Synthesis by Quantifier Instantiation in CVC4

Andrew Reynolds
May 4, 2015
Overview

• SMT solvers: how they work
• Synthesis Problem: $\exists f. \forall x. P( f, x )$

There exists a function $f$ such that for all $x$, $P( f, x )$

• New approaches for synthesis problems in an SMT solver [CAV 15]
  • Implemented in the SMT solver CVC4
• Evaluation
SMT solvers

• Are powerful tools used in many formal methods applications:
  • Software and Hardware verification
  • Automated Theorem Proving
  • Scheduling and Planning
  • Software synthesis

• Reason about Boolean combinations of *theory* constraints:
  • Linear arithmetic : $2a + 1 > 0$
  • Bitvectors : $\text{bvsgt}(a, \#\text{bin}0001)$
  • Arrays : $\text{select}(\text{store}(a, 5, b), c) = 5$
  • Datatypes : $\text{tail}(\text{cons}(a, b)) = b$
  • ....
SMT Solver for Theory T

- Combines:
  - Off the shelf SAT solver
  - (Possibly combined) decision procedure for decidable theory T
- Components communicate via DPLL(T) framework
SMT Solver for Theory T

- Determines if set of formulas F is \( T \)-satisfiable

\[ F \]

DPLL(T)

SAT Solver

Decision Procedure for T

• Determines if set of formulas F is \( T \)-satisfiable
SMT Solver for Theory T

\[ f(a) > 0 \land f(a) < 4 \]

SMT Solver

SAT Solver

Decision Procedure for T

unsat

sat

• Model, for example \( f(a) = 1 \)
SMT Solver for Theory T

\[ f(a) > 0 \land f(a) < -1 \]

SMT Solver

SAT Solver

Decision Procedure for T

DPLL(T)

- No model
  - unsat
  - sat
SMT Solver for Theory T

\[ f(a) > 0 \land f(a) < -1 \]

- For decidable theories (e.g. here T is $T_{UF} + T_{LIA}$)
  - Solver is terminating with either “unsat” or “sat”
SMT Solver + Quantified Formulas

SMT solver

Ground solver

SAT Solver

Decision Procedure for T

DPLL(T)

Quantifiers Module

• SMT solvers have limited support for (first-order) quantified formulas \( \forall \)
SMT Solver + Quantified Formulas

- For input $f(a) > 0 \land \forall x. f(x) < 0$
  - **Ground solver** maintains a set of ground (variable-free) constraints: $f(a) > 0$
  - **Quantifiers Module** maintains a set of axioms: $\forall x. f(x) < 0$
SMT Solver + Quantified Formulas

\[ f(a) > 0 \]

Ground solver

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

SAT Solver

Decision Procedure for T

Quantifiers Module
SMT Solver + Quantified Formulas

- Ground solver checks T-satisfiability of current set of constraints

SAT Solver → Decision Procedure for T → Quantifiers Module

\[ f(a) > 0 \]

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

Ground solver

• Ground solver checks T-satisfiability of current set of constraints
SMT Solver + Quantified Formulas

- Quantifiers Module adds instances of axioms
- Goal: add instances until ground solver can answer “unsat”
SMT Solver + Quantified Formulas

\[ f(a) > 0, f(a) < 0, f(b) < 0, \ldots \]

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

- Since \( f(a) > 0 \) and \( f(a) < 0 \)

Ground solver

SAT Solver

Decision Procedure for T

Quantifiers Module

unsat
SMT Solver + Quantified Formulas

• Generally, a **sound but incomplete** procedure
  • Difficult to answer sat (when have we added enough instances of $Q[x]$?)

Diagram:
- **Ground solver**
  - SAT Solver
  - Decision Procedure for $T$
- **Quantifiers Module**
  - $Q[x]$

Flow:
- $F, Q[t_1], Q[t_2], ...$
- DPLL($T$)
- sat
- unsat
- $\forall$ sat
- $\exists$ sat
- instances of $Q$
Approaches for Quantifiers in SMT

- Heuristic instantiation (good for “unsat”):
  - E-matching [Detlefs et al 2003, Ge et al 2007, de Moura/Bjorner 2007]

- Complete approaches (may answer “sat”):
  - Local theory extensions [Sofronie-Stokkermans 2005]
  - Complete instantiation [Ge/de Moura 2009]
  - Finite model finding [Reynolds et al 2013]

⇒ Each limited to a particular fragment
The Synthesis problem

\[ \exists f. \forall x. P(f, x) \]

There exists a function \( f \) such that for all \( x \), property \( P \) holds.

• Most existing approaches for synthesis
  • E.g. [Solar-Lezama et al 2006, Udupa et al 2013, Milicevic et al 2014]
  • Rely on specialized solver that makes subcalls to an SMT Solver
• Approach for synthesis in this talk:
  • Instrument an approach for synthesis entirely inside SMT solver
Running Example: Max of Two Integers

\[ \exists f. \forall x y. (f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y)) \]

- Specifies that \( f \) computes the maximum of integers \( x \) and \( y \)
- Solution:
  \[ f := \lambda x y. \text{ite}(x>y, x, y) \]
How does an SMT solver handle Max example?

$$
\exists f. \forall x y. (f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y))
$$
How does an SMT solver handle Max example?

\[
\begin{align*}
f : \text{Int} \times \text{Int} &\rightarrow \text{Int} \\
\forall xy. (f(x,y) \geq x \land f(x,y) \geq y \land \\
&\quad (f(x,y) = x \lor f(x,y) = y))
\end{align*}
\]

- Straightforward approach:
  - Treat \( f \) as an uninterpreted function
  - Succeed if SMT solver can find correct interpretation of \( f \), answer “sat”
  - However, this is challenging
    - SMT solvers have limited ability to find models when \( \forall \) are present
    - It is difficult to directly synthesize interpretation \( \lambda xy.\text{ite}(x>y,x,y) \)
Refutation-Based Synthesis

\[ \exists f. \forall x. P(f, x) \]

• Since SMT solvers are limited at answering “sat” when \( \forall \) are present, 
  \( \Rightarrow \) Can we instead use a \textit{refutation-based} approach for synthesis?
What if we negate the synthesis conjecture?

\[ \neg \exists f. \forall x. P(f, x) \]

• Negate the synthesis conjecture

• If we are in a *satisfaction-complete* theory T (e.g. linear arithmetic, bitvectors):
  • *F* is *T*-satisfiable if and only if \( \neg F \) is *T*-unsatisfiable
  • In such cases:
    • If SMT solver can establish \( \neg \exists f. \forall x. P(f, x) \) is *unsatisfiable*
    • Then we know that \( \exists f. \forall x. P(f, x) \) is satisfiable (*f* has a solution)
Challenge: Second-Order Quantification

\[ \neg \exists f. \forall x. P(f, x) \]

negate

\[ \forall f. \exists x. \neg P(f, x) \]

• Want to show negated formula is unsatisfiable

• Challenge: outermost quantification \( \forall f \) over function \( f \)
  • No SMT solvers directly support second-order quantification

• However, we can avoid this quantification using two approaches:
  1. When property \( P \) is single invocation for \( f \)
  2. When \( f \) is given syntactic restrictions
Challenge: Second-Order Quantification

\[ \neg \exists f. \forall x. P(f, x) \]

negate

\[ \forall f. \exists x. \neg P(f, x) \]

• Want to show negated formula is unsatisfiable
• Challenge: outermost quantification \( \forall f \) over function \( f \)
  • No SMT solvers directly support second-order quantification
• However, we can avoid this quantification using two approaches:
  1. When property \( P \) is single invocation for \( f \)  \( \iff \) Focus of this talk
  2. When \( f \) is given syntactic restrictions
Single Invocation Property : Max Example

\( \forall f. \exists x y. (f(x, y) < x \lor f(x, y) < y \lor (f(x, y) \neq x \land f(x, y) \neq y)) \)
Single Invocation Property: Max Example

\[ \forall f. \exists x y. (f(x, y) < x \lor f(x, y) < y \lor (f(x, y) \neq x \land f(x, y) \neq y)) \]

• **Single invocation** properties
  • Are properties such that:
    • All occurrences of \( f \) are of a particular form, e.g. \( f(x, y) \) above
  • Are a common class of properties useful for:
    • Software Synthesis (post-conditions describing the result of a function)

• Examples of properties that are not single invocation:
  • \( \forall c. \exists x y. c(x, y) = c(y, x) \), e.g. \( c \) is commutative
Single Invocation Property: Max Example

\[ \forall f. \exists x y. (f(x, y) < x \lor f(x, y) < y \lor \neg f(x, y) = x \land \neg f(x, y) = y) \]

Push quantification downwards

\[ \exists x y. \forall g. (g < x \lor g < y \lor \neg g = x \land \neg g = y) \]

- Occurrences of \( f(x, y) \) are replaced with integer variable \( g \)
- Resulting formula is equisatisfiable, and \textit{first-order}
Single Invocation Property : Max Example

\( \forall f. \exists x y. (f(x,y) < x \lor f(x,y) < y \lor
(f(x,y) \neq x \land f(x,y) \neq y)) \)

Push quantification downwards

\( \exists x y. \forall g. (g < x \lor g < y \lor
(g \neq x \land g \neq y)) \)

Skolemize, for fresh \( a \) and \( b \)

\( \forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b)) \)
Solving Max Example

$$\forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b))$$
Solving Max Example

\[ \forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b)) \]
Solving Max Example

\[\forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b))\]

Ground solver instances \(a/g, b/g\)

Quantifiers Module
Solving Max Example

\[ \forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b)) \]

Quantifiers Module

Ground solver

\[ a < b \land b < a \]

simplify
Solving Max Example

\[ \forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b)) \]

\(\text{Ground solver}\)

\(\text{Quantifiers Module}\)

unsat \implies \forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b)) \text{ is unsatisfable,}
implies original synthesis conjecture has a solution
How do we get solutions?

- Given refutation-based approach for synthesis conjecture $\exists f. \forall x. P(f(x), x)$
  - Solution for $f$ can be extracted from unsatisfiable core of instantiations
How do we get solutions?

\[ \exists f. \forall x. P(f(x), x) \]

negate, translate to FO

\[ \forall g. \neg P(g, k) \]

Ground solver

Quantifiers Module
How do we get solutions?

\[ \neg P(t_1, k), \ldots, \neg P(t_n, k) \]

**Ground solver**

**Quantifiers Module**

\[ \neg P(g, k) \]

\[ \exists f. \forall x. P(f(x), x) \]

instances

negate, translate to FO
How do we get solutions?

$$\neg P(t_1,k), \ldots, \neg P(t_n,k)$$

Ground solver

$$\forall g. \neg P(g,k)$$

Quantifiers Module

$\exists f. \forall x. P(f(x),x)$

negate, translate to FO

instances

unsat

$$\neg P(t_1,k), \ldots, \neg P(t_n,k) \models \text{false}$$
How do we get solutions?

\[-P(t_1, k), \ldots, -P(t_n, k)\]

Ground solver

\[\exists f. \forall x. P(f(x), x)\]

Quantifiers Module

instances

\[\forall g. -P(g, k)\]

negate, translate to FO

Claim the following is a solution for \(f\):

\[\lambda x. \text{ite}(P(t_1, k), t_1, \text{ite}(P(t_2, k), t_2, \ldots \text{ite}(P(t_{n-1}, k), t_{n-1}, t_n)\ldots)) [x/k] \]
Why is this a solution?

**Given**
\[ \exists f . \forall x . P ( f ( x ) , x ) \]

**Found**
\[ \neg P ( t_1 , k ) , \ldots , \neg P ( t_n , k ) \models \text{false} \]

**Claim** the following is a solution for \( f \):
\[
\lambda x . \text{ite} ( P ( t_1 , k ) , t_1 , \ldots , \text{ite} ( P ( t_{n-1} , k ) , t_{n-1} , t_n ) \ldots ) [ x / k ]
\]
Why is this a solution?

**Given** \( \exists f. \forall x. P(f(x), x) \)

**Found** \( \neg P(t_1, k), \ldots, \neg P(t_n, k) \models \text{false} \)

**Claim** the following is a solution for \( f \):
\[
\lambda x. \text{ite}( P(t_1, k), t_1, \\
\quad \text{ite}( P(t_2, k), t_2, \\
\quad \quad \ldots \\
\quad \quad \text{ite}( P(t_{n-1}, k), t_{n-1}, \\
\quad \quad \quad t_n) \ldots) [x/k]
\]

If \( P \) holds for \( t_1 \), return \( t_1 \)
Why is this a solution?

Given $\exists f. \forall x. P(f(x), x)$

Found $\neg P(t_1, k), \ldots, \neg P(t_n, k) \models false$

Claim the following is a solution for $f$:

$\lambda x. \text{ite}(P(t_1, k), t_1, \text{ite}(P(t_2, k), t_2, \ldots \text{ite}(P(t_{n-1}, k), t_{n-1}, t_n)\ldots)[x/k]$

If $P$ holds for $t_2$, return $t_2$
Why is this a solution?

Given  \( \exists f . \forall x . P(f(x), x) \)

Found  \( \neg P(t_1, k), \ldots, \neg P(t_n, k) \models \text{false} \)

Claim the following is a solution for \( f \):
\[
\lambda x. \ite(P(t_1, k), t_1, \\
\ite(P(t_2, k), t_2, \\
\ldots \\
\ite(P(t_{n-1}, k), t_{n-1}, t_n) \ldots) [x/k]
\]

\( \text{If } P \text{ holds for } t_{n-1}, \text{ return } t_{n-1} \)
Why is this a solution?

**Given** \( \exists f . \forall x . P(f(x), x) \)

**Found** \( \neg P(t_1, k), \ldots, \neg P(t_n, k) \models \text{false} \)

**Claim** the following is a solution for \( f \):

\[
\lambda x. \text{ite}( P(t_1, k), t_1, \\
\text{ite}( P(t_2, k), t_2, \\
\ldots \\
\text{ite}( P(t_{n-1}, k), t_{n-1}, \\
t_n) \ldots ) [x/k]
\]

Why does \( P(t_n, k) \) hold?
Why is this a solution?

Given \( \exists f. \forall x. P(f(x), x) \)

Found \( \neg P(t_1, k), \ldots, \neg P(t_{n-1}, k) \models P(t_n, k) \)

Claim the following is a solution for \( f \):
\[
\lambda x. \text{ite} ( P(t_1, k), t_1, \\
\text{ite} ( P(t_2, k), t_2, \\
\ldots \\
\text{ite} ( P(t_{n-1}, k), t_{n-1}, \\
\quad t_n) \ldots ) [x/k]
\]

Due to unsatisfiable core
Solution for Max Example

**Given** \( \exists f. \forall x y. (f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y)) \)
Solution for Max Example

**Given** \( \exists f. \forall x y. (f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y)) \)

**Found** 
\[ \neg (a \geq a \land a \geq b \land (a = a \lor a = b)), \]
\[ \neg (b \geq a \land b \geq b \land (b = a \lor b = b)) \quad |\quad = \quad \text{false} \]
Solution for Max Example

Given

$$\exists f. \forall xy. (f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y))$$

Found

$$\neg (a \geq a \land a \geq b \land (a = a \lor a = b)) \land \neg (b \geq a \land b \geq b \land (b = a \lor b = b)) \models false$$

Claim

the following is a solution for $$f$$:

$$\lambda xy. \text{ite}(a \geq a \land a \geq b \land (a = a \lor a = b), a, b)\ldots)[x/a][y/b]$$
Solution for Max Example

Given  \( \exists f. \forall x y. (f(x, y) \geq x \land f(x, y) \geq y \land (f(x, y) = x \lor f(x, y) = y)) \)

Found  \[ \neg (a \geq a \land a \geq b \land (a = a \lor a = b)), \quad \neg (b \geq a \land b \geq b \land (b = a \lor b = b)) \]

\( |\rightarrow \text{false} \)

Claim the following is a solution for \( f \):
\[ \lambda x y. \text{ite}(x \geq x \land x \geq y \land (x = x \lor x = y), x, y) \ldots \)
Solution for Max Example

**Given**
\[ \exists f. \forall xy. (f(x,y) \geq x \land f(x,y) \geq y \land (f(x,y) = x \lor f(x,y) = y)) \]

**Found**
\[ \neg (a \geq a \land a \geq b \land (a = a \lor a = b)) \]
\[ \neg (b \geq a \land b \geq b \land (b = a \lor b = b)) \]
\[ \models false \]

**Claim**
The following is a solution for \( f \):
\[ \lambda xy. \text{ite}(x \geq y, x, y) \]
Evaluation

• Implemented techniques in SMT solver CVC4
• Compared CVC4 against tools taken from 2014 SyGuS competition
  • In particular: enumerative CEGIS solver **ESolver** (Upenn)
• Of 243 benchmarks from this competition:
  • 176 were single invocation
### Results

<table>
<thead>
<tr>
<th></th>
<th>array (32)</th>
<th>bv (7)</th>
<th>hd (56)</th>
<th>icfp (50)</th>
<th>int (15)</th>
<th>let (8)</th>
<th>mutl (8)</th>
<th>Total (176)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>time</td>
<td>#</td>
<td>time</td>
<td>#</td>
<td>time</td>
<td>#</td>
<td>time</td>
</tr>
<tr>
<td>Esolver</td>
<td>3</td>
<td>467.6</td>
<td>2</td>
<td>71.6</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>1380.4</td>
</tr>
<tr>
<td>cvc4</td>
<td>30</td>
<td>1448.6</td>
<td>5</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0.1</td>
</tr>
</tbody>
</table>

- In total,
  - cvc4 finds solution for 35 that ESolver does not
  - ESolver finds solution for 2 that cvc4 does not
- Solves 25 benchmarks unsolved by any other known solver
  - Many of these in fraction of a second
Results: Max Example

<table>
<thead>
<tr>
<th>Solver</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Esolver</td>
<td>0.01</td>
<td>1377.10</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>cvc4</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.05</td>
<td>0.1</td>
<td>0.3</td>
<td>1.6</td>
<td>8.9</td>
<td>81.5</td>
</tr>
</tbody>
</table>

- For class of properties synthesizing function taking max of n integers
  - cvc4 scales well to max9+
  - No solver from SyGuS competition synthesized max5 with timeout of an hour
Summary

• Refutation-based approach for synthesis
• Solutions constructed from unsatisfiable core of instantiations
• Implemented in CVC4
• Highly competitive for single invocation properties

⇒ For more details, see CAV 15 paper
“Counterexample Guided Quantifier Instantiation for Synthesis in SMT”
with Morgan Deters, Viktor Kuncak, Cesare Tinelli, and Clark Barrett
Thanks!

- CVC4 publicly available at:
  http://cvc4.cs.nyu.edu/web/

- Handles inputs in the sygus language format *.sl
  - Techniques in this presentation enabled by argument "--cegqi-si"