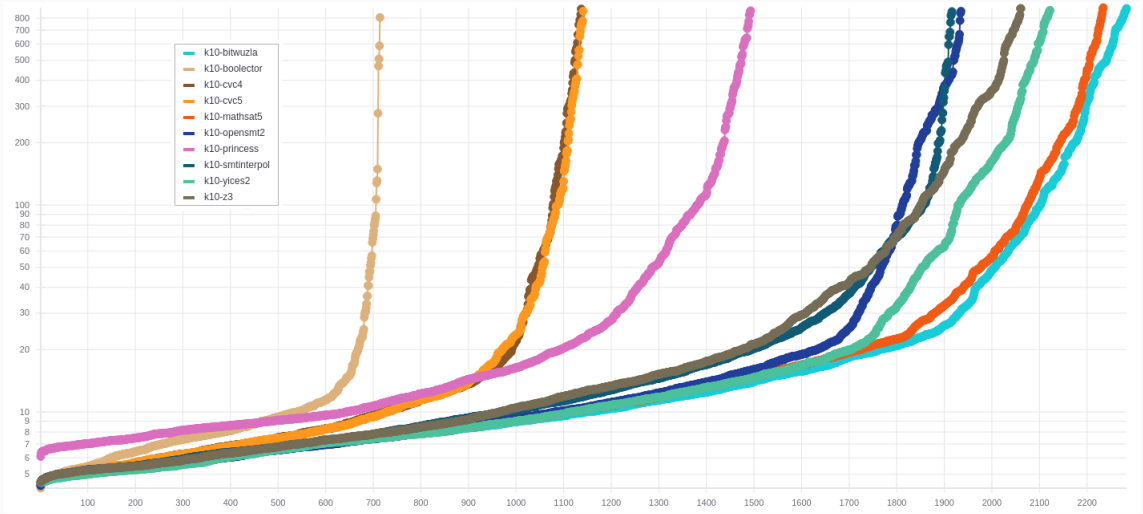
Performance of the New SMT Solvers

- Ran CPAchecker using BMC
- SV-COMP24 ReachSafety task-set
- SV-COMP24 settings, but 2 cores only
- Each solver run with 1 and 10 loop unrollings
- This is no case-study!
- Default encoding uses Bitvector theory for Integers
- Princess, SMTInterpol, OpenSMT2 do not support Bitectors and use other encodings

Performance of the New SMT Solvers using BMC (1 unrolling)



Performance of the New SMT Solvers using BMC (10 unrollings)



Normal conversation

- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.

Normal conversation

- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.
- Daniel: Sure, what is it?

Normal conversation

- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.
- Daniel: Sure, what is it?
- Coworker: There is this weird error message:
  Formula "and" expected argument of type Bool but received Bool

Normal conversation

- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.
- Daniel: Sure, what is it?
- Coworker: There is this weird error message:
  Formula "and" expected argument of type Bool but received Bool
- Daniel: Did you use a formula on an distinct context from the one you created it in?

Normal conversation

- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.
- Daniel: Sure, what is it?
- Coworker: There is this weird error message:
  Formula "and" expected argument of type Bool but received Bool
- Daniel: Did you use a formula on an distinct context from the one you created it in?
- Coworker: No....

Normal conversation

- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.
- Daniel: Sure, what is it?
- Coworker: There is this weird error message:
  Formula "and" expected argument of type Bool but received Bool
- Daniel: Did you use a formula on an distinct context from the one you created it in?
- Coworker: No.... i think ....

Normal conversation

- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.
- Daniel: Sure, what is it?
- Coworker: There is this weird error message:
  Formula "and" expected argument of type Bool but received Bool
- Daniel: Did you use a formula on an distinct context from the one you created it in?
- Coworker: No.... i think .... maybe...

Normal conversation

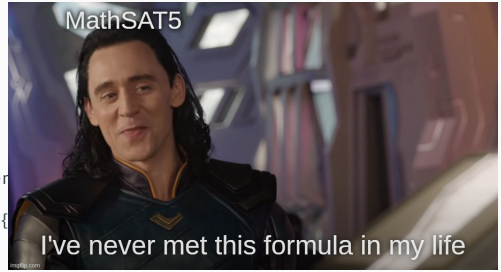
- Coworker: Daniel, do you have a moment? I have a problem with JavaSMT.
- Daniel: Sure, what is it?
- Coworker: There is this weird error message:
  Formula "and" expected argument of type Bool but received Bool
- Daniel: Did you use a formula on an distinct context from the one you created it in?
- Coworker: No.... i think .... maybe...

Example

```
Configuration config = Configuration.defaultConfiguration();

SolverContext ctx =
    SolverContextFactory.createSolverContext(config, ..., Solvers.MATHSAT5);

FormulaManager mgr = ctx.getFormulaManager();

BooleanFormulaManager bMgr = mgr.getBooleanFormulaManager();
BooleanFormula varA = bMgr.makeVariable("a");
// a AND not a
BooleanFormula formula = bMgr.and(varA, bMgr.not(varA));

SolverContext ctxTwo =
    SolverContextFactory.createSolverContext(config, ..., Solvers.MATHSAT5);

try (ProverEnvironment prover = ctxTwo.newProverEnvironment()) {
        prover.addConstraint(formula);
        boolean isUnsat = prover.isUnsat(); // Exception:
                                            // Formula "and" expected argument of type Bool but received Bool
}
```

Example

```
Configuration config = Configuration.defaultConfiguration();

SolverContext ctx =
    SolverContextFactory.createSolverContext(config, ..., Solvers.MATHSAT5);

FormulaManager mgr = ctx.getFormulaManager();

BooleanFormulaManager bMgr = mgr.getBooleanFormulaManager();
BooleanFormula varA = bMgr.makeVariable("a");
// a AND not a
BooleanFormula formula = bMgr.and(varA, bMgr.not(varA));

SolverContext ctxTwo =
    SolverContextFactory.createSolverContext(config, ..., Solver

try (ProverEnvironment prover = ctxTwo.newProverEnvironment()) {
        prover.addConstraint(formula);
        boolean isUnsat = prover.isUnsat(); // Exception:
                                            // Formula "and" expected argument of type Bool but received Bool
}
```



Common Errors that Repeatedly Happen

- Using formulas on distinct contexts/threads etc.
- Using the same solver on distinct threads

→ Different solvers support different things

→ Hard to know what solver supports what

→ Solution: automatic checks via a "Debug-Mode"

Example for Debug-Mode

```
Configuration config =
    Configuration.builder().setOption("solver.debugMode", "true").build();

SolverContext ctx =
    SolverContextFactory.createSolverContext(config, ..., Solvers.MATHSAT5);

FormulaManager mgr = ctx.getFormulaManager();

BooleanFormulaManager bMgr = mgr.getBooleanFormulaManager();
BooleanFormula varA = bMgr.makeVariable("a");
// a AND not a
BooleanFormula formula = bMgr.and(varA, bMgr.not(varA));

SolverContext ctxTwo =
    SolverContextFactory.createSolverContext(config, ..., Solvers.MATHSAT5);

try (ProverEnvironment prover = ctxTwo.newProverEnvironment()) {
        prover.addConstraint(formula);  // An exception is thrown that explains the issue
        boolean isUnsat = prover.isUnsat();
}
```

Future Work

- Add support for SMT-LIB2 only SMT solvers
- Support solver independent Interpolation
- Support solver independent Quantifier Elimination

References i

[1] Barbosa, H., Barrett, C., Brain, M., Kremer, G., Lachnitt, H., Mann, M., Mohamed, A., Mohamed, M., Niemetz, A., Nötzli, A., et al.: cvc5: A versatile and industrial-strength smt solver. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 415–442. Springer (2022). https://doi.org/10.1007/978-3-030-99524-9_24

[2] Barrett, C.W., Conway, C.L., Deters, M., Hadarean, L., Jovanovic, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Proc. CAV. pp. 171–177. LNCS 6806, Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_14

References ii

[3] Christ, J., Hoenicke, J., Nutz, A.: SMTInterpol: An interpolating SMT solver. In: Proc. SPIN. pp. 248–254. LNCS 7385, Springer (2012). https://doi.org/10.1007/978-3-642-31759-0_19

[4] Cimatti, A., Griggio, A., Schaafsma, B.J., Sebastiani, R.: The MathSAT5 SMT solver. In: Proc. TACAS. pp. 93–107. LNCS 7795, Springer (2013). https://doi.org/10.1007/978-3-642-36742-7_7

[5] Dutertre, B.: Yices 2.2. In: Proc. CAV. pp. 737–744. LNCS 8559, Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_49

References iii

[6] Hyvärinen, A.E.J., Marescotti, M., Alt, L., Sharygina, N.: OpenSMT2: An SMT solver for multi-core and cloud computing. In: Proc. SAT. pp. 547–553. LNCS 9710, Springer (2016). https://doi.org/10.1007/978-3-319-40970-2_35

[7] Karpenkov, E.G., Friedberger, K., Beyer, D.: JavaSMT: A unified interface for SMT solvers in Java. In: Proc. VSTTE. pp. 139–148. LNCS 9971, Springer (2016). https://doi.org/10.1007/978-3-319-48869-1_11

[8] de Moura, L.M., Bjørner, N.: Z3: An efficient SMT solver. In: Proc. TACAS. pp. 337–340. LNCS 4963, Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24

References iv

[9] Niemetz, A., Preiner, M.: BITWUZLA at the SMT-COMP 2020. arXiv/CoRR **2006**(01621) (June 2020), https://arxiv.org/abs/2006.01621

[10] Niemetz, A., Preiner, M., Biere, A.: BOOLECTOR 2.0. J. Satisf. Boolean Model. Comput. **9**(1), 53–58 (2014). https://doi.org/10.3233/sat190101, https://doi.org/10.3233/sat190101

[11] Rümmer, P.: A constraint sequent calculus for first-order logic with linear integer arithmetic. In: Proc. LPAR. pp. 274–289. LNCS 5330, Springer (2008). https://doi.org/10.1007/978-3-540-89439-1_20

References v

[12] Sebastiani, R., Trentin, P.: OPTIMATHSAT: A tool for optimization modulo theories. In: Proc. CAV. pp. 447–454. LNCS 9206, Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_27

Features of the New SMT Solvers

CVC5

- Successor of CVC4
- Integer, Rational, String, Separation-Logic, Bitvector, Floating-Point, Array and Quantifier support
- Features: Interpolation, Quantifier Elimination, Unsat-Core
- No support for concurrent usage!

Features of the New SMT Solvers

Bitwuzla

- Successor of Boolector
- Bitvector, Floating-Point, Array and Quantifier support
- Features: Quantifier Elimination, Assumption Solving, Unsat-Core
- No support for concurrent usage!

Features of the New SMT Solvers

OpenSMT2

- Integer, Rational and Array support
- Features: Interpolation (only QF_UF, QF_LIA and QF_LRA logics!)
- User needs to specify a logic beforehand!