Recent Advances in Instantiation-Based Techniques and their Implementation in CVC4

Andrew Reynolds

July 2, 2016
Outline

• CVC4

• SMT solver architecture
  ...and how it extends to $\forall$ reasoning via quantifier instantiation:
  $$\forall x. \psi [x] \Rightarrow \psi [t]$$

• Recent strategies for quantifier instantiation in CVC4:
  • E-matching, conflict-based, model-based, counterexample-guided

• Challenges, future work
CVC4: Past and Present Team Members

Clark Barrett (NYU)
Cesare Tinelli (U Iowa)
Morgan Deters (NYU)
Kshitij Bansal (Google)
François Bobot (CEA)
Chris Conway (Google)
Liana Hadarean (Mentor Graphics)
Dejan Jovanović (SRI)
Tim King (Google)
Tianyi Liang (Two Sigma)
Andrew Reynolds (U Iowa)
Nestan Tsiskaridze (U Iowa)
Martin Brain (U Oxford)
Guy Katz (Stanford)
Paul Meng (U Iowa)
CVC4: Past and Present Support
CVC4 is Expressive and Featureful

• **Boolean combinations of theory constraints**
  • UF, Arrays
  • Linear real/integer arithmetic
  • Bitvectors
  • (Co)inductive datatypes
  • Strings
  • Sets with Cardinality

• **Mixed constraints** over all built-in theories

• **Quantifiers** \( \forall \)

• **Models, proofs, unsat cores**
CVC4 is Expressive and Featureful

- Boolean combinations of theory constraints
  - UF, Arrays
  - Linear real/integer arithmetic
  - Bitvectors
  - (Co)inductive datatypes
  - Strings
  - Sets with Cardinality
- Mixed constraints over all built-in theories
- Quantifiers $\forall$
- Models, proofs, unsat cores

⇒ Focus of this talk
Approaches for Satisfiability of $\forall$ in Tools

• First order theorem provers focus on $\forall$ reasoning
  ...but have been extended in the past decade to theory reasoning

• SMT solvers focus mostly on ground theory reasoning
  ...but have been extended in the past decade to $\forall$ reasoning
Approaches for Satisfiability of $\forall$ in Tools

- First order theorem provers focus on $\forall$ reasoning
  ...but have been extended in the past decade to theory reasoning:
  - **Vampire, E, SPASS, iProver**
    - First-order resolution + superposition [Robinson 65, Nieuwenhuis/Rubio 99]
    - AVATAR in Vampire [Voronkov 14, Reger et al 15]
    - Some instantiation-based [Ganzinger/Korovin 03]

- SMT solvers focus mostly on ground theory reasoning
  ...but have been extended in the past decade to $\forall$ reasoning
Approaches for Satisfiability of $\forall$ in Tools

• First order theorem provers focus on $\forall$ reasoning
  ...but have been extended in the past decade to theory reasoning:
  • **Vampire, E, SPASS, iProver**
    • First-order resolution + superposition [Robinson 65, Nieuwenhuis/Rubio 99]
    • AVATAR in Vampire [Voronkov 14, Reger et al 15]
    • Some instantiation-based [Ganzinger/Korovin 03]

• SMT solvers focus mostly on ground theory reasoning
  ...but have been extended in the past decade to $\forall$ reasoning:
  • **Z3, CVC4, VeriT, Alt-Ergo**
    • Some superposition-based [deMoura et al 09]
    • Mostly instantiation-based [Detlefs et al 03, deMoura et al 07, Ge et al 09, ...]
Approaches for Satisfiability of $\forall$ in Tools

• First order theorem provers focus on $\forall$ reasoning
  ...but have been extended in the past decade to theory reasoning:
  • Vampire, E, SPASS, iProver
    • First-order resolution + superposition [Robinson 65, Nieuwenhuis/Rubio 99]
    • AVATAR in Vampire [Voronkov 14, Reger et al 15]
    • Some instantiation-based [Ganzinger/Korovin 03]

• SMT solvers focus mostly on ground theory reasoning
  ...but have been extended in the past decade to $\forall$ reasoning:
  • Z3, CVC4, VeriT, Alt-Ergo
    • Some superposition-based [deMoura et al 09]
    • Mostly instantiation-based [Detlefs et al 03, deMoura et al 07, Ge et al 09, ...]
Quantified Formulas in DPLL(T): Basics

\[(P(a) \lor f(b) = a + 1)\]
\[ (\neg \forall x. P(x) \lor \forall y. \neg P(y) \lor R(y)) \]
\[ (\forall x. f(x) = g(x) + h(x) \lor \neg R(a)) \]

⇒ Given the above input
Quantified Formulas in DPLL(T): Basics

Consider the propositional abstraction of the formula

• Atoms may encapsulate quantified formulas with Boolean structure
  • E.g. $\forall y. \neg P(y) \lor R(y)$

• $(P(a) \lor f(b) > a + 1)$
  $(\neg \forall x. P(x) \lor \forall y. \neg P(y) \lor R(y))$
  $(\forall x. f(x) = g(x) + h(x) \lor \neg P(a))$

• Consider the propositional abstraction of the formula
  • Atoms may encapsulate quantified formulas with Boolean structure
    • E.g. $\forall y. \neg P(y) \lor R(y)$
Quantified Formulas in DPLL(T): Basics

- Find propositional satisfying assignment via off-the-shelf SAT solver
Quantified Formulas in DPLL(T): Basics

- $P(a) \rightarrow \text{true}$
- $\forall y. \neg P(y) \lor R(y) \rightarrow \text{true}$
- $f(b) = a + 1 \rightarrow \text{true}$
- $\forall x. f(x) = g(x) + h(x) \rightarrow \text{true}$
- $\forall x. P(x) \rightarrow \text{false}$

- Find propositional satisfying assignment via off-the-shelf SAT solver
Quantified Formulas in DPLL(T): Basics

\[(P(a) \lor f(b) > a + 1)\]
\[\neg \forall x. P(x) \lor \forall y. \neg P(y) \lor R(y)\]
\[\forall x. f(x) = g(x) + h(x) \lor \neg P(a)\]

SAT Solver

- \(P(a)\) → true
- \(f(b) > a + 1\) → true
- \(\forall x. P(x)\) → false
- \(\forall y. \neg P(y) \lor R(y)\) → true
- \(\forall x. f(x) = g(x) + h(x)\) → true

⇒ Consider original atoms
Quantified Formulas in DPLL(T): Basics

SAT Solver

\[
(P(a) \lor f(b) > a + 1) \\
(\neg \forall x. P(x) \lor \forall y. \neg P(y) \lor R(y)) \\
(\forall x. f(x) = g(x) + h(x) \lor \neg P(a))
\]

Propositional assignment can be seen as a set of T-literals \( M \)

- Must check if \( M \) is T-satisfiable

\( M \)

\( \Rightarrow \) Propositional assignment can be seen as a set of T-literals \( M \)
- Must check if \( M \) is T-satisfiable
Quantified Formulas in DPLL(T): Basics

\[
(P(a) \lor f(b) > a + 1) \\
(\neg \forall x. P(x) \lor \forall y. \neg P(y) \lor R(y)) \\
(\forall x. f(x) = g(x) + h(x) \lor \neg P(a))
\]

\[\Rightarrow\text{Distribute ground literals to T-solvers, } \forall \text{ literals to quantifiers module}\]
Quantified Formulas in DPLL(T): Basics

SAT Solver

$$\begin{align*} &P(a) \lor f(b) > a + 1 \\ &\neg \forall x. P(x) \lor \forall y. \neg P(y) \lor R(y) \\ &\forall x. f(x) = g(x) + h(x) \lor \neg P(a) \end{align*}$$

These solvers may choose to add conflicts/lemmas to clause set

$$\Rightarrow$$ These solvers may choose to add conflicts/lemmas to clause set
DPLL($T_1$+..+$T_n$)+Quantifiers: Overview

SAT Solver

T-Clauses $F$

Satisfying Assignment $M$

Conflicts, lemmas

$M_1$

$T_1$-solver

$M_n$

$T_n$-solver

Quantifiers Module

...when $F$ is propositionally unsatisfiable

$\Rightarrow$ Each of these components may:
- Report $M$ is $T$-unsatisfiable by reporting conflict clauses
- Report lemmas if they are unsure

[Nieuwenhuis/Oliveras/Tinelli 06]
DPLL($T_1+..+T_n$)+Quantifiers: Overview

SAT Solver

T-Clauses $F$

...when $F$ is propositionally unsatisfiable

Satisfying Assignment $M$

$T_1$-solver

...$M_1$

$T_n$-solver

...$M_n$

Quantifiers Module

$Q$

⇒ If no component adds a lemma, then it must be the case that $M$ is $T_1+\ldots+T_n$-satisfiable

[van Nieuwenhuis/Oliveras/Tinelli 06]
In this talk: DPLL(T) + Quantifiers, simplified

For purposes of this talk, partition $M$ into quantifier-free part $E$, and set of $\forall$ formulas $Q$. 

For $\exists$ formulas $Q$, use $\forall$ quantifiers.

\[ \forall \mathcal{F} \exists \mathcal{Q} \]
In this talk: DPLL(T)+Quantifiers, simplified

SAT Solver → T-Clauses F → Satisfying Assignment M → ... → Theory solver(s) → E → E is T-satisfiable → Quantifiers Module → Q

⇒ Theory solvers determine whether E is T-(un)satisfiable
In this talk: DPLL(T)+Quantifiers, simplified

If $E$ is T-satisfiable, quantifiers module may be invoked

$E \cup Q$ is T-satisfiable

⇒ If $E$ is T-satisfiable, quantifiers module may be invoked
In this talk: DPLL(T)+Quantifiers, simplified

SAT Solver

T-Clauses $F$

Satisfying Assignment $M$

Theory solver(s)

E

Lemmas

$E \cup Q$ is T-satisfiable

⇒ Will discuss how the quantifiers module is implemented
DPLL(T)+Quantifiers, further simplified

Ground Solver

T-clauses $F$

Lemmas

Quantifiers Module

• Inputs:
  • Set of ground T-literals $E$
  • Set of $\forall$ formulas $Q$

• Outputs:
  • “$E \cup Q$ is T-satisfiable”, or
    $\Rightarrow F$ is T-satisfiable
  • Set of lemmas to add to $F$

unsat

ground literals $E$

$\forall$ formulas $Q$

sat

Set of $\exists$ formulas $Q$
DPLL(T)+Quantifiers, further simplified

- T-clauses \( F \)
- Lemmas
- ground literals \( E \)
- \( \forall \) formulas \( Q \)
- Inputs:
  - Set of ground T-literals \( E \)
  - Set of \( \forall \) formulas \( Q \)
- Outputs:
  - “\( E \cup Q \) is T-satisfiable”, or
  - \( \Rightarrow F \) is T-satisfiable
  - Set of lemmas to add to \( F \)
- Recurrent Questions:
  - Which lemmas do we add?
  - How do we know \( E \cup Q \) is T-satisfiable?
  - When do we invoke it?
Quantifier Instantiation

\[ \forall x. P(x) \]

\[ P(a), P(b) \]

\[ f(b) > a + 1 \]

Quantifiers Module
Quantifier Instantiation

Universal quantification handled by Instantiation

- Choose ground term(s) \( t \), lemma(s) say \( \forall x. P(x) \) implies \( P(a) \)
- May be applied ad infinitum, for \( x \rightarrow a, b, c, d, \ldots \)
- Selection of instances is the core challenge
Quantifiers Module : Recurrent Question

• Which instances do we add?
  • E-matching [Detlefs et al 03]
  • Conflict-based quantifier instantiation [Reynolds et al FMCAD14]
  • Model-based quantifier instantiation [Ge, de Moura CAV09]
  • Counterexample-guided quantifier instantiation [Reynolds et al CAV15]
Techniques for Quantifier Instantiation: Overview

Ground Solver

Instances of $\forall$ in $Q$

F, ...

Satisfying assignment $E, Q$

Quantifiers Module

Conflict-Based

E-matching

Model Based

CE-Guided

Generally, used for quantifiers with UF

Generally, used for quantifiers w/o UF

$E \cup Q$ is T-satisfiable

sat

unsat
Techniques for Quantifier Instantiation: Overview

- Ground Solver
  - E, Q
  - Satisfying assignment
  - Instances of $\forall$ in $Q$
  - $F, ...$

- Quantifiers Module
  - Conflict-Based
  - E-matching
  - Model Based
  - CE-Guided
    - Generally, used for quantifiers with UF
    - Generally, used for quantifiers w/o UF

$E \cup Q$ is T-satisfiable

$\Rightarrow$ Will describe details of each of these strategies
E-matching

- Introduced in Nelson’s Phd Thesis [Nelson 80]
  - Implemented in early SMT solvers, e.g. Simplify [Deltefs et al 03]
- Most widely used and successful technique for quantifiers in SMT
  - Software verification
    - Boogie/Dafny, Leon, SPARK, Why3
  - Automated Theorem Proving
    - Sledgehammer
- Variants implemented in numerous solvers:
  - Z3 [deMoura et al 07], CVC3 [Ge et al 07], CVC4, Princess [Ruemmer 12], VeriT, Alt-Ergo
E-matching

\[ E \cdot P(a) \neg P(b) R(c) \neg R(a) S(e) \]

\[ Q \cdot \forall x. P(x) \lor R(x) \]

E-matching
E-matching

\[ \forall x. P(x) \lor R(x) \]

- \( P(a) \)
- \( \neg P(b) \)
- \( R(c) \)
- \( \neg R(a) \)
- \( S(e) \)
E-matching

$\forall x. P(x) \lor R(x)$

**Pattern**

$\neg P(a)$
$\neg P(b)$
$R(c)$
$\neg R(a)$
$S(e)$

**Idea**: choose instances based on pattern matching
E-matching

\[ \exists x. P(x) \lor R(x) \]

Pattern

\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(a) \lor R(a) \]

\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(b) \lor R(b) \]
E-matching

\[ \neg P(a) \]
\[ \neg P(b) \]
\[ R(c) \]
\[ \neg R(a) \]
\[ S(e) \]

\[ (\forall x. P(x) \lor R(x)) \implies P(a) \lor R(a) \]
\[ (\forall x. P(x) \lor R(x)) \implies P(b) \lor R(b) \]
\[ (\forall x. P(x) \lor R(x)) \implies P(c) \lor R(c) \]
E-matching: Functions, Equality

\[ P(a, c) \]
\[ f(b) = a \]

\[ \forall xy. P(f(x), y) \Rightarrow g(x) = y \]
E-matching: Functions, Equality

\[ P(a, c) \]
\[ f(b) = a \]

\[ \forall xy. P(f(x), y) \Rightarrow g(x) = y \]

\[ \Rightarrow \text{In E-matching, Pattern } \textit{matching} \text{ takes into account equalities in } E \]
E-matching: Functions, Equality

E:
- \( P(a, c) \)
- \( f(b) = a \)

Q:
- \( \forall xy. P(f(x), y) \Rightarrow g(x) = y \)

Pattern
E-matching: Functions, Equality

P(a,c)  
f(b)=a  
E-matching

∀xy. P(f(x),y) \Rightarrow g(x)=y

P(a,c)
E-matching: Functions, Equality

E
\[ P(a,c) \]
\[ f(b) = a \]

Q
\[ \forall xy. P(f(x), y) \Rightarrow g(x) = y \]

E-matching

\[ T = P(a, c) \]

a = f(b)

b
c

Congruence closure of E
E-matching: Functions, Equality

E
\[ P(a, c) \]
\[ f(b) = a \]

Q
\[ \forall xy. P(f(x), y) \Rightarrow g(x) = y \]

\[ P(f(b), c) \]

...E implies \( P(a, c) \iff P(f(b), c) \)

\[ a = f(b) \]
\[ b \]
\[ c \]

T = \( P(a, c) \)
E-matching: Functions, Equality

E:
- \( P(a,c) \)
- \( f(b)=a \)

Q:
- \( \forall xy. P(f(x),y) \Rightarrow g(x)=y \)
- \( P(f(b),c) \)
E-matching: Intuition

• Say E-matching returns the instance $(\forall x. \Psi \Rightarrow \Psi\{x\rightarrow t\})$

$
\Rightarrow \textit{Why is this instance useful?}$
E-matching: Intuition

• Say E-matching returns the instance ($\forall x. \Psi \Rightarrow \Psi\{x\rightarrow t\}$) 

$\Rightarrow$ Why is this instance useful? 

• We are interested in satisfiability of $E \cup Q$
E-matching: Intuition

• Say E-matching returns the instance ($\forall x. \Psi \Rightarrow \Psi\{x \mapsto t\}$)

⇒ *Why is this instance useful?*

• We are interested in satisfiability of $E \cup Q$

• Assume pattern $p$ is a subterm of $\Psi$, e.g. $\forall x. \Psi[p]$
E-matching: Intuition

• Say E-matching returns the instance \((\forall x . \Psi \Rightarrow \Psi\{x\rightarrow t\})\)

\[\Rightarrow Why \text{ is this instance useful?}\]

• We are interested in satisfiability of \(E \cup Q\)

• Assume pattern \(p\) is a subterm of \(\Psi\), e.g. \(\forall x . \Psi[p]\)

• E-matching finds a ground term \(g\) from \(E\), where \(g=p\{x\rightarrow t\}\) is implied by \(E\)
E-matching: Intuition

• Say E-matching returns the instance ($\forall x . \Psi \Rightarrow \Psi[x \mapsto t]$)

⇒ Why is this instance useful?

• We are interested in satisfiability of $E \cup Q$

• Assume pattern $p$ is a subterm of $\Psi$, e.g. $\forall x . \Psi[p]$

• E-matching finds a ground term $g$ from $E$, where $g = p[x \mapsto t]$ is implied by $E$

• Thus: $\Psi[g]$ is implied by $E \cup \{\Psi[p] \{x \mapsto t\}\}$
E-matching: Intuition

• Say E-matching returns the instance (\(\forall x. \Psi \Rightarrow \Psi\{x\rightarrow t\}\) )

  \(\Rightarrow\) *Why is this instance useful?*

• We are interested in satisfiability of \(E \cup Q\)

• Assume pattern \(p\) is a subterm of \(\Psi\), e.g. \(\forall x. \Psi[p]\)

• E-matching finds a ground term \(g\) from \(E\), where \(g=p\{x\rightarrow t\}\) is implied by \(E\)

• **Thus:** \(\Psi[g]\) is implied by \(E \cup \{ \Psi[p]\{x\rightarrow t\}\}\)

  \(\Rightarrow\) *In other words, from \(Q\), we learn information \(\Psi[g]\) about a term \(g\) from \(E\)*
E-matching: Intuition

• Say E-matching returns the instance \((\forall x. \, \Psi \Rightarrow \Psi\{x \mapsto t\})\)

  \(\Rightarrow\) Why is this instance useful?

• We are interested in satisfiability of \(E \cup Q\)

• Assume pattern \(p\) is a subterm of \(\Psi\), e.g. \(\forall x. \, \Psi[p]\)

• E-matching finds a ground term \(g\) from \(E\), where \(g=p\{x \mapsto t\}\) is implied by \(E\)

• Thus: \(\Psi[g]\) is implied by \(E \cup \{\Psi[p]\ \{x \mapsto t\}\}\)

  \(\Rightarrow\) In other words, from \(Q\), we learn information \(\Psi[g]\) about a term \(g\) from \(E\)

Learn \(P(a, c) \Rightarrow g(b) = c\) as a result of
\[
\{P(a, c), f(b) = a\} \cup \{P(f(b), c) \Rightarrow g(b) = c\}
\]

\(E\) with new instance
E-matching: Challenges

• E-matching has no standard way of selecting patterns
• E-matching generates too many instances
  • Many instances may overload the ground solver
• E-matching is incomplete
  • It may be non-terminating
  • When it terminates, we generally cannot answer “∀E ∪ Q is T-satisfiable”
    • Although for some fragments+variants, we may guarantee (termination ⇔ model)
      • Decision Procedures via Triggers [Dross et al 13]
      • Local Theory Extensions [Bansal et al 15]
        ⇒ Typically are established by a separate pencil-and-paper proof
E-matching: Pattern Selection

• In practice, **pattern selection** can be done either by:
  • The user, via annotations, e.g. (! ... :pattern ((P x)))
  • The SMT solver itself

• Recurrent questions:
  • **Which terms** be we permit as patterns? Typically, applications of UF:
    • Use \( f(x, y) \) but not \( x + y \) for \( \forall xy. f(x, y) = x + y \)
  • **What if multiple** patterns exist? Typically use all available patterns:
    • Use both \( P(x) \) and \( R(x) \) for \( \forall x. P(x) \lor R(x) \)
  • **What if no appropriate term** contains all variables? May use “multi-patterns”:
    • \( \{ R(x, y), R(y, z) \} \) for \( \forall xyz. (R(x, y) \land R(y, z)) \Rightarrow R(x, z) \)

• Pattern selections may impact performance significantly **[Leino et al 16]**
E-matching: Pattern Selection

• In practice, **pattern selection** can is done either by:
  • The user, via annotations, e.g. (! ... :pattern ((P x)))
  • The SMT solver itself

• Recurrent questions:
  • **Which terms** be we permit as patterns? Typically, applications of UF:
    • Use $f(x, y)$ but not $x+y$ for $\forall xy. f(x, y)=x+y$
  • **What if multiple** patterns exist? Typically use all available patterns:
    • Use both $P(x)$ and $R(x)$ for $\forall x. P(x) \lor R(x)$
  • **What if no appropriate term** contains all variables? May use “multi-patterns”:
    • $\{ R(x, y), R(y, z) \}$ for $\forall xyz. (R(x, y) \land R(y, z)) \Rightarrow R(x, z)$

• Pattern selections may impact performance significantly [Leino et al 16]
  • ...and may share similarities with literal selection heuristics in ATP, a la [Reger et al 16]?
• Typical problems in applications:
  • $F$ contains 1000s of clauses
• Typical problems in applications:
  • $F$ contains 1000s of clauses
  • Satisfying assignments contain 1000s of terms in $E$, 100s of $\forall$ in $Q$
E-matching: Too Many Instances

- Typical problems in applications:
  - $F$ contains 1000s of clauses
  - Satisfying assignments contain 1000s of terms in $E$, 100s of $\forall$ in $Q$
  - Leads to 100s
E-matching: Too Many Instances

- Typical problems in applications:
  - $F$ contains 1000s of clauses
  - Satisfying assignments contain 1000s of terms in $E$, 100s of $\forall$ in $Q$
  - Leads to 100s, 1000s
E-matching: Too Many Instances

• Typical problems in applications:
  • $F$ contains 1000s of clauses
  • Satisfying assignments contain 1000s of terms in $E$, 100s of $\forall$ in $Q$
  • Leads to 100s, 1000s, 10000s of instances
E-matching: Too Many Instances

$E_3$  
$Q$  
$\sim 100000$  
$\sim 100$  

$F_1$, $F_2$, $F_3$  
$\sim 10000$  

$\Rightarrow$ Ground solver is overloaded, loop becomes slow, ...solver times out
E-matching: Too Many Instances

<table>
<thead>
<tr>
<th># Instances</th>
<th>cvc3</th>
<th align="center"></th>
<th>cvc4</th>
<th align="center"></th>
<th>z3</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>%</td>
<td align="center">#</td>
<td>%</td>
<td align="center">#</td>
<td>%</td>
</tr>
<tr>
<td>1-10</td>
<td>1464</td>
<td align="center">13.49%</td>
<td>1007</td>
<td align="center">8.87%</td>
<td>1321</td>
</tr>
<tr>
<td>10-100</td>
<td>1755</td>
<td align="center">16.17%</td>
<td>1853</td>
<td align="center">16.31%</td>
<td>2554</td>
</tr>
<tr>
<td>100-1000</td>
<td><strong>3816</strong></td>
<td align="center">35.16%</td>
<td><strong>3680</strong></td>
<td align="center">32.40%</td>
<td><strong>4553</strong></td>
</tr>
<tr>
<td>1000-10k</td>
<td>1893</td>
<td align="center">17.44%</td>
<td>2468</td>
<td align="center">21.73%</td>
<td>1779</td>
</tr>
<tr>
<td>10k-100k</td>
<td>1162</td>
<td align="center">10.71%</td>
<td>1414</td>
<td align="center">12.45%</td>
<td>823</td>
</tr>
<tr>
<td>100k-1M</td>
<td>560</td>
<td align="center">5.16%</td>
<td>607</td>
<td align="center">5.34%</td>
<td>376</td>
</tr>
<tr>
<td>1M-10M</td>
<td>193</td>
<td align="center">1.78%</td>
<td>330</td>
<td align="center">2.91%</td>
<td>139</td>
</tr>
<tr>
<td>&gt;10M</td>
<td>10</td>
<td align="center">0.09%</td>
<td>0</td>
<td align="center">0.00%</td>
<td><strong>8</strong></td>
</tr>
</tbody>
</table>

| >10M | 10  | 0.09%  | 0    | 0.00%  | **8**| 0.07%  |

(for 8 of benchmarks z3 solves, its E-matching procedure adds more than 10M instances)

- Evaluation on 33032 SMTLIB, TPTP, Isabelle benchmarks
- E-matching often requires **many instances**
  (Above, 16.6% required >10k, max 19.5M by z3 on a software verification benchmark from TPTP)
E-matching: Incompleteness

\[ E \]

empty

\[ Q \]

\[ \forall x. P(x) \]
\[ \forall x. \neg P(x) \]

\[ \Rightarrow \] E-matching is an incomplete procedure
E-matching: Incompleteness

If E-matching produces no instances, this does not guarantee \( E \cup Q \) is T-satisfiable.

\[
\begin{align*}
E & \{ \text{empty} \} \\
Q & \{ \forall x. P(x), \forall x. \neg P(x) \}
\end{align*}
\]

\( \Rightarrow \) If E-matching produces no instances, this does not guarantee \( E \cup Q \) is T-satisfiable.
E-matching: Summary

• Using matching ground terms from \( E \) against patterns in \( Q \):
  • From \( Q \), learn constraints about ground terms \( g \) from \( E \)

• Challenges
   Use conflict-based instantiation \([\text{Reynolds+Tinelli+deMoura FMCAD14}]\)
   Use model-based instantiation \([\text{Ge+deMoura CAV09}]\)
E-matching: Summary

• Using matching ground terms from $E$ against patterns in $Q$:
  • From $Q$, learn constraints about ground terms $g$ from $E$

• Challenges
  • What can we do when there too many instances to add?
  • What can we do when there are no instances to add, problem is “sat”?
E-matching: Summary

• Using matching ground terms from $E$ against patterns in $Q$:
  • From $Q$, learn constraints about ground terms $g$ from $E$

• Challenges
  • What can we do when there too many instances to add?
    $\Rightarrow$ Use conflict-based instantiation [Reynolds/Tinelli/deMoura FMCAD14]
  • What can we do when there are no instances to add, problem is “sat”?
    $\Rightarrow$ Use model-based instantiation [Ge/deMoura CAV09]
Conflict-Based Instantiation

• Implemented in solvers:
  • CVC4 [Reynolds et al 14], recently in VeriT [Barbosa16]

• Basic idea:
  1. Try to find a “conflicting” instance such that $E \cup \Psi \{x \rightarrow t\}$ implies $\bot$
     (by contrast, E-matching does not distinguish such instances)
  2. If one such instance can be found, return that instance only
     (and do not run E-matching)

$\Rightarrow$ Leads to fewer instances, improved ability of ground solver to answer “unsat”
Conflict-Based Instantiation

\[ \neg P(a), \neg P(b), \neg P(c), \neg R(a), \neg R(d), \neg R(e), \neg R(c) \]

E-matching

\[ \forall x. P(x) \lor R(x) \]

E-matching

Conflict-Based

Model Based
Conflict-Based Instantiation

\begin{align*}
&\forall x. P(x) \lor R(x) \\
&\neg P(a), \neg P(b) \\
&\neg P(c), \neg R(a) \\
&R(d), \neg R(e) \\
&\neg R(c)
\end{align*}

\Rightarrow \text{E-matching would produce } \{x\rightarrow a\}, \{x\rightarrow b\}, \{x\rightarrow c\}, \{x\rightarrow d\}, \{x\rightarrow e\}
Consider what we learn from these instances:

\[
E, Q, P(a) ∨ R(a) \models P(a) ∨ R(a) \\
E, Q, P(b) ∨ R(b) \models P(b) ∨ R(b) \\
E, Q, P(c) ∨ R(c) \models P(c) ∨ R(c) \\
E, Q, P(d) ∨ R(d) \models P(d) ∨ R(d) \\
E, Q, P(e) ∨ R(e) \models P(e) ∨ R(e)
\]

\[
(∀x. P(x) ∨ R(x)) \Rightarrow P(a) ∨ R(a) \\
(∀x. P(x) ∨ R(x)) \Rightarrow P(b) ∨ R(b) \\
(∀x. P(x) ∨ R(x)) \Rightarrow P(c) ∨ R(c) \\
(∀x. P(x) ∨ R(x)) \Rightarrow P(d) ∨ R(d) \\
(∀x. P(x) ∨ R(x)) \Rightarrow P(e) ∨ R(e)
\]
Conflict-Based Instantiation

\[ \forall x. P(x) \lor R(x) \]

\[ \rightarrow \]

Consider what we learn from these instances:

\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(a) \lor R(a) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(b) \lor R(b) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(c) \lor R(c) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(d) \lor R(d) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(e) \lor R(e) \]

By \( E \), we know \( P(a) \Leftrightarrow T \)
Conflict-Based Instantiation

Consider what we learn from these instances:

\[
E, Q, P(a) \lor R(a) \models T \\
E, Q, P(b) \lor R(b) \models P(b) \lor R(b) \\
E, Q, P(c) \lor R(c) \models P(c) \lor R(c) \\
E, Q, P(d) \lor R(d) \models P(d) \lor R(d) \\
E, Q, P(e) \lor R(e) \models P(e) \lor R(e)
\]
Conflict-Based Instantiation

Consider what we learn from these instances:

\[ E, Q, P(a) \lor R(a) \land T \]
\[ E, Q, P(b) \lor R(b) \land \bot \lor R(b) \]
\[ E, Q, P(c) \lor R(c) \land P(c) \lor R(c) \]
\[ E, Q, P(d) \lor R(d) \land P(d) \lor R(d) \]
\[ E, Q, P(e) \lor R(e) \land P(e) \lor R(e) \]

We know \( P(b) \leftrightarrow \bot \)
Consider what we learn from these instances:

\[
\begin{align*}
E, Q, P(a) & \lor R(a) \quad \implies T \\
E, Q, P(b) & \lor R(b) \quad \implies R(b) \\
E, Q, P(c) & \lor R(c) \quad \implies P(c) \lor R(c) \\
E, Q, P(d) & \lor R(d) \quad \implies P(d) \lor R(d) \\
E, Q, P(e) & \lor R(e) \quad \implies P(e) \lor R(e)
\end{align*}
\]
Conflict-Based Instantiation

Consider what we learn from these instances:

\[
\begin{align*}
E, Q, P(a) \lor R(a) & \models T \\
E, Q, P(b) \lor R(b) & \models R(b) \\
E, Q, P(c) \lor R(c) & \models R(c) \\
E, Q, P(d) \lor R(d) & \models P(d) \lor R(d) \\
E, Q, P(e) \lor R(e) & \models P(e) \lor R(e)
\end{align*}
\]

We know \(P(c) \iff \bot\).
Conflict-Based Instantiation

\[ P(a), \neg P(b), \neg P(c), \neg R(a), R(d), \neg R(e), \neg R(c) \]

\[ \forall x. P(x) \lor R(x) \]

\[ \Rightarrow \text{Consider what we learn from these instances:} \]

\[ E, Q, P(a) \lor R(a) \Rightarrow T \]
\[ E, Q, P(b) \lor R(b) \Rightarrow R(b) \]
\[ E, Q, P(c) \lor R(c) \Rightarrow R(c) \]
\[ E, Q, P(d) \lor R(d) \Rightarrow T \]
\[ E, Q, P(e) \lor R(e) \Rightarrow P(e) \lor R(e) \]

We know \[ R(d) \Leftrightarrow T \]
Conflict-Based Instantiation

Consider what we learn from these instances:

\begin{align*}
E, Q, P(a) \lor R(a) & \models T \\
E, Q, P(b) \lor R(b) & \models R(b) \\
E, Q, P(c) \lor R(c) & \models R(c) \\
E, Q, P(d) \lor R(d) & \models T \\
E, Q, P(e) \lor R(e) & \models P(e)
\end{align*}

We know \( R(e) \Leftrightarrow \bot \)
Consider what we learn from these instances:

\[
\begin{align*}
E, Q, P(a) \lor R(a) &\models T \\
E, Q, P(b) \lor R(b) &\models R(b) \\
E, Q, P(c) \lor R(c) &\models \bot \\
E, Q, P(d) \lor R(d) &\models T \\
E, Q, P(e) \lor R(e) &\models P(e)
\end{align*}
\]

We know \( R(c) \iff \bot \)
Consider what we learn from these instances:

\[
\begin{align*}
E, Q, P(a) \lor R(a) & \models T \\
E, Q, P(b) \lor R(b) & \models R(b) \\
E, Q, P(c) \lor R(c) & \models \bot \\
E, Q, P(d) \lor R(d) & \models T \\
E, Q, P(e) \lor R(e) & \models P(e)
\end{align*}
\]
Conflict-Based Instantiation

\[ \forall x. P(x) \lor R(x) \]

\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(a) \lor R(a) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(b) \lor R(b) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(c) \lor R(c) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(d) \lor R(d) \]
\[ (\forall x. P(x) \lor R(x)) \Rightarrow P(e) \lor R(e) \]

\[ P(c) \lor R(c) \] is a conflicting instance for \((E, Q)\)!
Consider what we learn from these instances:

- $E, Q, P(a) \lor R(a) \Rightarrow T$
- $E, Q, P(b) \lor R(b) \Rightarrow R(b)$
- $E, Q, P(c) \lor R(c) \Rightarrow \bot$
- $E, Q, P(d) \lor R(d) \Rightarrow T$
- $E, Q, P(e) \lor R(e) \Rightarrow P(e)$

Since $P(c) \lor R(c)$ suffices to derive $\bot$, return only this instance.

**Conflict-Based Instantiation**

- $\forall x. P(x) \lor R(x)$
- $E$
- $Q$
- \Rightarrow Conflict-based Instantiation

$(\forall x. P(x) \lor R(x)) \Rightarrow P(a) \lor R(a)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(b) \lor R(b)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(c) \lor R(c)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(d) \lor R(d)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(e) \lor R(e)$
Conflict-Based Instantiation

• Why are conflicts important?
  • As with the ground case, they prune the search space of DPLL(T)
    • Given a conflicting instance for \((E, Q)\) is added to the clause set \(F\)
      • Solver is forced to choose a new sat assignment \((E', Q')\)

Conflicting instance found, Backtrack

\(E, Q \rightarrow E', Q' \rightarrow \ldots \rightarrow E'', Q'' \rightarrow \ldots\)

unsat
Conflict-Based Instantiation: EUF

\[ a \neq c, f(b) = b, \]
\[ g(b) = a, f(a) = a, \]
\[ h(f(a)) = d, h(b) = c \]

\[ \forall x. f(g(x)) = h(f(x)) \]
Consider the instance \( \forall x. f(g(x)) = h(f(x)) \) for \((E, Q)\)?

- Is this conflicting for \((E, Q)\)?
Conflict-Based Instantiation: EUF

\[ a \neq c, f(b) = b, \quad g(b) = a, f(a) = a, \quad h(f(a)) = d, h(b) = c \]

\[ E, Q, f(g(b)) = h(f(b)) \mathrel{\models_E} f(g(b)) = h(f(b)) \]
Conflict-Based Instantiation: EUF

Consider the equivalence classes of $E$

\[
\begin{align*}
&\; a \neq c, f(b) = b, \\
&\; g(b) = a, f(a) = a, \\
&\; h(f(a)) = d, h(b) = c
\end{align*}
\]

\[
\forall x. f(g(x)) = h(f(x))
\]

\[
E_Q, f(g(b)) = h(f(b)) \models E \; f(g(b)) = h(f(b))
\]

Consider the equivalence classes of $E$
Conflict-Based Instantiation: EUF

Build partial definitions for functions in terms of representatives

\[ E, Q, f(g(b)) = h(f(b)) \models_E f(g(b)) = h(f(b)) \]
Conflict-Based Instantiation: EUF

\[ a \neq c, f(b) = b, \]
\[ g(b) = a, f(a) = a, \]
\[ h(f(a)) = d, h(b) = c, a = g(b) = f(a) \]

\[ \forall x. f(g(x)) = h(f(x)) \]

\[ E, Q, f(g(b)) = h(f(b)) \models_E f(g(b)) = h(f(b)) \]
Conflict-Based Instantiation: EUF

\[ \forall x. f(g(x)) = h(f(x)) \]

\[ a = g(b) = f(a) \]
\[ b = f(b) \]
\[ c = h(b) \]
\[ d = h(f(a)) \]

\[ E, Q, f(g(b)) = h(f(b)) \models_E f(g(b)) = h(b) \]
Conflict-Based Instantiation: EUF

\[ a \neq c, f(b) = b, \]
\[ g(b) = a, f(a) = a, \]
\[ h(f(a)) = d, h(b) = c, \]
\[ a = g(b) = f(a) \]
\[ b = f(b) \]
\[ c = h(b) \]
\[ d = h(f(a)) \]

\[ E, Q, f(g(b)) = h(f(b)) \models_E f(g(b)) = c \]
Conflict-Based Instantiation: EUF

\[ a \neq c, f(b) = b, \]
\[ g(b) = a, f(a) = a, \]
\[ h(f(a)) = d, h(b) = c, \]
\[ a = g(b) = f(a) \]
\[ b = f(b) \]
\[ c = h(b) \]
\[ d = h(f(a)) \]

\[ \forall x. f(g(x)) = h(f(x)) \]
Conflict-Based Instantiation: EUF

\[ E \left\{ a \neq c, f(b) = b, \right. \]
\[ g(b) = a, f(a) = a, \]
\[ h(f(a)) = d, h(b) = c \]

\[ Q \left\{ \forall x. f(g(x)) = h(f(x)) \right. \]

\[ a = g(b) = f(a) \]
\[ b = f(b) \]
\[ c = h(b) \]
\[ d = h(f(a)) \]

\[ E, Q, f(g(b)) = h(f(b)) \models_E a = c \]
Conflict-Based Instantiation: EUF

\[ a \neq c, f(b) = b, \quad g(b) = a, f(a) = a, \quad h(f(a)) = d, h(b) = c \]

\[ a = g(b) = f(a) \]
\[ b = f(b) \]
\[ c = h(b) \]
\[ d = h(f(a)) \]

\[ E, Q, f(g(b)) = h(f(b)) \models_E a = c \]
Conflict-Based Instantiation: EUF

\[ a \neq c, \quad f(b) = b, \quad g(b) = a, \quad f(a) = a, \quad h(f(a)) = d, \quad h(b) = c \]

From \( E \), we know \( a \neq c \)

\[ E, Q, f(g(b)) = h(f(b)) \models_E \]

From \( E \), we know \( a \neq c \)
Conflict-Based Instantiation: EUF

\( E \)

\[ a \neq c, f(b) = b, \]
\[ g(b) = a, f(a) = a, \]
\[ h(f(a)) = d, h(b) = c \]

\( Q \)

\[ \forall x. f(g(x)) = h(f(x)) \]

\[ E, Q, f(g(b)) = h(f(b)) \models_E \]

\( \perp \)

\[ f(g(b)) = h(f(b)) \text{ is a conflicting instance for } (E, Q) ! \]
Consider the same example, but where we don’t know $a \neq c$.

- Is the instance $f(g(b)) = h(f(b))$ still useful?
Conflict-Based Instantiation: EUF

\[ \forall x. f(g(x)) = h(f(x)) \]

Build partial definitions

\[ \ldots, f(b) = b, \quad g(b) = a, f(a) = a, \quad h(f(a)) = d, h(b) = c \]

\[ a = g(b) = f(a) \]

\[ b = f(b) \]

\[ c = h(b) \]

\[ d = h(f(a)) \]
Conflict-Based Instantiation: EUF

E

\[
\begin{align*}
\ldots, f(b) &= b, \\
g(b) &= a, f(a) &= a, \\
h(f(a)) &= d, h(b) &= c
\end{align*}
\]

Q

\[\forall x. f(g(x)) = h(f(x))\]

Check entailment

\[E, Q, f(g(b)) = h(f(b)) \models_E f(g(b)) = h(f(b))\]
Conflict-Based Instantiation: EUF

\[ \forall x. f(g(x)) = h(f(x)) \]

\[ \ldots, f(b) = b, \quad g(b) = a, f(a) = a, \quad h(f(a)) = d, h(b) = c \]

\[ a = g(b) = f(a) \]
\[ b = f(b) \]
\[ c = h(b) \]
\[ d = h(f(a)) \]

\[ E, Q, f(g(b)) = h(f(b)) \models_E a = c \]
Conflict-Based Instantiation: EUF

\[ E \subseteq Q, f(g(b)) = h(f(b)) \models_E a = c \]

Instance is not conflicting, but propagates an equality between two existing terms in \( E \)
Conflict-Based Instantiation: EUF

\[ E, Q, f(g(b)) = h(f(b)) \models_E a = c \]

\[ \forall x. f(g(x)) = h(f(x)) \]

\[ \ldots, f(b) = b, g(b) = a, f(a) = a, h(f(a)) = d, h(b) = c, a = g(b) = f(a) \]

\[ a = g(b) = f(a), b = f(b), c = h(b), d = h(f(a)) \]

\[ f(g(b)) = h(f(b)) \text{ is a propagating instance for } (E, Q) \]

⇒ These are also useful
Conflict-Based Instantiation

Given:
- Set of ground T-literals \( E \)
- Quantified formulas \( Q \)

**Conflict-based instantiation:**
1. If there exists a *conflicting instance* \( \forall x. \Psi \{ x \rightarrow t \} \)
   - Returns \( \forall x. \Psi \Rightarrow \forall x. \Psi \{ x \rightarrow t \} \) only
2. If there exists *propagating instance(s)*, \( \forall x. \Psi \{ x \rightarrow t_i \} \) for \( i = 1, \ldots, n \)
   - Returns \( \forall x. \Psi_1 \Rightarrow \Psi_1 \{ x \rightarrow t_1 \}, \ldots, \forall x. \Psi_n \Rightarrow \Psi_n \{ x \rightarrow t_n \} \) only
3. Otherwise:
   - Returns “unknown” (and the quantifiers module will resort to E-matching)
Conflict-Based Instantiation: Impact

- Using conflict-based instantiation (cvc4+ci), require an order of magnitude fewer instances for showing “UNSAT” wrt E-matching alone.

(taken from [Reynolds et al FMCAD14], evaluation On SMTLIB, TPTP, Isabelle benchmarks)
Conflicting instances found on ~75% of rounds (IR)

Configuration **cvc4+ci**:
- Calls E-matching 1.5x fewer times overall
- As a result, returns 5x fewer instantiations

### Conflict-Based Instantiation: Impact

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IR</td>
<td>% IR</td>
<td># Inst</td>
</tr>
<tr>
<td>TPTP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cvc4</td>
<td>71,634</td>
<td>100.0</td>
<td>878,957,688</td>
</tr>
<tr>
<td>cvc4+ci</td>
<td>208,970</td>
<td>20.3</td>
<td>150,351,384</td>
</tr>
<tr>
<td>Isabelle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cvc4</td>
<td>6,969</td>
<td>100.0</td>
<td>119,008,834</td>
</tr>
<tr>
<td>cvc4+ci</td>
<td>21,756</td>
<td>22.4</td>
<td>28,196,846</td>
</tr>
<tr>
<td>SMT-LIB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cvc4</td>
<td>14,032</td>
<td>100.0</td>
<td>60,650,746</td>
</tr>
<tr>
<td>cvc4+ci</td>
<td>58,003</td>
<td>20.0</td>
<td>32,305,788</td>
</tr>
</tbody>
</table>
Conflict-Based Instantiation: Impact

- CVC4 with conflicting instances **cvc4+ci**
  - Solves the **most benchmarks** for TPTP and Isabelle
  - Requires almost an order of magnitude **fewer instantiations**

<table>
<thead>
<tr>
<th></th>
<th>TPTP</th>
<th></th>
<th>Isabelle</th>
<th></th>
<th>SMT-LIB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Inst</td>
<td>Solved</td>
<td>Inst</td>
<td>Solved</td>
<td>Inst</td>
</tr>
<tr>
<td><strong>cvc3</strong></td>
<td>5,245</td>
<td>627.0M</td>
<td>3,827</td>
<td>186.9M</td>
<td>3,407</td>
<td>42.3M</td>
</tr>
<tr>
<td><strong>z3</strong></td>
<td>6,269</td>
<td>613.5M</td>
<td>3,506</td>
<td>67.0M</td>
<td><strong>3,983</strong></td>
<td>6.4M</td>
</tr>
<tr>
<td><strong>cvc4</strong></td>
<td>6,100</td>
<td>879.0M</td>
<td>3,858</td>
<td>119.0M</td>
<td>3,680</td>
<td>60.7M</td>
</tr>
<tr>
<td><strong>cvc4+ci</strong></td>
<td><strong>6,616</strong></td>
<td><strong>150.9M</strong></td>
<td><strong>4,082</strong></td>
<td><strong>28.2M</strong></td>
<td>3,747</td>
<td>32.4M</td>
</tr>
</tbody>
</table>

⇒ *A number of hard benchmarks can be solved without resorting to E-matching at all*
Conflict-Based Instantiation: Challenges

• How do we *find* conflicting instances?
  • Idea: construct instances via a stronger version of matching
    • Intuition: for $\forall x. P(x) \lor Q(x)$, will only match $P(x)$ where $P(t) \iff \bot$
      (For technical details, see [Reynolds et al FMCAD2014])
• What about conflicts involving *multiple quantified formulas*?
• What if our quantified formulas that contain *theory symbols*?
Model-based Instantiation

- Implemented in solvers:
  - Z3 [Ge et al CAV09], CVC4 [Reynolds et al CADE13]

- Basic idea:
  1. Build interpretation $M$ for all uninterpreted functions in the signature
  2. If this interpretation satisfies all formulas in $Q$, answer “sat”
     - e.g. interpretation $f^M = \lambda x. 1$ satisfies $\forall x. f(x) > 0$

$\Rightarrow$ Ability *to answer “sat”*
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]

\[ \forall x. P(x) \lor R(x) \]

Ground Solver

E

\[ \neg P(a) \]
\[ P(b) \]
\[ \neg R(b) \]
\[ \neg R(c) \]

Q

\[ \forall x. P(x) \lor R(x) \]

MBQI

Conflict-Based

E-matching

Model-Based
Model-based Instantiation

\[ \forall x. P(x) \lor R(x) \]

\[ \lnot P(a), P(b), \lnot R(b), \lnot R(c) \]

Ground Solver

Build interpretation \( M \) of predicates

- This interpretation must satisfy \( E \)
Model-based Instantiation

\[ \forall x. P(x) \lor R(x) \]

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]

Ground Solver

Build interpretation \( M \) of predicates

- This interpretation must satisfy \( E \)
- Missing values may be filled in arbitrarily
Model-based Instantiation

- \( \forall x. P(x) \lor R(x) \)
- \( \neg P(a), P(b), \neg R(b), \neg R(c) \)

Does \( M \) satisfy \( Q \)?

- Check (un)satisfiability of: \( \exists x. \neg (P^M(x) \lor R^M(x)) \)
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]

Check: \[ \exists x. \neg (P^M(x) \lor R^M(x)) \]
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]

Check: \[ \neg \left( P^M(k) \lor R^M(k) \right) \]

\[ \Rightarrow \text{Skolemize} \]
Model-based Instantiation

$$\neg P(a), P(b), \neg R(b), \neg R(c)$$
$$\forall x. P(x) \lor R(x)$$

Check: $$\neg (\text{ite}(k=a,\bot,\text{ite}(k=b,T,T)) \lor \text{ite}(k=b,\bot,\text{ite}(k=c,\bot,\bot)))$$
Model-based Instantiation

\( \forall x. P(x) \lor R(x) \)

\( \neg P(a), P(b), \neg R(b), \neg R(c) \)

Ground Solver

\( \neg P(a) \)
\( P(b) \)
\( \neg R(b) \)
\( \neg R(c) \)

Check: \( \neg (k \neq a \lor \bot) \)

\( \neg P(a), P(b), \neg R(b), \neg R(c) \)

\( \forall x. P(x) \lor R(x) \)

MBQI

\( P^M \Leftrightarrow \lambda x. \)
\( \text{ite} (x = a, \bot, \text{ite} (x = b, T, T)) \)

\( R^M \Leftrightarrow \lambda x. \)
\( \text{ite} (x = b, \bot, \text{ite} (x = c, \bot, \bot)) \)

\( \Rightarrow \) Simplify

Conflict-Based

E-matching

Model-Based
Model-based Instantiation

$$\forall x. P(x) \lor R(x)$$
$$\neg P(a)$$
$$\neg P(b)$$
$$\neg R(b)$$
$$\neg R(c)$$

Ground Solver

$$P^M \iff \lambda x. \text{ite}(x=a, \bot, \text{ite}(x=b, T, \text{ite}(x=c, \bot, T)))$$

$$R^M \iff \lambda x. \text{ite}(x=b, \bot, \text{ite}(x=c, \bot, \bot)))$$

Check: \( k=a \)
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]

\[ \forall x. P(x) \lor R(x) \]

\[ \forall x. P(x) \lor R(x) \]

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]

\[ \forall x. P(x) \lor R(x) \]

Check: \( k=a \)

⇒ Satisfiable! There are values \( k \) for which \( M \) does not satisfy \( Q \)
Model-based Instantiation

\[
\neg P(a), P(b), \neg R(b), \neg R(c)
\]
\[
\forall x. P(x) \lor R(x)
\]

Ground Solver

\[
\neg P(a), P(b), \neg R(b), \neg R(c)
\]
\[
\forall x. P(x) \lor R(x)
\]

MBQI

Return

(\forall x. P(x) \lor R(x)) \Rightarrow P(a) \lor R(a)

\Rightarrow \text{Add one instance for one such value of } k \text{ for which } M \text{ did satisfy } Q

Check: k=a
Model-based Instantiation

- \neg P(a), P(b), \neg R(b), \neg R(c)
- \forall x. P(x) \lor R(x)
- \neg (\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a)

Ground Solver

MBQI

E

- \neg P(a)
- P(b)
- \neg R(b)
- \neg R(c)

Q

- \forall x. P(x) \lor R(x)
Model-based Instantiation

\[ \exists x. P(x) \lor R(x) \]

\[ \neg P(a) \]

\[ \neg P(b) \]

\[ \neg R(b) \]

\[ \neg R(c) \]

\[ \neg R(a) \]

\[ E' \]

\[ Q' \]

\[ \forall x. P(x) \lor R(x) \]

\[ \neg (\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a) \]

\[ \neg (\forall x. P(x) \lor R(x)) \lor P(c) \lor R(c) \]

Ground Solver

MBQI

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]

\[ \forall x. P(x) \lor R(x) \]

• Repeat as necessary

⇒ “Model refinement loop”
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]

\[ \forall x. P(x) \lor R(x) \]

\[ \neg (\forall x. P(x) \lor R(x)) \lor \neg P(a) \lor \neg R(a) \]

\[ \neg (\forall x. P(x) \lor R(x)) \lor \neg P(c) \lor \neg R(c) \]
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(c) \lor R(c) \]

Ground Solver

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(c) \lor R(c) \]

M''

Check: \[ \exists x. \neg (P^{M''}(x) \lor R^{M''}(x)) \]
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(c) \lor R(c) \]

Ground Solver

MBQI

Check: \( k = a \land k \neq a \)
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor \neg P(a) \lor \neg R(a) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor \neg P(c) \lor \neg R(c) \]

**Ground Solver**

Check: \( k=\text{a} \land k\neq\text{a} \)

\( \Rightarrow \) Unsatisfiable, there are no values \( k \) for which \( M'' \) does not satisfy \( Q'' \)
Model-based Instantiation

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(c) \lor R(c) \]

\[ \neg P(a), P(b), \neg R(b), \neg R(c) \]
\[ \forall x. P(x) \lor R(x) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a) \]
\[ \neg (\forall x. P(x) \lor R(x)) \lor P(c) \lor R(c) \]

\[ \lambda x. \]
\[ \text{ite}(x=a, \bot, \text{ite}(x=b, T, \text{ite}(x=c, T, T))) \]

\[ \lambda x. \]
\[ \text{ite}(x=a, T, \text{ite}(x=b, \bot, \text{ite}(x=c, \bot, \bot))) \]
Finite Model Finding in CVC4

In CVC4, model-based Instantiation used for improving scalability of FMF
Finite Model Finding in CVC4

Model-Based E-matching Conflict-Based

Enumerate these instances lazily, via a:

\[ \text{Terminating: enumerate at most } n \text{ instances} \]

\[ \text{Efficient in practice: typically terminates after } m \ll n \text{ instances} \]

\[ \forall x : U . \ P(x) \lor R(x) \]

\[ \neg P(a) \]
\[ P(b) \]
\[ \neg R(b) \]
\[ R(c) \]
\[ P(a) \]
\[ P(c) \]

\[ \neg P(a) \]
\[ P(b) \]
\[ \neg R(b) \]
\[ R(c) \]
\[ \neg R(a) \]
\[ \neg P(c) \]

\[ U^M = \{ a, b, c, d \} \]

\[ P^M \leftrightarrow \ldots \]

\[ R^M \leftrightarrow \ldots \]

\[ \forall x : U . \ P(x) \lor R(x) \]

\[ \ldots \lor P(a) \lor R(a) \]
\[ \ldots \lor P(b) \lor R(b) \]
\[ \ldots \lor P(c) \lor R(c) \]
\[ \ldots \lor P(d) \lor R(d) \]

\[ Q^M \]
\[ a \ b \ c \ d \]

via exhaustive instantiation
Finite Model Finding in CVC4

Enumerate these instances lazily, via a:

- Terminating: enumerate at most \( n \) instances
- Efficient in practice: typically terminates after \( m \ll n \)

\[
\forall x: U. P(x) \lor R(x)
\]

\[
\neg P(a) \quad P(b) \\
\neg R(b) \quad R(a) \\
\neg R(c) \quad P(c) \\
\neg R(a) \quad \neg P(c) \\
\]

via model-based instantiation

\[
Q^M = \{ a, b, c, d \} \\
P^M \equiv \ldots \\
R^M \equiv \ldots \\
\]

\[
Q = \top \quad T \quad \bot \quad T
\]
Model-based Instantiation: Impact

- 1203 satisfiable benchmarks from the TPTP library
- Graph shows # instances required by exhaustive instantiation
  - E.g. $\forall x,y,z : U \cdot P(x, y, z)$, if $|U|=4$, requires $4^3=64$ instances
Model-based Instantiation: Impact

- CVC4 Finite Model Finding + Exhaustive instantiation
  - Scales only up to ~150k instances with a 30 sec timeout
Model-based Instantiation: Impact

- CVC4 Finite Model Finding + Model-Based instantiation \cite{ReynoldsetalCADE13}
  - Scales to >2 billion instances with a 30 sec timeout, only adds fraction of possible instances
E-matching, Conflict-Based, Model-based:

- **Common thread:** satisfiability of $\forall + UF +$ theories is hard!
  - E-matching:
    - Pattern selection, matching modulo theories
  - Conflict-based:
    - Matching is incomplete, entailment tests are expensive
  - Model-based:
    - Models are complex, interpreted domains (e.g. Int) may be infinite
E-matching, Conflict-Based, Model-based:

- **Common thread:** satisfiability of \( \forall + UF + \) theories is hard!
  - E-matching:
    - Pattern selection, matching modulo theories
  - Conflict-based:
    - Matching is incomplete, entailment tests are expensive
  - Model-based:
    - Models are complex, interpreted domains (e.g. Int) may be infinite

⇒ But reasoning about \( \forall + \text{pure theories isn’t as bad:} \)
  - Classic \( \forall \)-elimination algorithms are decision procedures for \( \forall \) in:
    - LRA [Ferrante+Rackoff 79, Loos+Wiespfenning 93], LIA [Cooper 72], datatypes, ...
E-matching, Conflict-Based, Model-based:

• **Common thread:** satisfiability of $\forall + UF +$ theories is hard!
  
  • E-matching:
    • Pattern selection, matching modulo theories
  
  • Conflict-based:
    • Matching is incomplete, entailment tests are expensive
  
  • Model-based:
    • Models are complex, interpreted domains (e.g. Int) may be infinite

$\Rightarrow$ But reasoning about $\forall + pure$ theories isn’t as bad:

• Classic $\forall$-elimination algorithms are decision procedures for $\forall$ in:
  
  • LRA [Ferrante+Rackoff 79, Loos+Wiespfenning 93], LIA [Cooper 72], datatypes, ...

• Can classic $\forall$-elimination algorithms be leveraged in an DPLL(T) context?
  
Techniques for Quantifier Instantiation

Ground Solver

Instances of \( \forall \) in \( Q \)

Satisfying assignment \( E, Q \)

\[ E \cup Q \text{ is } T\text{-satisfiable} \]

Quantifiers Module

- Conflict-Based
- E-matching
- Model Based

Generally, used for quantifiers with UF

CE-Guided

Generally, used for quantifiers w/o UF

unsat

sat
Techniques for Quantifier Instantiation

Ground Solver

Quantifiers Module

- Conflict-Based
- E-matching
- Model Based

CE-Guided

Instances of $\forall$ in $Q$

A decision procedure for $\forall$ in LIA, LRA, ...

$E \cup Q$ is T-satisfiable

$\Rightarrow$ Classic $\forall$-elimination algorithms can be cast as counterexample-guided instantiation procedures

$sat$

$unsat$

$F, \ldots$

Satisfying assignment $E, Q$

Classic $\forall$-elimination algorithms can be cast as counterexample-guided instantiation procedures

⇒ Classic $\forall$-elimination algorithms can be cast as counterexample-guided instantiation procedures
Counterexample-Guided Instantiation

• Variants implemented in number of tools:
  • Z3 [Bjorner 2012, Bjorner/Janota 2016]
  • Yices [Dutertre 2015]
  • CVC4 [Reynolds et al 2015]

• High-level idea:
  • Quantifier elimination (e.g. for LIA) says: \( \exists x. \psi[x] \Leftrightarrow \psi[t_1] \lor \ldots \lor \psi[t_n] \) for finite \( n \)
Counterexample-Guided Instantiation

• Variants implemented in number of tools:
  - Z3 [Bjorner 2012, Bjorner/Janota 2016]
  - Yices [Dutertre 2015]
  - CVC4 [Reynolds et al 2015]

• High-level idea:
  • Quantifier elimination (e.g. for LIA) says: \( \forall x. \neg \psi[x] \iff \neg \psi[t_1] \land \ldots \land \neg \psi[t_n] \) for finite \( n \)
    (consider the dual)
Counterexample-Guided Instantiation

• Variants implemented in number of tools:
  • Z3 [Bjorner 2012, Bjorner/Janota 2016]
  • Yices [Dutertre 2015]
  • CVC4 [Reynolds et al 2015]

• High-level idea:
  • Quantifier elimination (e.g. for LIA) says: \( \forall x. \neg \psi[x] \Leftrightarrow \neg \psi[t_1] \land \ldots \land \neg \psi[t_n] \) for finite \( n \)
  • Enumerate these instances lazily, via a counterexample-guided loop, that is:
    • Terminating: enumerate at most \( n \) instances
    • Efficient in practice: typically terminates after \( m<<n \) instances
Consider $\forall$ in the theory of linear integer arithmetic LIA:

$$\exists abc. \ (a=b+5 \land \forall x. \ (x>a \lor x<b \lor x-c<3))$$
Consider $\forall$ in the theory of linear integer arithmetic LIA:

$$\exists a b c . \ (a = b + 5 \land \forall x . \ (x > a \lor x < b \lor x - c < 3))$$

- Outermost existentials $a, b, c$ are treated as free constants
Counterexample-Guided Instantiation

Ground Solver

\[ a = b + 5 \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

CE-Guided
Counterexample-Guided Instantiation

\[ a = b + 5 \]
\[ \forall x. \ (x > a \lor x < b \lor x - c < 3) \]

→ Use counterexample-guided instantiation
Counterexample-Guided Instantiation

\[ a = b + 5 \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

Counterexample-Guided Instantiation

\[ \exists k. \neg (k > a \lor k < b \lor k - c < 3) \]

⇒ With respect to model-based instantiation:

• Similar: check satisfiability of \( \exists k. \neg (k > a \lor k < b \lor k - c < 3) \)
With respect to model-based instantiation:

• Similar: check satisfiability of $\exists k. \neg (k > a \lor k < b \lor k - c < 3)$

• Key difference: use the same (ground) solver for $F$ and counterexample $k$ for $Q$
Counterexample-Guided Instantiation

- $a = b + 5$
- $\forall x. (x > a \lor x < b \lor x - c < 3)$
- $C \Rightarrow (k \leq a \land k \geq b \land k \geq c + 3)$
Counterexample-Guided Instantiation

\[ a = b + 5 \]

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

G \implies (k \leq a \land k \geq b \land k \geq c + 3)

C \implies (k \leq a \land k \geq b \land k \geq c + 3)

C is a fresh Boolean variable: "A counterexample k exists for \( \forall x. (x > a \lor x < b \lor x - c < 3) \)"
Counterexample-Guided Instantiation

- Three cases:
  
  \[ a = b + 5, \ldots, \ \forall x. (x > a \lor x < b \lor x - c < 3) \]
  
  \[ C \Rightarrow (k \leq a \land k \geq b \land k \geq c + 3) \]

Ground Solver

CE-Guided Instantiation

CE-Guided

Solver

\[ F \]

instances
Counterexample-Guided Instantiation

• Three cases:
  1. $F$ is unsatisfiable

$F$ is unsatisfiable
Counterexample-Guided Instantiation

• Three cases:

1. $F$ is unsatisfiable  
   $\neg C \in \neg \text{sat}$

2. $F$ is satisfiable, $\neg C \in E$ for all assignments $E$  
   $\Rightarrow$ answer “sat”
Counterexample-Guided Instantiation

\[ a=b+5, \ldots, \]
\[ \forall x (x>a \lor x<b \lor x-c<3) \]
\[ C \implies (k \leq a \land k \geq b \land k \geq c+3) \]

CE-Guided

Ground Solver

\[ \forall x (x>a \lor x<b \lor x-c<3) \]

Three cases:

1. \( F \) is unsatisfiable \( \implies \) \( a \) Ũs ŋ Đ ͞ unsat \( \)
2. \( F \) is satisfiable, \( \neg G \in E \) for all assignments \( E \) \( \implies \) answer \( \) sat \( \)
3. \( F \) is satisfiable, \( C \in E \) for some assignment \( E \) \( \implies \) add an instance to \( F \)

\[ \exists t (t>a \lor t<b \lor t-c<3) \]

where \( k \in \text{FV}(t) \)

3 . \( F \) is satisfiable, \( C \in E \) for some assignment \( E \) \( \implies \) add an instance to \( F \)
Counterexample-Guided Instantiation

- Three cases:
  1. \( F \) is unsatisfiable
  2. \( F \) is satisfiable, \( \neg C \in E \) for all assignments \( E \)
  3. \( F \) is satisfiable, \( C \in E \) for some assignment \( E \)

\[
a = b + 5, \ldots, \\
\forall x. (x > a \lor x < b \lor x - c < 3) \\
C \Rightarrow (k \leq a \land k \geq b \land k \geq c + 3)
\]
Counterexample-Guided Instantiation

- Three cases:
  1. \( F \) is unsatisfiable
  2. \( F \) is satisfiable, \( \neg C \in E \) for all assignments \( E \)
  3. \( F \) is satisfiable, \( C \in E \) for some assignment \( E \)

\[
a = b + 5, \ldots, \\
\forall x. (x > a \lor x < b \lor x - c < 3) \\
C \Rightarrow (k \leq a \land k \geq b \land k \geq c + 3)
\]
Counterexample-Guided Instantiation

Ground Solver

\[ a = b + 5 \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]
\[ C \Rightarrow (k \leq a \land k \geq b \land k \geq c + 3) \]
Counterexample-Guided Instantiation

c, a = b + 5,
k ≤ a
k ≥ b
k ≥ c + 3

∀x. (x > a ∨ x < b ∨ x - c < 3)

CE-Guided Instantiation

E

Q

∀x. (x > a ∨ x < b ∨ x - c < 3)

CE-Guided

Ground Solver

CEGQI

CE-Guided
Counterexample-Guided Instantiation

\[ a = b + 5 \]

\( \forall x. (x > a \lor x < b \lor x - c < 3) \)

\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]

C, \( a = b + 5 \),
\( k \leq a \)
\( k \geq b \)
\( k \geq c + 3 \)

Build model \( M \) for \( E \)

Ground Solver

CE-GQI

\( a^M = 5 \)
\( b^M = 0 \)
\( c^M = 0 \)
\( k^M = 3 \)
Counterexample-Guided Instantiation

\[ a = b + 5 \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]
\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]

**Ground Solver**

- \( C, a = b + 5, k \leq a \)
- \( k \geq b \)
- \( k \geq c + 3 \)

**CEGQI**

- \( a^M = 5 \)
- \( b^M = 0 \)
- \( c^M = 0 \)
- \( k^M = 3 \)

**Take lower bounds of \( k \) in \( E \)**
Counterexample-Guided Instantiation

\[ a = b + 5 \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]
\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]

Compute their value in \( M \):
\[ a^M = 5 \]
\[ b^M = 0 \]
\[ c^M = 0 \]
\[ k^M = 3 \]
Counterexample-Guided Instantiation

\[ a = b + 5 \]

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]

\[ C, a = b + 5, \]

\[ k \leq a \]

\[ k \geq b \]

\[ k \geq c + 3 \]

Add instance for lower bound that is maximal in \( M \)
Counterexample-Guided Instantiation

Ground Solver

C, a=b+5, k≤a, k≥b, k≥c+3

CEGQI

∀x. (x>a ∨ x<b ∨ x-c<3)

CE-Guided

a=b+5
∀x. (x>a ∨ x<b ∨ x-c<3)
¬C ∨ (k≤a ∧ k≥b ∧ k≥c+3)

C,a=b+5, k≤a, k≥b, k≥c+3

E

Q

∀x. (x>a ∨ x<b ∨ x-c<3) ∴ c+3>a ∨ c+3<b

a^M=5
b^M=0
c^M=0
k^M=3

in M

k≥b = 0
k≥c+3 = 3

∀x. (x>a ∨ x<b ∨ x-c<3) ⇒ c+3>a ∨ c+3<b
Counterexample-Guided Instantiation

Ground Solver

CEGQI

\[ a = b + 5 \]
\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor c + 3 > a \lor c + 3 < b \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]
\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]
Counterexample-Guided Instantiation

\[ a = b + 5, \ c + 3 < b, \ k \leq a, \ k \geq b, \ k \geq c + 3 \]

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

\[ \neg (a = b + 5) \]

\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor c + 3 > a \lor c + 3 < b \]

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]
Counterexample-Guided Instantiation

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

Ground Solver

CE-Guided

E

\[ C, a = b + 5, c + 3 < b, \]
\[ k \leq a \]
\[ k \geq b \]
\[ k \geq c + 3 \]

CEGQI

Q

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

Build model \( M \) for \( E \)

\[ a^M = 5 \]
\[ b^M = 0 \]
\[ c^M = -4 \]
\[ k^M = 3 \]

\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor c + 3 > a \lor c + 3 < b \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]
\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]
Counterexample-Guided Instantiation

\( \forall x. (x > a \lor x < b \lor x - c < 3) \lor (c + 3 > a \lor c + 3 < b) \)

\( \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor (c + 3 > a \lor c + 3 < b) \)

\( \forall x. (x > a \lor x < b \lor x - c < 3) \lor (c + 3 > a \lor c + 3 < b) \)

Take lower bounds of \( k \) in \( E \)

\( C, a = b + 5, c + 3 < b, k \leq a, k > b, k \geq c + 3 \)

\( a^M = 5, b^M = 0, c^M = -4, k^M = 3 \)
Counterexample-Guided Instantiation

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

Ground Solver

CE-Guided

\[ a = b + 5 \]

\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor c + 3 > a \lor c + 3 < b \]

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

\[ \neg C \lor (\neg a \land k \geq b \land k \geq c + 3) \]

\[ C, a = b + 5, c + 3 < b, k \leq a, k \geq b, k \geq c + 3 \]

CEGQI

\[ a^M = 5 \]

\[ b^M = 0 \]

\[ c^M = -4 \]

\[ k^M = 3 \]

\[ k \geq b = 0 \]

\[ k \geq c + 3 = -1 \]

Compute their value in M
Counterexample-Guided Instantiation

Counterexample-Guided Instantiation

Ground Solver

\[ \forall x. (x>a \lor x<b \lor x-c<3) \]

CEGQI

\[ a=b+5 \]

\[ \neg \forall x. (x>a \lor x<b \lor x-c<3) \land c+3>a \land c+3<b \]

\[ \forall x. (x>a \lor x<b \lor x-c<3) \land \neg C \lor (k\leq a \land k\geq b \land k\geq c+3) \]

\[ C, a=b+5, c+3<b, k\leq a, k\geq b, k\geq c+3 \]

\[ E, Q, a^M=5, b^M=0, c^M=-4, k^M=3 \]

Add instance for lower bound that is maximal in \( M \)

\[ \forall x. (x>a \lor x<b \lor x-c<3) \Rightarrow b>a \lor b<b \lor b-c<3 \]
Counterexample-Guided Instantiation

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

Ground Solver

CE-Guided

\[ \exists \, x. (x > a \lor x < b \lor x - c < 3) \]

\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \]

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

CEGQI

\[ a = b + 5 \]

\[ c = -4 \]

\[ k = 3 \]

\[ b = 0 \]

\[ c = -4 \]

\[ k = -1 \]

Add instance for lower bound that is maximal in \( M \)

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \Rightarrow b > a \lor b - c < 3 \]
Counterexample-Guided Instantiation

\[ a = b + 5 \]
\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor c + 3 > a \lor c + 3 < b \]
\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor b > a \lor b < c + 3 \]
\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]
\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]
Counterexample-Guided Instantiation

Ground Solver

CEGQI

CE-Guided

\[ a = b + 5 \]

\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor c + 3 > a \lor c + 3 < b \]

\[ \neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor b > a \lor b < c + 3 \]

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

\[ \neg C \lor (k \leq a \land k \geq b \land k \geq c + 3) \]
Counterexample-Guided Instantiation

Ground Solver

\( \lnot C \)
\( a=b+5 \)
\( c+3<a \)
\( b<c+3 \)

CE-GQI

\( \forall x. (x>a \lor x<b \lor x-c<3) \)
\( \neg C \lor (k \leq a \land k \geq b \land k \geq c+3) \)
Counterexample-Guided Instantiation

\[ \forall x. (x > a \lor x < b \lor x - c < 3) \]

Ground Solver

CE-Guided

E

\neg C
a = b + 5
c + 3 < a
b < c + 3

Q

\forall x. (x > a \lor x < b \lor x - c < 3)

CEGQI

\neg C

\frac{a = b + 5}{\neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor c + 3 > a \lor c + 3 < b}

\frac{\neg \forall x. (x > a \lor x < b \lor x - c < 3) \lor b > a \lor b < c + 3}{\forall x. (x > a \lor x < b \lor x - c < 3)}

\frac{\neg C \lor (k \leq a \land k \geq b \land k \geq c + 3)}{a \ b \ c + 3 \ x > a \ x < b \ x < c + 3 \ x > a \lor x < b \lor x - c < 3}
Counterexample-Guided Instantiation

Ground Solver

CE-Guided

Ground

Solver

CE-Guided

E

¬C

da = b + 5

c + 3 < a

b < c + 3

Q

∀x.(x > a ∨ x < b ∨ x - c < 3)

⇒ ∃abc.(a = b + 5 ∧ ∀x.(x > a ∨ x < b ∨ x - c < 3))

is LIA-satisfiable
Counterexample-Guided Instantiation

• Decision procedure for $\forall$ in various theories:
  • Linear real arithmetic (LRA)
    • Maximal lower (minimal upper) bounds
      • [Loos+Wiespfenning 93]
    • Interior point method:
      • [Ferrante+Rackoff 79]
  • Linear integer arithmetic (LIA)
    • Maximal lower (minimal upper) bounds ($+c$)
      • [Cooper 72]
  • Bitvectors/finite domains
    • Value instantiations
  • Datatypes, ...

$\Rightarrow$ Termination argument for each: enumerate at most a finite number of instances

$l_1<k, \ldots, l_n<k \rightarrow \{x \rightarrow l_{\text{max}} + \delta \}$

...may involve virtual terms $\delta \in \mathbb{R}$

$l_{\text{max}} < k < u_{\text{min}} \rightarrow \{x \rightarrow (l_{\text{max}} - u_{\text{min}}) / 2 \}$

$F[k] \rightarrow \{x \rightarrow k^M\}$
Summary: DPLL(T)+Instantiation

T-clauses $F$

Ground Solver
- SAT Solver
- T-Decision Procedures

Lemmas

Quantifiers Module
- Conflict-Based
- E-matching
- CE-Guided
- Model-Based

unsat

ground literals $E$

$\forall$ formulas $Q$

sat
Summary: DPLL(T)+Instantiation

Ground Solver
unsat
T-clauses $F$
Quantifiers Module
ground literals $E$
$\forall$ formulas $Q$

CVC4

T-Decision Procedures
SAT Solver

Conflict-Based
E-matching
Model-Based
CE-Guided
Future Challenges

• Improve performance and precision of existing approaches
  • Many engineering challenges when implementing E-matching, conflict-based instantiation

• Develop new approaches for \( \forall + \text{UF} \) theories that:
  • Are efficient in practice
    • E-matching is efficient for \( \forall + \text{UF} \), ce-guided approaches are efficient for \( \forall \) theories
      • Under what conditions, and to what degree, can these techniques be combined?

• Are decision procedures for various fragments
  • Extensions of Bernays-Shonfinkel
  • Array Property fragments
  • Local theory extensions
  • \( \forall \) over pure theories that emit quantifier elimination
Thanks for listening

• CVC4: