Finding Conflicting Instances of Quantified Formulas in SMT

Andrew Reynolds
Cesare Tinelli
Leonardo De Moura
July 18, 2014
Outline of Talk

• SMT solvers:
  – Efficient methods for ground constraints
  – Heuristic methods for quantified formulas
  \( \Rightarrow \) Can we reduce dependency on heuristic methods?

• New method for quantifiers in SMT
  – Finds conflicting instances of quantified formulas

• Experimental results

• Summary and Future Work
Satisfiability Modulo Theories (SMT)

- **SMT solvers**
  - Are efficient for problems over ground constraints $G$
  - Determine the satisfiability of $G$ using a combination of:
    - Off-the-shelf SAT solver
    - Efficient ground decision procedures, e.g.
      - Uninterpreted Functions
      - Linear arithmetic
      - Arrays
      - Datatypes
      - ...

- **Used in many applications:**
  - Software/hardware verification
  - Scheduling and Planning
  - Automated Theorem Proving

[f(3) \neq f(c) \]
[c=2 \lor c+1 \leq 0 \]
[a+1 = \text{read}(A,b) \]
[tail(l_1) = \text{cons}(a,l_2) \]
DPLL(T)-Based SMT Solver

\[ f(a) = 5 \lor f(b) = f(c) \]
\[ f(a) \geq 10 \lor \text{read}(B, 5) \leq f(c) \]
DPLL(T)-Based SMT Solver

SAT Solver

Ground Theory Solvers

\[ f(a) = 5 \lor f(b) = f(c) \]
\[ f(a) \geq 10 \lor \text{read}(B, 5) \leq f(c) \]

M “context”
DPLL(T)-Based SMT Solver

\[ f(a) = 5 \lor f(b) = f(c) \]
\[ f(a) \geq 10 \lor \text{read}(B, 5) \leq f(c) \]

\[ f(a) = 5 \]
\[ f(a) \geq 10 \]

SAT Solver

Ground Theory Solvers

T-consistent

T-inconsistent

UNSAT
SMT + Quantified Formulas

• SMT solvers have **limited support** for:
  – First-order universally quantified formulas $\mathcal{Q}$

\[
\begin{align*}
  f(a) &= 5 \lor f(b) = f(c) \\
  f(a) \geq 10 \lor \text{read}(B, 5) &\leq f(c) \\
  \forall x. f(x) &< 0
\end{align*}
\]

• Used in an increasing number of applications, for:
  – Defining axioms for symbols not supported natively
  – Encoding frame axioms, transition systems, ...
  – Universally quantified conjectures

• When universally quantified formulas $\mathcal{Q}$ are present, problem is generally **undecidable**
  – Approaches for $G \cup \mathcal{Q}$ in SMT are usually **heuristic**
SMT Solver + Quantified Formulas

SAT Solver

Ground Theory Solvers

Quantifiers Module

\[ f(a) = 5 \lor f(b) = f(c) \]
\[ f(a) \geq 10 \lor \text{read}(B, 5) \leq f(c) \]

\[ \forall x. f(x) < 0 \]
SMT Solver + Quantified Formulas

- Find (T-consistent) context M
SMT Solver + Quantified Formulas

- We must answer: “is $M \cup Q$ consistent?”
  - Problem is generally **undecidable**
Quantifier Instantiation

- **Instantiation-based** approaches:
  - Add instances of quantified formulas, based on some strategy
  - E.g. based on patterns (known as “E-matching”)

\[
\begin{align*}
\forall x. f(x) < 0 \\
f(a) = 5 \lor f(b) = f(c) \\
f(a) \geq 10 \lor \text{read( B, 5 ) } \leq f(c) \\
f(a) \geq 10 \\
f(b) = f(c) \\
\end{align*}
\]
Instantiation-Based Approaches

• Complete approaches:
  – E.g. Complete instantiation, local theory extensions, finite model finding, Inst-Gen
    • Cons: only work for limited fragments

• General approaches:
  – Heuristic E-matching
    • Cons: only for UNSAT, highly heuristic, often inefficient
Motivation

• In this talk: new method for quantified formulas
  – Goals:
    • Reduce dependency on heuristic methods
    • Applicable to arbitrary quantified formulas
  – Not goals:
    • Completeness (thus, focus only on UNSAT)
• If \( M \) is inconsistent according to ground theory,
SAT Solver reports a single conflict clause — typically, can be determined efficiently.

\[ \neg f(a) \geq 10 \lor \neg f(a) = 5 \]
Quantifiers : Heuristc Instantiation?

- The decision problem for $M \cup Q$ is undecidable,

\[ f(a) \geq 10 \]
\[ f(c) = f(b) \]
\[ \forall x. f(x) < 0 \]

M is T-consistent
Quantifiers : Heuristic Instantiation?

\[ \forall x. f(x) < 0 \]

- Add a potentially large set of instances, heuristically
  - This can overload the ground solver
Conflicting Instances

⇒ Can we make the quantifiers module behave more like a theory solver?

• Idea: find cases when $M \cup Q$ is UNSAT:
  – Find grounding substitution $\sigma$
    • Such that $M \models T \land \neg Q\sigma$

• $Q\sigma$ is a conflicting instance
• First, determine if a conflicting instance exists
  – If not, resort to heuristic instantiation
Limit of Approach

- **Caveat**: No complete method will determine whether a conflicting instance exists for \((M, Q)\)
- Thus, our approach:
  1. Uses an incomplete procedure to determine a conflicting instance for \((M, Q)\)
  2. If not, resort to E-matching for \((M, Q)\)

⇒ *In practice, Step 1 succeeds for a majority of* \((M, Q)\)
E-matching vs Conflicting Instances

In example, $g(h(x))$ matches ground term $g(b)$

- That is:
  - $M \models_T g(b) = g(h(x))\sigma$, for $\sigma = \{x \rightarrow a\}$

$\Rightarrow E$-matching for $(M, Q)$ returns $\sigma$
E-matching vs Conflicting Instances

- In this example, for \( \sigma = \{ x \rightarrow a \} \):

1. Ground terms match each sub-term from \( Q \)
   - \( M \models_T g(b) = g(h(x)) \sigma \)
   - \( M \models_T f(a) = f(x) \sigma \)

2. ...and the body of \( Q \) is falsified:
   - \( M \models_T f(x) \neq g(h(x)) \sigma \)

\( \Rightarrow M \cup Q \sigma \) is UNSAT
E-matching vs Conflicting Instances

In this example, for $\sigma = \{ x \rightarrow a \}$:

1. Ground terms match each sub-term from $Q$:
   - $M \models_{T} g(b) = g(h(x))\sigma$
   - $M \models_{T} f(a) = f(x)\sigma$

2. ...and the body of $Q$ is falsified:
   - $M \models_{T} f(x) \neq g(h(x))\sigma$

$\therefore M \cup Q\sigma$ is UNSAT

In paper, limit $T$ to EUF
E-matching vs Conflicting Instances

- Conflicting Instances
  - Consider flat form of $Q$:

  $\forall x. f(x) = g(h(x))$

  - Matching constraints $\mu$
  - Flattened body $\Psi$

- Conflicting substitution $\sigma$ for $(M, Q)$ is such that:
  - $M$ entails $\mu \sigma$
  - $M$ entails $\neg \Psi \sigma$
Equality-Inducing Instances

What if we relax constraint 2?

- Modified example, for $\sigma = \{ x \rightarrow a \}$:
  1. Ground terms match each sub-term from $Q$
     - $\models_T g(b)=g(h(x))\sigma$
     - $\models_T f(a)=f(x)\sigma$
  2. ...but the body of $Q$ is not falsified:
     - $\not\models_T f(x)\neq g(h(x))\sigma$
Equality-Inducing Instances

• Still, it may be useful to add the instance $Q \{ x \mapsto a \}$
  – In this example, $Q \{ x \mapsto a \}$ entails $g(b) = f(a)$

$\Rightarrow \{ x \mapsto a \}$ is an equality-inducing substitution

• Mimics T-propagation done by theory solvers
Instantiation Strategy

**InstantiationRound**\((Q, M)\)

1. Return a (single) **conflicting** instance for \((Q, M)\)
2. Return a set of **equality-inducing** instances for \((Q, M)\)
3. Return instances based on **E-matching** for \((Q, M)\)

- **Three configurations:**
  - **cvc4**: step (3)
  - **cvc4+c**: steps (1), (3)
  - **cvc4+ci**: steps (1),(2),(3)
Experimental Results

- **Implemented** techniques in SMT solver **CVC4**
- UNSAT benchmarks from:
  - TPTP
  - Isabelle
  - SMT Lib
- **Solvers:**
  - **cvc3, z3**
  - 3 configurations: **cvc4, cvc4+c, cvc4+ci**
### UNSAT Benchmarks Solved

<table>
<thead>
<tr>
<th></th>
<th>cvc3</th>
<th>z3</th>
<th>cvc4</th>
<th>cvc4+c</th>
<th>cvc4+ci</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPTP</td>
<td>5234</td>
<td>6268</td>
<td>6100</td>
<td>6413</td>
<td>6616</td>
</tr>
<tr>
<td>Isabelle</td>
<td>3827</td>
<td>3506</td>
<td>3858</td>
<td>3983</td>
<td>4082</td>
</tr>
<tr>
<td>SMTLIB</td>
<td>3407</td>
<td>3983</td>
<td>3680</td>
<td>3721</td>
<td>3747</td>
</tr>
<tr>
<td>Total</td>
<td>12468</td>
<td>13757</td>
<td>13638</td>
<td>14117</td>
<td>14445</td>
</tr>
</tbody>
</table>

- Configuration cvc4+ci solves the most (14,445)
  - Against cvc4 : 1,049 vs 235 (+807)
  - Against z3: 1,998 vs 1,310 (+688)
  - 359 that no implementation of E-matching (cvc3, z3, cvc4) can solve
# Instantiations for Solved Benchmarks

<table>
<thead>
<tr>
<th></th>
<th>TPTP</th>
<th>Isabelle</th>
<th>SMT lib</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Inst</td>
<td>Solved</td>
</tr>
<tr>
<td>cvc3</td>
<td>5245</td>
<td>627.0M</td>
<td>3827</td>
</tr>
<tr>
<td>z3</td>
<td>6269</td>
<td>613.5M</td>
<td>3506</td>
</tr>
<tr>
<td>cvc4</td>
<td>6100</td>
<td>879.0M</td>
<td>3858</td>
</tr>
<tr>
<td>cvc4+c</td>
<td>6413</td>
<td>190.8M</td>
<td>3983</td>
</tr>
<tr>
<td>cvc4+ci</td>
<td>6616</td>
<td>150.9M</td>
<td>4082</td>
</tr>
</tbody>
</table>

- cvc4+ci
  - Solves the most benchmarks for TPTP and Isabelle
  - Requires almost an order of magnitude fewer instantiations

- Improvements less noticeable on SMT LIB
  - Due to encodings that make heavy use of theory symbols
    - Method for finding conflicting instances is more incomplete
### Instances Produced

<table>
<thead>
<tr>
<th></th>
<th>IR</th>
<th>E-matching</th>
<th>Conflicting</th>
<th>C-Inducing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IR</td>
<td>#</td>
<td>IR</td>
</tr>
<tr>
<td>smtlib</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cvc4</td>
<td>14032</td>
<td>100.0%</td>
<td>60.7M</td>
<td></td>
</tr>
<tr>
<td>cvc4+c</td>
<td>51696</td>
<td>24.3%</td>
<td>41.0M</td>
<td>75.7%</td>
</tr>
<tr>
<td>cvc4+ci</td>
<td>58003</td>
<td>20.0%</td>
<td>32.3M</td>
<td>71.6%</td>
</tr>
<tr>
<td>TPTP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cvc4</td>
<td>71634</td>
<td>100.0%</td>
<td>879.0M</td>
<td>78.3%</td>
</tr>
<tr>
<td>cvc4+c</td>
<td>201990</td>
<td>21.7%</td>
<td>190.1M</td>
<td>76.4%</td>
</tr>
<tr>
<td>cvc4+ci</td>
<td>208970</td>
<td>20.3%</td>
<td>150.4M</td>
<td></td>
</tr>
<tr>
<td>Isabelle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cvc4</td>
<td>6969</td>
<td>100.0%</td>
<td>119.0M</td>
<td>71.1%</td>
</tr>
<tr>
<td>cvc4+c</td>
<td>18160</td>
<td>28.9%</td>
<td>54.0M</td>
<td></td>
</tr>
<tr>
<td>cvc4+ci</td>
<td>21756</td>
<td>22.4%</td>
<td>28.2M</td>
<td>64.0%</td>
</tr>
</tbody>
</table>

- **Conflicting instances found on ~75% of IR**
- **cvc4+ci**: 
  - Requires **3.1x** more instantiation rounds w.r.t. cvc4
  - Calls E-matching **1.5x** fewer times overall
    - As a result, adds **5x** fewer instantiations
Details on Solved Problems

• For the 30,081 benchmarks we considered:
  – cvc4+ci solves more (14,445) than any other
  – 359 are solved *uniquely* by cvc4+c or cvc4+ci
    • Techniques *increase precision* of SMT solver
  – cvc4+ci does not use E-matching 21% of the time
    • 94 benchmarks unsolved by E-matching implementations
    • Techniques *reduce dependency* on heuristic instantiation
Competitions : CASC J7

• Partly due to techniques:
  – Won TFA division
  – Finished only behind Vampire/E(s) in FOF division

<table>
<thead>
<tr>
<th>Typed First-order Theorems +<em>/</em>-</th>
<th>CVC4 1.4-TFA</th>
<th>Princess 140704</th>
<th>SPASS+T 2.2.19</th>
<th>SPASS+T 2.2.20</th>
<th>Beagle 0.9</th>
<th>Zipperpos 0.4-TFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved</td>
<td>179.200</td>
<td>176.200</td>
<td>173.200</td>
<td>173.200</td>
<td>173.200</td>
<td>80.200</td>
</tr>
<tr>
<td>Av. CPU Time</td>
<td>4.47</td>
<td>11.81</td>
<td>3.44</td>
<td>3.57</td>
<td>5.49</td>
<td>6.57</td>
</tr>
<tr>
<td>Solutions</td>
<td>0.200</td>
<td>0.200</td>
<td>173.200</td>
<td>173.200</td>
<td>0.200</td>
<td>80.200</td>
</tr>
<tr>
<td>μEfficiency</td>
<td>797</td>
<td>307</td>
<td>402</td>
<td>402</td>
<td>623</td>
<td>313</td>
</tr>
<tr>
<td>SOTAC</td>
<td>0.22</td>
<td>0.21</td>
<td>0.19</td>
<td>0.19</td>
<td>0.20</td>
<td>0.27</td>
</tr>
<tr>
<td>Core Usage</td>
<td>1.30</td>
<td>1.19</td>
<td>1.83</td>
<td>1.79</td>
<td>1.21</td>
<td>0.99</td>
</tr>
<tr>
<td>New Solved</td>
<td>33.59</td>
<td>35.50</td>
<td>30.50</td>
<td>30.50</td>
<td>28.93</td>
<td>44.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First-order Theorems</th>
<th>CVC4 1.4-FOF</th>
<th>iProver 1.4</th>
<th>leanCoP 2.2</th>
<th>Prover9 1.109a</th>
<th>Zipperpos 0.4-FOF</th>
<th>Muscadet 4.4</th>
<th>Princess 140704</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved</td>
<td>375.400</td>
<td>339.400</td>
<td>321.400</td>
<td>310.00</td>
<td>215.400</td>
<td>216.400</td>
<td>158.400</td>
</tr>
<tr>
<td>Av. CPU Time</td>
<td>13.19</td>
<td>29.31</td>
<td>22.88</td>
<td>17.9</td>
<td>46.03</td>
<td>18.11</td>
<td>55.15</td>
</tr>
<tr>
<td>Solutions</td>
<td>372.400</td>
<td>339.400</td>
<td>321.400</td>
<td>310.00</td>
<td>215.400</td>
<td>214.00</td>
<td>158.400</td>
</tr>
<tr>
<td>μEfficiency</td>
<td>571.00</td>
<td>361.00</td>
<td>466.00</td>
<td>18.0</td>
<td>228.00</td>
<td>216.00</td>
<td>129.00</td>
</tr>
<tr>
<td>SOTAC</td>
<td>0.22</td>
<td>0.18</td>
<td>0.17</td>
<td>0.13</td>
<td>0.15</td>
<td>0.14</td>
<td>0.13</td>
</tr>
<tr>
<td>Core Usage</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.99</td>
</tr>
<tr>
<td>New Solved</td>
<td>5.6</td>
<td>5.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Competitions: SMT COMP 2014

• Partly due to techniques:
  – Official winner in 11 division with quantifiers
  – (Unofficially) beat z3 in AUFLIA, UFLIA, UF, ...

![UF Table]

Division COMPLETE: The winner is CVC4

<table>
<thead>
<tr>
<th>Solver</th>
<th>Errors</th>
<th>Solved</th>
<th>Not Solved</th>
<th>Remaining</th>
<th>CPU Time (on solved instances)</th>
<th>Weighted medal score weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVC4</td>
<td>0</td>
<td>2732</td>
<td>98</td>
<td>0</td>
<td>87682.16</td>
<td>3.217</td>
</tr>
<tr>
<td>[Z3]</td>
<td>0</td>
<td>1802</td>
<td>1028</td>
<td>0</td>
<td>21936.93</td>
<td>1.400</td>
</tr>
<tr>
<td>CVC3</td>
<td>0</td>
<td>1682</td>
<td>1148</td>
<td>0</td>
<td>31862.96</td>
<td>1.219</td>
</tr>
<tr>
<td>veriT</td>
<td>0</td>
<td>1410</td>
<td>1420</td>
<td>0</td>
<td>7880.76</td>
<td>0.857</td>
</tr>
</tbody>
</table>
Summary and Future Work

• Conflict-based method for quantifiers in SMT
  – Supplements existing techniques
  – Improves performance, both in:
    • Number of instantiations required for UNSAT
    • Number of UNSAT benchmarks solved

• Future work:
  – More incremental instantiation strategies
  – Specialize techniques to other theories
    • Handle quantified formulas containing (e.g.) linear arithmetic
  – Completeness criteria
Thank You

• Solver is publicly available:
  http://cvc4.cs.nyu.edu/

• Techniques enabled by option:
  “cvc4 --quant-cf ...”