Instantiation for Quantified Formulas in SMT: Techniques and Practical Aspects

Andrew Reynolds
June 24, 2016
In this Talk

\((\forall x. P(x) \lor f(b) = b+1) \land \exists y. (\neg P(y) \land f(y) < y)\)

• Focus on techniques for establishing \textit{T-satisfiability} of formulas with:
In this Talk

\((\forall x. P(x) \lor f(b) = b+1) \land \exists y. (\neg P(y) \land f(y) < y)\)

- Focus on techniques for establishing *T-satisfiability* of formulas with:
  - **Boolean structure**
In this Talk

$$(\forall x. P(x) \lor f(b)=b+1) \land \exists y. (\neg P(y) \land f(y)<y)$$

- Focus on techniques for establishing \textit{T-satisfiability} of formulas with:
  - **Boolean structure**
  - Constraints in a background theory T, e.g. UFLIA
In this Talk

$$(\forall x. P(x) \lor f(b) = b+1) \land \exists y. (\neg P(y) \land f(y) < y)$$

• Focus on techniques for establishing \textit{T-satisfiability} of formulas with:
  • Boolean structure
  • Constraints in a background theory T, e.g. \textbf{UFLIA}
  • \textit{Existential and Universal Quantifiers}
Outline

• Background
• 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:
  • E-matching, conflict-based, model-based, counterexample-guided
• Challenges, future work
Quantified formulas $\forall$ in SMT

• Are of importance to applications:
  • Automated theorem proving:
    • Background axioms $\{\forall x. g(e,x) = g(x,e) = x, \forall x. g(x,g(y,z)) = g(g(x,y),x), \forall x. g(x,i(x)) = e\}$
  • Software verification:
    • Unfolding $\forall x. foo(x) = bar(x+1)$, code contracts $\forall x. \text{pre}(x) \Rightarrow \text{post}(f(x))$
    • Frame axioms $\forall x. x \neq t \Rightarrow A'(x) = A(x)$
  • Function Synthesis:
    • Conjectures $\forall i: \text{input}. \exists o: \text{output}. R[o,i]$
  • Planning:
    • Specifications $\exists p: \text{plan}. \forall t: \text{time}. F[P,t]$
Quantified formulas \( \forall \) in SMT

- Are of importance to **applications**:
  - Automated theorem proving:
    - Background axioms \( \forall x.g(e,x)=g(x,e)=x, \forall x.g(x,g(y,z))=g(g(x,y),x), \forall x.g(x,i(x))=e \)
  - Software verification:
    - Unfolding \( \forall x.\text{foo}(x)=\text{bar}(x+1) \), code contracts \( \forall x.\text{pre}(x) \Rightarrow \text{post}(f(x)) \)
    - Frame axioms \( \forall x.x\neq t \Rightarrow A'(x)=A(x) \)
  - Function Synthesis:
    - Conjectures \( \forall i:\text{input.}\exists o:\text{output.R}[o,i] \)
  - Planning:
    - Specifications \( \exists p:\text{plan.}\forall t:\text{time.F}[P,t] \)

- Are very challenging in **theory**:
  - Establishing T-satisfiability of formulas with \( \forall \) is generally undecidable
Quantified formulas $\forall$ in SMT

- Are of importance to applications:
  - Automated theorem proving:
    - Background axioms \( \forall x. g(e, x) = g(x, e) = x, \forall x. g(x, g(y, z)) = g(g(x, y), x), \forall x. g(x, i(x)) = e \)  
  - Software verification:
    - Unfolding \( \forall x. \text{foo}(x) = \text{bar}(x+1) \), code contracts \( \forall x. \text{pre}(x) \Rightarrow \text{post}(f(x)) \)  
    - Frame axioms \( \forall x. x \neq t \Rightarrow A'(x) = A(x) \)  
  - Function Synthesis:
    - Conjectures \( \forall i: \text{input}. \exists o: \text{output}. R[o, i] \)  
    - Planning:
      - Specifications \( \exists p: \text{plan}. \forall t: \text{time}. F[P, t] \)  
- Are very challenging in theory:
  - Establishing T-satisfiability of formulas with $\forall$ is generally undecidable  
- Can be handled well in practice:
  - Efficient decision procedures for decidable fragments, e.g. Bernays-Shonfinkel  
  - Heuristic techniques have high success rates in the general case
Background: Theory

- A *theory* $T$ is a pair $(\Sigma_T, I_T)$, where:
  - $\Sigma_T$ is set of function symbols, the *signature* of $T$
    - E.g. $\Sigma_{\text{LIA}} = \{+, -, <, \leq, >, \geq, 0, 1, 2, 3, \ldots\}$
  - $I_T$ is a set of *interpretations*
    - E.g. each $I \in I_{\text{LIA}}$ interpret functions in $\Sigma_{\text{LIA}}$ in standard way:
      - $1+1=2, 1+2=3, 1>0 = T, 0>1 = \perp$, ...
    - Interpretation of free constants chosen arbitrarily

- A formula $\Psi$ is $T$-satisfiable if there is an $I \in I_T$ that interprets $\Psi$ as $T$
  - We call $I$ a *model* of $\Psi$
    - E.g the formula $(a+1>b)$ is LIA-satisfiable with a model $I$ where $I(a) = 2$ and $I(b) = 0$
Background: Quantifiers

- **Universal** quantification:
  \[ \forall x: \text{Int.} . P(x) \]
  
  \( P \) is true for all integers \( x \)

- **Existential** quantification:
  \[ \exists x: \text{Int.} . \neg Q(x) \]
  
  \( Q \) is false for some integer \( x \)
Background: Quantifiers

• Universal quantification:

\[ \forall x: \text{Int.} \ P(x) \]

\( P \) is true for all integers \( x \)

• Existential quantification:

\[ \exists x: \text{Int.} \ \neg Q(x) \rightarrow \neg \forall x: \text{Int.} \ Q(x) \]

\( \Rightarrow \) For consistency, assume existential quantification is rewritten as universal quantification
Theoretical Complexity

- Checking T-satisfiability of:

  $$(\forall x. P(x) \lor Q(x) \lor x=a) \land P(b) \land Q(c)$$

- Bernays-Shonfinkel (function-free + equality) is **decidable** \(\text{(NEXPTIME)}\)

  $$(\forall xy. \exists z. x+y+z>2 \lor 0 \leq z+x)$$

- Case of \(\forall\) in pure theories is often **decidable**, e.g. linear arithmetic

  $$(\forall x. P(x) \Rightarrow P(x+1)) \land P(a) \land \neg P(b) \land a<b$$

- However, general case is **undecidable**!
Approaches for Satisfiability of ∀ in Tools

• First order theorem provers focus on ∀ 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 ∀ 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**
    - First-order resolution + superposition [Robinson 65, Nieuwenhuis/Rubio 99, Prevosto/Waldman 06]
    - AVATAR in Vampire [Voronkov 14, Reger et al 15]
  - **iProver**
    - InstGen calculus [Ganzinger/Korovin 03]
  - **Princess, Beagle, ...**

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

• First order theorem provers focus on ∀ reasoning
  ...but have been extended in the past decade to theory reasoning:
  • **Vampire, E, SPASS**
    • First-order resolution + superposition [Robinson 65, Nieuwenhuis/Rubio 99, Prevosto/Waldman 06]
    • AVATAR in Vampire [Voronkov 14, Reger et al 15]
  • **iProver**
    • InstGen calculus [Ganzinger/Korovin 03]
  • **Princess, Beagle, ...**

• SMT solvers focus mostly on ground theory reasoning
  ...but have been extended in the past decade to ∀ 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 07, ...]
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**
    - First-order resolution + superposition [Robinson 65, Nieuwenhuis/Rubio 99, Prevosto/Waldman 06]
    - AVATAR in Vampire [Voronkov 14, Reger et al 15]
  - **iProver**
    - InstGen calculus [Ganzinger/Korovin 03]
  - **Princess, Beagle, ...**

• 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, ...]

⇒ Focus of this talk
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

SAT Solver

- 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

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

SAT Solver

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

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

⇒ 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) ∨ f(b) > a + 1)
(¬∀x.P(x) ∨ ∀y.¬P(y) ∨ R(y))
(∀x.f(x) = g(x) + h(x) ∨ ¬P(a))

SAT Solver

UF-Solver
P(a)

LIA-Solver
f(b) > a + 1

Quantifiers Module

⇒ Distribute ground literals to T-solvers, ∀ literals to quantifiers module
Quantified Formulas in DPLL(T): Basics

SAT Solver

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

\( P(a) \lor f(b) > a + 1 \)

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

\( \forall x. f(x) = g(x) + h(x) \lor \neg P(a) \)

\( \neg \exists x. P(x) \)

\( \forall x. f(x) = g(x) + h(x) \lor \forall y. \neg P(y) \lor R(y) \)

\( P(a) \)

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

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

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

⇒ 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 $M$
- $T_1$-solver
- $T_n$-solver

...when $F$ is propositionally unsatisfiable

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

Conflicts, lemmas

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

SAT Solver

T-Clauses $F$

Satisfying Assignment $M$

$T_1$-solver

$T_n$-solver

Quantifiers Module

$M_1$

$M_n$

$Q$

... when $F$ is propositionally unsatisfiable

... when $M$ is $T_1+...+T_n$-satisfiable

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

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

Unlike the ground case where decision procedures exist for $T_1, \ldots, T_n$, ...there is no general decision procedure for $\forall$-formulas $Q$, thus:

- This procedure may not terminate!
- Regardless, we want techniques that:
  - Are refutation-sound ("unsat" can be trusted)
  - Are model-sound ("sat" can be trusted)
  - Terminate for many $F$
In this talk: DPLL(T)+Quantifiers, simplified

\[ \Rightarrow \text{For purposes of this talk, partition } M \text{ into quantifier-free part } E, \text{ and set of } \forall \text{ formulas } Q \]
In this talk: DPLL(T)+Quantifiers, simplified

Theory solvers determine whether $E$ is T-(un)satisfiable

$E$ is T-satisfiable

Conflicts, lemmas

SAT Solver

Satisfying Assignment $M$

T-Clauses $F$

Quantifiers Module

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

SAT Solver

T-Clauses \( F \)

Satisfying Assignment \( M \)

Theory solver(s)

\( E \)

\( E \cup Q \) is T-satisfiable

Quantifiers Module

\( E \), \( E \) is T-satisfiable

Lemmas

\( Q \)

\( E \cup Q \) is T-satisfiable

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

SAT Solver

Satisfying Assignment \( M \)

Theory solver(s)

T-Clauses \( F \)

Lemmas

\[ E \]

Quantifiers Module

\[ E \cup Q \]

\( \Rightarrow \) The remainder of the talk will discuss how the quantifiers module is implemented
DPLL(T)+Quantifiers, further simplified

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

**Outputs:**
- $E \cup Q$ is T-satisfiable, or
- $F$ is T-satisfiable
- Set of lemmas to add to $F$
DPLL(T)+Quantifiers, further simplified

- **Ground Solver**
  - T-clauses $F$ → $\exists$ formulas $Q$
  - Inputs: Set of ground $T$-literals $E$
  - Outputs: $E \cup Q$ is $T$-satisfiable, or $F$ is $T$-satisfiable

- **Quantifiers Module**
  - Lemmas
  - Inputs: Set of $\forall$ formulas $Q$
  - Outputs: 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?
Which lemmas do we add: Basics

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

E

Q

Quantifiers Module
Which lemmas do we add: Basics

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

- Existential quantification (negated universals) handled by Skolemization
  - Introduce a fresh witness \( k \), lemma says \( \exists x. \neg P(x) \) implies \( \neg P(k) \)
  - Need only be applied once

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

...
Which lemmas do we add: Basics

Universal quantification handled by **Instantiation**

- Choose ground term(s) \( t \), lemma(s) say \( \forall x. f(x) = g(x) + h(x) \) implies \( f(t) = g(t) + h(t) \)

\[ \Rightarrow \text{May be applied ad infinitum!} \]
Quantifiers Module: Recurrent Questions

• 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]
  • …
Quantifiers Module: Recurrent Questions

• 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]
  • …

• How do we know $E \cup Q$ *is satisfiable*?
  • For some strategies and fragments, saturation $\Rightarrow E \cup Q$ is satisfiable
    • E.g. model-based, counterexample-guided
Quantifiers Module : Recurrent Questions

- 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]
  - ...

- How do we know $E \cup Q$ is satisfiable?
  - For some strategies and fragments, saturation $\Rightarrow E \cup Q$ is satisfiable
    - E.g. model-based, counterexample-guided

- *When* do we invoke the quantifiers module?
  - Eagerly, during the DPLL(T) search [Detlefs et al 03, deMoura/Bjorner CAV07], or
  - Lazily, only if $E \cup Q$ is a *complete* satisfying assignment
Techniques for Quantifier Instantiation: Overview

Ground Solver

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

Instances of $\forall$ in $Q$

Satisfying assignment $E, Q$

$F, \ldots$

$\text{unsat}$

$\text{sat}$
Techniques for Quantifier Instantiation: Overview

- Ground Solver
- \( F, \ldots \)
- Satisfying assignment \( E, Q \)
- Instances of \( \forall \) in \( Q \)

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

- \( \Rightarrow \) 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 [Detlefs 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

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

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

Q

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

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

\( \Rightarrow \text{Idea: choose instances based on pattern matching} \)
E-matching

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

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

Pattern

E-matching

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

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

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

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

E

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

Q

E-matching

Model Based

Conflict-Based
E-matching: Functions, Equality

In E-matching, Pattern **matching** takes into account equalities in \( E \)
E-matching: Functions, Equality

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

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

E

- $P(a, c)
- f(b) = a$

E-matching

Q

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

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

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

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

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

**E-matching**

**Congruence closure of \( E \)**
- \[ a = f(b) \]
- \[ b \]
- \[ c \]

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

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

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

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

\[ a = f(b) \]

\[ b \]

\[ c \]

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

E

P(a,c)
f(b)=a

Q

∀xy. P(f(x),y) ⇒ g(x)=y

E-matching

(∀xy. P(f(x),y) ⇒ g(x)=y) ⇒ P(f(b),c) ⇒ g(b)=c
E-matching

Given:
- Set of ground T-literals $E$
- Quantified formula $\forall \mathbf{x} . \Psi$, where $\mathbf{x}$ is a tuple of variables
- A pattern $p$ contain all variables in $\mathbf{x}$
- A ground term $g$ from $E$

Formally:
- We say $g$ matches $p$ modulo $E$ under the substitution $\{x \rightarrow t\}$ if $E \models_T g = p|_{x \rightarrow t}$
E-matching

Given:
- Set of ground T-literals $E$
- Quantified formula $\forall \mathbf{x}. \Psi$, where $\mathbf{x}$ is a tuple of variables
- A pattern $p$ contain all variables in $\mathbf{x}$
- A ground term $g$ from $E$

Formally:
- We say $g$ matches $p$ modulo $E$ under the substitution $\{x \mapsto t\}$ if $E \models T \ g = p\{x \mapsto t\}$
  
  usually restricted such that $T$ is theory of equality
E-matching

Given:
• Set of ground T-literals $E$
• Quantified formula $\forall x . \Psi$, where $x$ is a tuple of variables
• A pattern $p$ contain all variables in $x$
• A ground term $g$ from $E$

Formally:
• We say $g$ matches $p$ modulo $E$ under the substitution $\{x \mapsto t\}$ if $E \models T g \equiv p \{x \mapsto t\}$
• E-matching:
  1. Chooses (a set) of patterns $p_1, \ldots, p_m$ for $\forall x . \Psi$
  2. Computes sets of pairs $\{x \mapsto t_{j_1}, g_{j_1}\}, \ldots, \{x \mapsto t_{j_n}, g_{j_n}\}$ where $g_{j_i}$ matches $p_{j_i}$ modulo $E$
  3. Returns the instances $(\forall x . \Psi \Rightarrow \Psi \{x \mapsto t_{11}\}), \ldots, (\forall x . \Psi \Rightarrow \Psi \{x \mapsto t_{nm}\})$
E-matching: Intuition

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

$\Rightarrow Why$ $is$ $this$ $instance$ $useful?$
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 $(\forall x . \Psi \Rightarrow \Psi\{x \rightarrow t\})$
  \[ \Rightarrow \text{Why is this instance useful?} \]

• 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 $E$
E-matching: Intuition

• Say E-matching returns the instance ($\forall x. \Psi \Rightarrow \Psi\{x\rightarrow t\}$)  
  ⇒ 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 $g=p\{x\rightarrow t\}$ is implied by $E$
• Thus: $\Psi[g]$ is implied by $E \cup \{\Psi[p]\ \{x\rightarrow t\}\}$
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 $E$
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 \(E \cup \{\Psi[p] \{x \rightarrow t\}\}\)
  
  \(\Rightarrow In\ other\ words,\ from\ Q,\ we\ learn\ information\ \Psi[g]\ about\ a\ term\ g\ from\ E\)

  \[P(a, c) \Rightarrow g(b) = c\] is implied by

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

  \(E\) with new instance
From this instance, we learn \( g(b) = c \)
E-matching

\[ P(a,c) \]
\[ f(b) = a \]
\[ \forall x. g(x) \neq c \]
\[ \forall xy. P(f(x),y) \implies g(x) = y \]

\(~(\forall xy. P(f(x),y) \implies g(x) = y) \lor \neg P(f(b),c) \lor g(b) = c\)
\[
P(a, c) \\
f(b) = a \\
\forall x. g(x) \neq c \\
\forall x y. P(f(x), y) \Rightarrow g(x) = y \\
\neg (\forall x y. P(f(x), y) \Rightarrow g(x) = y) \lor \neg P(f(b), c) \lor g(b) = c
\]
E-matching

\[ P(a,c) \]
\[ f(b) = a \]
\[ \forall x. g(x) \neq c \]
\[ \forall xy. P(f(x), y) \Rightarrow g(x) = y \]
\[ \neg (\forall xy. P(f(x), y) \Rightarrow g(x) = y) \lor \neg P(f(b), c) \lor g(b) = c \]

\[ \forall x. g(x) \neq c \Rightarrow g(b) \neq c \]

\[ \Rightarrow \text{New terms lead to new instances} \]
E-matching

\[ P(a,c) \]
\[ f(b) = a \]
\[ \forall x. g(x) \neq c \]
\[ \forall xy. P(f(x), y) \implies g(x) = y \]
\[ \neg(\forall xy. P(f(x), y) \implies g(x) = y) \lor \neg P(f(b), c) \lor g(b) = c \]
\[ \neg(\forall x. g(x) \neq c) \lor g(b) \neq c \]

\[ E' \]
\[ Q' \]

Ground Solver

E-matching

Conflict-Based

Model Based
E-matching

\[ P(a, c) \]
\[ f(b) = a \]
\[ \forall x. g(x) \neq c \]
\[ \forall xy. P(f(x), y) \Rightarrow g(x) = y \]
\[ \neg (\forall xy. P(f(x), y) \Rightarrow g(x) = y) \lor \neg P(f(b), c) \lor g(b) = c \]
\[ \neg (\forall x. g(x) \neq c) \lor g(b) \neq c \]

\[ \Rightarrow \text{Success!} \]
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 “$E \cup Q$ is T-satisfiable”
  - Although for some fragments+variants, we may guarantee (termination $\Leftrightarrow$ model)
    - Decision Procedures via Triggers [Dross et al 13]
    - Local Theory Extensions [Bansal et al 15]
        $\Rightarrow$ 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: Too Many Instances

- Typical problems in applications:
  - $F$ contains 1000s of clauses
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 $Q$
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

$E_3 \approx 10000$ $Q \approx 100$ $F_1 \approx 10000$ $F_2 \approx 100000$ $F_3 \approx 10000$

$\Rightarrow$ 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>cvc4</th>
<th>z3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>%</td>
<td>#</td>
</tr>
<tr>
<td>1-10</td>
<td>1464</td>
<td>13.49%</td>
<td>1007</td>
</tr>
<tr>
<td>10-100</td>
<td>1755</td>
<td>16.17%</td>
<td>1853</td>
</tr>
<tr>
<td>100-1000</td>
<td>3816</td>
<td>35.16%</td>
<td>3680</td>
</tr>
<tr>
<td>1000-10k</td>
<td>1893</td>
<td>17.44%</td>
<td>2468</td>
</tr>
<tr>
<td>10k-100k</td>
<td>1162</td>
<td>10.71%</td>
<td>1414</td>
</tr>
<tr>
<td>100k-1M</td>
<td>560</td>
<td>5.16%</td>
<td>607</td>
</tr>
<tr>
<td>1M-10M</td>
<td>193</td>
<td>1.78%</td>
<td>330</td>
</tr>
<tr>
<td>&gt;10M</td>
<td>10</td>
<td>0.09%</td>
<td>0</td>
</tr>
</tbody>
</table>

- 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)
In fact, E-matching may be exponential, above produces $2^{32}$ instances.

Thus, we limit the number of matches per ground term/pattern pair.
E-matching: Non-termination

Ground Solver

\[ \forall x. f(f(x)) = f(x) \]
\[ f(a) = a \]

⇒ E-matching may be non-terminating
E-matching: Non-termination

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

\[ f(a) = a \]
E-matching: Non-termination

\[ f(a) = a \]
\[ \forall x. f(f(x)) = f(x) \]

- **E-matching**
- **Model Based**
- **Conflict-Based**

\[ \forall x. f(f(x)) = f(x) \]
\[ f(a) = a \]
\[ f(f(a)) = f(a) \]

**Ground Solver**

\[ f(a) = a \]
\[ \forall x. f(f(x)) = f(x) \]
E-matching: Non-termination

E:
- \( f(a) = a \)
- \( f(f(a)) = f(a) \)

Q:
- \( \forall x. f(f(x)) = f(x) \)

Ground Solver

E-matching

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

\[
\begin{align*}
\overline{f(a) = a} \\
\overline{f(f(a)) = f(a)} 
\end{align*}
\]
E-matching: Non-termination

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

- \( f(a) = a \)
- \( f(f(a)) = f(a) \)
- \( f(f(f(a))) = f(f(a)) \)

Ground Solver

\[ \exists x. f(f(x)) = f(x) \]

E-matching

Conflict-Based

Model Based
E-matching: Non-termination

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

LOOPS INDEFINITELY

\[ \Rightarrow \text{Situation is referred to as a matching loop} \]
E-matching: Non-termination

Various ways to avoid matching loops, e.g. by:
- Restricting pattern selection
- Fair instantiations strategies (track “levels”)

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

\[
f(a) = a
\]

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

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

... LOOPS INDEFINITELY

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

\[
f(a) = a
\]

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

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

...
E-matching: Incompleteness

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

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

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

$E$:
- empty

$Q$:
- $\forall x. P(x)$
- $\forall x. \neg P(x)$

$\Rightarrow$ 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 $E$
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 are 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?
    ⇒Use conflict-based instantiation [Reynolds/Tinelli/deMoura FMCAD14]
  • What can we do when there are no instances to add, problem is “sat”?
    ⇒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)

\( \Rightarrow \text{Leads to fewer instances, improved ability of ground solver to answer “unsat”} \)
Conflict-Based Instantiation

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

\[ \forall x. P(x) \lor R(x) \]
Conflict-Based Instantiation

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

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

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

\[ E, Q, P(a) \lor R(a) \Rightarrow P(a) \lor R(a) \]

\[ E, Q, P(b) \lor R(b) \Rightarrow P(b) \lor R(b) \]

\[ E, Q, P(c) \lor R(c) \Rightarrow P(c) \lor R(c) \]

\[ E, Q, P(d) \lor R(d) \Rightarrow P(d) \lor R(d) \]

\[ E, Q, P(e) \lor R(e) \Rightarrow P(e) \lor R(e) \]
Consider what we learn from these instances:

- $E, Q, P(a) ∨ R(a) \models T ∨ R(a)$
- $E, Q, P(b) ∨ R(b) \models P(b) ∨ R(b)$
- $E, Q, P(c) ∨ R(c) \models P(c) ∨ R(c)$
- $E, Q, P(d) ∨ R(d) \models P(d) ∨ R(d)$
- $E, Q, P(e) ∨ R(e) \models P(e) ∨ R(e)$

By $E$, we know $P(a) \iff T$
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:

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

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

We know \( P(b) \leftrightarrow \bot \)
Conflict-Based Instantiation

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

\( \Rightarrow \) Consider what we learn from these instances:

\[ E, Q, P(a) \lor R(a) \Rightarrow T \]
\[ E, Q, P(b) \lor R(b) \Rightarrow R(b) \]
\[ E, Q, P(c) \lor R(c) \Rightarrow P(c) \lor R(c) \]
\[ E, Q, P(d) \lor R(d) \Rightarrow P(d) \lor R(d) \]
\[ E, Q, P(e) \lor R(e) \Rightarrow P(e) \lor R(e) \]
Consider what we learn from these instances:

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

Consider what we learn from these instances:

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

We know \( R(d) \Leftrightarrow T \)
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 \( R(e) \Leftrightarrow \bot \)
Conflict-Based Instantiation

Consider what we learn from these instances:

- $P(a), \neg P(b)$
- $\neg P(c), \neg R(a)$
- $R(d), \neg R(e)$
- $\neg R(c)$

We know $R(c) \iff \bot$

$(\forall x. P(x) \lor R(x)) \Rightarrow P(a) \lor R(a)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(b) \lor R(b)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(c) \lor R(c)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(d) \lor R(d)$
$(\forall x. P(x) \lor R(x)) \Rightarrow P(e) \lor R(e)$
Consider what we learn from these instances:

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

Consider what we learn from these instances:

\((\forall x. P(x) \lor R(x)) \implies P(a) \lor R(a)\)
\((\forall x. P(x) \lor R(x)) \implies P(b) \lor R(b)\)
\((\forall x. P(x) \lor R(x)) \implies P(c) \lor R(c)\)
\((\forall x. P(x) \lor R(x)) \implies P(d) \lor R(d)\)
\((\forall x. P(x) \lor R(x)) \implies P(e) \lor R(e)\)

\(P(c) \lor R(c)\) is a conflicting instance for \((E,Q)\)!
Conflict-Based Instantiation

Consider what we learn from these instances:

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

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

\[ E, Q \rightarrow E', Q' \rightarrow E'', Q'' \rightarrow \ldots \rightarrow \text{unsat} \]
Conflict-Based Instantiation: EUF

\[(a \neq c, f(b) = b, g(b) = a, f(a) = a, h(f(a)) = d, h(b) = c), \quad \forall x. f(g(x)) = h(f(x))\]
Consider the instance \( \forall x. f(g(x)) = h(f(x)) \) for \((E, Q)\)?

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

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

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

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

Consider the equivalence classes of $E$

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

$a = g(b) = f(a)$
$b = f(b)$
$c = h(b)$
$d = h(f(a))$

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

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

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

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)) \}

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

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

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

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

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

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

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

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

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

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

\[ E, Q, f(g(b)) = h(f(b)) \models_{E} a = c \]

Diagram:
- E
- Q
- CBQI
Conflict-Based Instantiation: EUF

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

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

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

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

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

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

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

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

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

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

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

Build partial definitions
Conflict-Based Instantiation: EUF

\[
E \left\{ \ldots, f(b) = b, \quad g(b) = a, f(a) = a, \quad h(f(a)) = d, h(b) = c \right\}
\]

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

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

Check entailment
Conflict-Based Instantiation: EUF

E

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

Q

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

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

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

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

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

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

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

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

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

\[ \Rightarrow \text{These are also useful} \]
Conflict-Based Instantiation

Given:

- Set of ground T-literals $E$
- Quantified formulas $Q$

**Conflict-based instantiation:**

1. If there exists a *conflicting instance* $E, \Psi\{x \rightarrow t\} \models T \bot$
   - Returns $\{\forall x. \Psi \Rightarrow \Psi\{x \rightarrow t\}\}$ only

2. If there exists *propagating instance(s)* $E, \Psi_i\{x \rightarrow t_i\} \models T s_i = u_i$, 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

Given:
• Set of ground T-literals $E$
• Quantified formulas $Q$

• Conflict-based instantiation:
  1. If there exists a conflicting instance $E, \Psi(x \rightarrow t) \models_T \bot$
     • Returns $\{\forall x. \Psi \Rightarrow \Psi(x \rightarrow t)\}$ only
  2. If there exists propagating instance(s), $E, \Psi_i(x \rightarrow t_i) \models_T s_i = u_i$, for $i = 1, ..., n$
     • Returns $\{\forall x. \Psi_1 \Rightarrow \Psi_1(x \rightarrow t_1), ..., \forall x. \Psi_n \Rightarrow \Psi_n(x \rightarrow t_n)\}$ 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

<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>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>5,245</td>
<td>627.0M</td>
<td>3,827</td>
</tr>
<tr>
<td>z3</td>
<td>6,269</td>
<td>613.5M</td>
<td>3,506</td>
</tr>
<tr>
<td>cvc4</td>
<td>6,100</td>
<td>879.0M</td>
<td>3,858</td>
</tr>
<tr>
<td>cvc4+ci</td>
<td><strong>6,616</strong></td>
<td>150.9M</td>
<td><strong>4,082</strong></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 \textit{find} conflicting instances?
- What about conflicts involving \textit{multiple quantified formulas}?
- What if our quantified formulas that contain \textit{theory symbols}?
Conflict-Based Instantiation: Challenges

• How do we \textit{find} conflicting instances?
Conflict-Based Instantiation: Challenges

• How do we *find* conflicting instances?
  • Naively:
    1. Produce all instances $\Psi_1, \ldots, \Psi_n$ via E-matching for $(E,Q)$
    2. For $i=1,\ldots,n$, check if $\Psi_i$ is a conflicting instance for $(E,Q)$
Conflict-Based Instantiation: Challenges

• How do we find conflicting instances?
  • Naively:
    1. Produce all instances $\Psi_1, \ldots, \Psi_n$ via E-matching for $(E,Q)$
    2. For $i=1,\ldots,n$, check if $\Psi_i$ is a conflicting instance for $(E,Q)$

$\Rightarrow$ but $n$ may be very large!
**Conflict-Based Instantiation: Challenges**

- **How do we find conflicting instances?**
  - Naively:
    1. Produce all instances $\Psi_1, \ldots, \Psi_n$ via E-matching for $(E, Q)$
    2. For $i=1, \ldots, n$, check if $\Psi_i$ is a conflicting instance for $(E, Q)$
  - In practice: it can be done more efficiently:
    - Basic idea: construct instances via a *stronger version of matching*
      - Intuition: for $\forall x. P(x) \lor Q(x)$, will *only* match $P(x)$ with $P(t) \iff \bot$
        (For technical details, see [Reynolds et al FMCAD2014])
Conflict-Based Instantiation: Challenges

• What about conflicts involving *multiple quantified formulas*?

\[ E \left\{ \begin{array}{l} P_0(a) \\ \neg P_{100}(a) \end{array} \right\} \quad Q \left\{ \begin{array}{l} \forall x. P_0(x) \Rightarrow P_1(x) \\ \forall x. P_1(x) \Rightarrow P_2(x) \\ \vdots \\ \forall x. P_{99}(x) \Rightarrow P_{100}(x) \end{array} \right\} \]
Conflict-Based Instantiation: Challenges

• What about conflicts involving *multiple quantified formulas*?

\[
\begin{align*}
E & \quad \begin{cases}
    P_0(a) \land \neg P_{100}(a) \\
    \neg P_0(a) \\
    \neg P_{100}(a)
\end{cases} & \quad Q & \begin{cases}
    \forall x. P_0(x) \Rightarrow P_1(x) \\
    \forall x. P_1(x) \Rightarrow P_2(x) \\
    \ldots \\
    \forall x. P_{99}(x) \Rightarrow P_{100}(x)
\end{cases}
\end{align*}
\]

• Want to find:

\[
E, P_0(a) \Rightarrow P_1(a), P_1(a) \Rightarrow P_2(a), \ldots, P_{99}(a) \Rightarrow P_{100}(a) \vdash E \perp
\]

⇒ Current implementations would take 100 rounds to infer this
Conflict-Based Instantiation: Challenges

- What about quantified formulas that contain \textit{theory symbols}? 

\[
\begin{align*}
\text{E} & \quad \text{f}(1) = 5 \\
\text{Q} & \quad \forall xy. f(x+y) > x + 2y
\end{align*}
\]
Conflict-Based Instantiation: Challenges

• What about quantified formulas that contain *theory symbols*?

\[ E \left\{ \begin{array}{l}
 f(1) = 5 \\
 Q \left\{ \begin{array}{l}
 \forall x y . f(x+y) > x + 2y
\end{array} \right. \\
\end{array} \right. \]

• Want to find, e.g.:
  • \( E, f(-3+4) > -3 + 2 \cdot 4 \) \|_{\text{UFLIA}} f(-3+4) > -3 + 2 \cdot 4
Conflict-Based Instantiation: Challenges

- What about quantified formulas that contain *theory symbols*?

\[
\begin{align*}
\exists & \left(f(1) = 5\right) \quad \text{Q} \\
\forall & \left(\forall xy. f(x+y) > x+2*y\right)
\end{align*}
\]

- Want to find, e.g.:
  - \(\exists, f(-3+4) > -3+2*4 \models_{\text{UFLIA}} f(1) > 5\)
Conflict-Based Instantiation: Challenges

• What about quantified formulas that contain *theory symbols*?

\[
\begin{align*}
E & \quad f(1)=5 \\
Q & \quad \forall x y. f(x+y)>x+2*y
\end{align*}
\]

• Want to find, e.g.:
  - \( E, f(-3+4)>-3+2*4 \) \( \models_{\text{UFLIA}} 5>5 \) By \( E \), we know \( f(1)=5 \)
Conflict-Based Instantiation: Challenges

• What about quantified formulas that contain *theory symbols*?

\[ E, f(-3+4) > -3+2*4 \models \text{UFLIA} \]
\[ \forall xy. f(x+y) > x+2*y \]

• Want to find, e.g.:
  • \( E, f(-3+4) > -3+2*4 \models \text{UFLIA} \)
Conflict-Based Instantiation: Challenges

• What about quantified formulas that contain *theory symbols*?

\[ \exists f(1) = 5 \quad \forall x y. f(x+y) > x + 2y \]

• Want to find, e.g.:
  • \( E, f(-3+4) > -3 + 2 \cdot 4 \ \models_{\text{UFLIA}} \bot \)

\[ \Rightarrow \text{In practice, finding such instances cannot be done efficiently} \]
Conflict-Based Instantiation: Summary

• Instantiation technique for \((E, Q)\), where:
  
  \[ \Rightarrow \text{From } Q, \text{ derive conflicts } \perp, \text{ and equalities } g_1 = g_2 \text{ between ground terms } g_1, g_2 \text{ from } E \]

• Run with higher priority to E-matching
  
  • Resort to E-matching only if no conflicting or propagating instances can be found

• Leads to fewer instances, greater ability to answer “unsat”
Model-based Instantiation

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

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

\[ \Rightarrow \text{What if } E \cup Q \text{ is satisfiable?} \]
What if $E \cup Q$ is satisfiable?

- Use model-based quantifier instantiation (MBQI)
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
     • e.g. $P^M \iff \lambda x. \text{ite}(x>0, T, \bot)$
  2. If this interpretation satisfies all formulas in $Q$, answer “sat”
     • e.g. interpretation $M$ satisfies $\forall x. x>4 \Rightarrow P(x)$

$\Rightarrow$ 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

EB

MBQI

Conflict-Based

E-matching

Model-Based
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

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

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

Build interpretation \( M \) of predicates
- This interpretation must satisfy \( E \)
Model-based Instantiation

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

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

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

Build interpretation $M$ of predicates
- This interpretation must satisfy $E$
- Missing values may be filled in arbitrarily
Model-based Instantiation

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

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

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

\( M \)

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

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

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

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

Ground Solver

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

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

MBQI

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

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

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

**Ground Solver**

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

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

**MBQI**

\[ P^M \leftrightarrow \lambda x. \neg P(a), P(b), \neg R(b), \neg R(c) \]

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

\[ \neg (P^M(k) \lor R^M(k)) \]

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

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

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

Ground Solver

Check:

\[ \neg \left( \text{ite}(k=a, \bot, \text{ite}(k=b, T, T)) \lor \text{ite}(k=b, \bot, \text{ite}(k=c, \bot, \bot)) \right) \]

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

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

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

Ground Solver

Check: \( \neg (k \neq a \lor \perp) \)
Model-based Instantiation

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

$$\neg P(a)$$

$$\neg P(b)$$

$$\neg R(b)$$

$$\neg R(c)$$

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

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

Check: \( k = a \)

\( \Rightarrow \) Simplify
Model-based Instantiation

Model-Based

Q

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

E

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

Ground Solver

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

MBQI

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

Check: \( k=a \)

\( \Rightarrow \text{Satisfiable! There are values } k \text{ for which } M \text{ does not satisfy } Q \)
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

Check: \( k=a \)

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

\[ \Rightarrow \text{Add one instance for one such value of } k \text{ for which } M \text{ did satisfy } Q \]
Model-based Instantiation

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

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

\[ \Rightarrow \text{Subsequent models must satisfy } P(x) \lor R(x) \text{ for } x \rightarrow a \]
Model-based Instantiation

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

\( \neg P(a) \)

\( \neg P(b) \)

\( \neg R(b) \)

\( \neg R(c) \)

\( \neg R(a) \)

\( E' \)

\( Q' \)

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

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

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

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

Ground Solver

MBQI

• Repeat as necessary

\( \Rightarrow \) Model refinement loop
Model-based Instantiation

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

Ground Solver

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

\( P(c) \)

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

Ground Solver

MBQI

Conflict-Based

E-matching

Model-Based
Model-based Instantiation

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

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

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

E''

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

Q''

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

Check: \( k=a \land k \neq a \)

MBQI

\( P^M'' \iff \lambda x. \)
\( \text{ite}(x=a, \bot, \text{ite}(x=b, T, \text{ite}(x=c, T, T))) \)

\( R^M'' \iff \lambda x. \)
\( \text{ite}(x=a, T, \text{ite}(x=b, \bot, \text{ite}(x=c, \bot, \bot))) \)

M''
Model-based Instantiation

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

Check: \( k = a \land k \neq a \)

\Rightarrow \text{Unsatisfiable, there are no values } k \text{ for which } M'' \text{ does not satisfy } Q \]
Model-based Instantiation

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

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

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

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

\[ \text{sat} , \text{model } M'' \]
Model-based Instantiation: Completeness

• Seen techniques for which:
  • Ground Solver may answer
  • Quantifiers Module (+ model-based instantiation) may answer

• Under what conditions are these techniques *terminating*?
Model-based Instantiation: Completeness

• Seen techniques for which:
  • Ground Solver may answer **unsat**
  • Quantifiers Module (+ model-based instantiation) may answer **sat**

• Under what conditions are these techniques **terminating**?
  A. If the domains of ∀ are interpreted as finite
     • E.g. quantified bitvectors [Wintersteiger et al 13]
Model-based Instantiation: Completeness

• Seen techniques for which:
  • Ground Solver may answer unsat
  • Quantifiers Module (+ model-based instantiation) may answer sat

• Under what conditions are these techniques terminating?
  A. If the domains of $\forall$ are interpreted as finite
     • E.g. quantified bitvectors [Wintersteiger et al 13]
  B. If the domains of $\forall$ may be interpreted as finite in a model
     • Finite model finding [Reynolds et al 13]
Model-based Instantiation: Completeness

• Seen techniques for which:
  • Ground Solver may answer
  • Quantifiers Module (+ model-based instantiation) may answer

• Under what conditions are these techniques terminating?
  A. If the domains of $\forall$ are interpreted as finite
     • E.g. quantified bitvectors [Wintersteiger et al 13]
  B. If the domains of $\forall$ may be interpreted as finite in a model
     • Finite model finding [Reynolds et al 13]
  C. If the domains of $\forall$ are infinite
     ...but it can be argued that only finitely many instances will be generated
     • E.g. essentially uninterpreted fragment [Ge+deMoura 09], ...
Model-based Instantiation: Impact

• 1203 satisfiable benchmarks from the TPTP library
  • Graph shows # instances required by exhaustive instantiation
    • E.g. $\forall xyz: U. P(x, y, z)$, 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 [Reynolds et al CADE13]
  - Scales to >2 billion instances with a 30 sec timeout, only adds fraction of possible instances
Model-based Instantiation: Challenges

- Conflict-Based
- E-matching
- Model-Based
Model-based Instantiation: Challenges

• How do we build interpretations $M$?
  • Typically, build interpretations $f^M$ that are almost constant:
    • e.g. $f^M := \lambda x. \text{ite}(x=t_1, v_1, \text{ite}(x=t_2, v_2, \ldots, \text{ite}(x=t_n, v_n, v_{\text{def}}) \ldots))$
Model-based Instantiation: Challenges

• How do we build interpretations $M$?

  • Typically, build interpretations $f^M$ that are almost constant:
    • e.g. $f^M := \lambda x. \text{ite}(x=t_1, v_1, \text{ite}(x=t_2, v_2, \ldots, \text{ite}(x=t_n, v_n, v_{\text{def}}) \ldots))$

...but models may need to be more complex when theories are present:

\[
\forall xy: \text{Int.} \ (f(x, y) \geq x \land f(x, y) \geq y) \quad \quad \Rightarrow \quad \quad f^M := \lambda xy. \text{ite}(x \geq y, x, y)
\]

\[
\forall x: \text{Int.} \ 3g(x) + 5h(x) = x \quad \quad \Rightarrow \quad \quad g^M := \lambda x. 5x \quad h^M := \lambda x. -3x
\]

\[
\forall xy: \text{Int.} \ u(x+y) + 11v(w(x)) = x+y \quad \quad \Rightarrow \quad \quad ???
\]
Putting it Together

Quantifiers Module

- Conflict-Based
- E-matching
- Model Based
Putting it Together

- **Input:**
  - Ground literals $E$
  - Quantified formulas $Q$

![Diagram of Quantifiers Module]

- Conflict-Based
- E-matching
- Model Based
Putting it Together

Quantifiers Module

E-matching

Model Based

Conflict-Based

\( E \land Q \) is unsat

\( E, \neg P(a) \models \bot \)

where \( \forall x. P(x) \in Q \)
Putting it Together

Quantifiers Module

E-matching

Model Based

Conflict-Based

E \land Q \text{ is unsat}

where \( \forall x. P(x) \in Q \)

pattern matching

E, \neg P(a) \models \bot

P(a), P(b), P(c), P(d), P(e), P(f), ...

where \( P(b), P(c), P(d), P(e), P(f), ... \)
Putting it Together

Quantifiers Module

E-matching

Model Based

Conflict-Based

\( E \land Q \) is unsat

model for \( E \)

pattern matching

where \( \forall x . P(x) \in Q \)

\( P(a), P(b), P(c), P(d), P(e), P(f), \ldots \)

where \( E, \neg P(a) \models \perp \)
Putting it Together

Quantifiers Module

E-matching

Model Based

Conflict-Based

\[ E \land Q \text{ is unsat} \]

\[ E \lor Q \text{ is sat, model } M \]

where \( \forall x. P(x) \in Q \)

M is not a model for Q

\[ P(a), P(b), P(c), P(d), P(e), P(f), \ldots \]

\[ P(z), \text{ where } M \vDash P(z) \]

\[ P(a), \text{ where } E, \neg P(a) \vDash \bot \]

pattern matching

where \( \neg \exists x. P(x) \vDash \bot \)

Putting it Together
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

$\Rightarrow$ 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, ...
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 $\forall + pure$ 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 implemented in an SMT context?
Techniques for Quantifier Instantiation

Ground Solver

- \( F, \ldots \)
- Unsatisfied (unsat)
- Satisfying assignment \( E, Q \)

Quantifiers Module

- Conflict-Based
- E-matching
- Model Based

Generally, used for quantifiers with UF

Generally, used for quantifiers w/o UF

\( E \cup Q \) is T-satisfiable

Instances of \( \forall \) in \( Q \)
Techniques for Quantifier Instantiation

Ground Solver

- Instances of $\forall$ in $Q$
- Satisfying assignment $E, Q$
- $F, \ldots$

Quantifiers Module

- Conflict-Based
- E-matching
- Model Based

CE-Guided

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

$\Rightarrow$ 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: \( \forall x. \neg \psi[x] \leftrightarrow \neg \psi[t_1] \land \ldots \land \neg \psi[t_n] \) 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: $\forall x. \neg \psi[x] \iff \neg \psi[t_1] \land \cdots \land \neg \psi[t_n]$ 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
Counterexample-Guided Instantiation

⇒ Consider ∀ in the theory of linear integer arithmetic LIA:
  ∃abc . (a=b+5 ∧ ∀x . (x>a ∨ x<b ∨ x−c<3) )
Consider $\forall$ in the theory of linear integer arithmetic LIA:

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

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

ground solver

E

Q

\( a=b+5 \)

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

F
Counterexample-Guided Instantiation

$E \quad a=b+5$

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

$\Rightarrow$ Use counterexample-guided instantiation
Counterexample-Guided Instantiation

With respect to model-based instantiation:

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

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

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

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

- **Key difference:** use the same (ground) solver for \( F \) and counterexample \( k \) for \( Q \)

⇒ **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

Ground Solver

CE-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

\[ a = b + 5 \]

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

\[ G \Rightarrow (k \leq a \land k \geq b \land k \geq c + 3) \]

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

\( C \) 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:

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

CE-Guided Ground Solver

instances

\( F \)
Counterexample-Guided Instantiation

1. $F$ is unsatisfiable

$\Rightarrow$ answer “unsat”
Counterexample-Guided Instantiation

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

- Three cases:
  1. \( F \) is unsatisfiable \( \rightarrow \) answer \( \text{unsat} \)
  2. \( F \) is satisfiable, \( \neg C \in E \) for all assignments \( E \) \( \rightarrow \) answer \( \text{sat} \)
  3. \( F \) is satisfiable, \( \neg C \not\in E \) \( \rightarrow \) answer \( \text{sat} \)

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

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

Three cases:

1. \( F_1 \) is unsatisfiable \( \Rightarrow \) unsat

2. \( F_1 \) is satisfiable, \( \neg C \in E \text{ for all assignments } E \) \( \Rightarrow \) answer “sat”
Counterexample-Guided Instantiation

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

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

where \( k \notin FV(t) \)

CE-Guided

Ground Solver

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

CE-Guided Instantiation

return

\[ \ldots \Rightarrow t > a \lor t < b \lor t - c < 3 \]

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

• **Three cases:**
  1. $F$ is unsatisfiable
     ⇒ answer “unsat”
  2. $F$ is satisfiable, $\neg C \in E$ for all assignments $E$
     ⇒ answer “sat”
  3. $F$ is satisfiable, $C \in E$ for some assignment $E$
     ⇒ 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 \( E \)

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

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

CE-Guided

Ground Solver

Unsat

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

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

\[ \Rightarrow \text{answer “unsat”} \]

Sat

\[ \ldots \Rightarrow t > a \lor t < b \lor t-c < 3 \]

\[ \Rightarrow \text{answer “sat”} \]

\[ \Rightarrow \text{add an instance to } F \]

(...which \( t \)?)
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

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

**CE-Guided**
Counterexample-Guided Instantiation

E

C, a=b+5, 
k≤a 
k≥b 
k≥c+3

Q

∀x. (x>a ∨ x<b ∨ x-c<3)

Ground Solver

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

CE-GQI

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

Build model M for E
Counterexample-Guided Instantiation

\[ a = b + 5 \]

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

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

C, \ a = b + 5, \ k \leq a \]

\[ k \geq b \]

\[ k \geq c + 3 \]

Ground Solver

CE-GQI

Take lower bounds of k in E

E

CE-Guided

Q

\[ a^M = 5 \]

\[ b^M = 0 \]

\[ c^M = 0 \]

\[ k^M = 3 \]
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, \( a = b + 5 \),
\[ k \leq a \]
\[ k \geq b \]
\[ k \geq c + 3 \]

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

Compute their value in \( M \)

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

in \( M \)
\[ k \geq b = 0 \]
\[ k \geq c + 3 = 3 \]
Counterexample-Guided Instantiation

Ground Solver

CEQG

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

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

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

\[ a = b + 5 \]

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

\[ k \leq a \]

\[ k \geq b \]

\[ k \geq c + 3 \]

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

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

\[ k \leq a \]

\[ k \geq b \]

\[ k \geq c + 3 \]

\[ a^M = 5 \]

\[ b^M = 0 \]

\[ c^M = 0 \]

\[ k^M = 3 \]

\[ \text{in } M \]

\[ k \geq b = 0 \]

\[ k \geq c + 3 = 3 \]
Counterexample-Guided Instantiation

Counterexample-Guided Instantiation (CE-GI)

Ground Solver

CEGQI

a = b + 5
\neg \forall x. (x > a \vee x < b \vee x - c < 3) \vee c + 3 > a \vee c + 3 < b

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

Ground Solver

E

\(c, a=b+5, c+3<b,\)

\(k \leq a\)

\(k \geq b\)

\(k \geq c+3\)

CEGQI

Q

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

\(a = b + 5\)

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

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

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

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

**Ground Solver**

\[ a = b + 5 \]

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

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

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

**CEGQI**

\[ a^M = 5 \]

\[ b^M = 0 \]

\[ c^M = -4 \]

\[ k^M = 3 \]

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

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

\[ k \leq a \]

\[ k \geq b \]

\[ k \geq c + 3 \]

**E**

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

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

\[ k \leq a \]

\[ k \geq b \]

\[ k \geq c + 3 \]

**Q**
Counterexample-Guided Instantiation

Ground Solver

CE-Guided

Take lower bounds of $k$ in $E$

$E$

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

$Q$

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

$CEGQI$

$a^M=5$
$b^M=0$
$c^M=-4$
$k^M=3$

$k\geq b, k\geq c+3$

$a=b+5$

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

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

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

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

Ground Solver

CE-Guided

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

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

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

C, a = b + 5, c + 3 < b,

\[ k \leq a \]

\[ k \geq b \]

\[ k \geq c + 3 \]

\[ a^M = 5 \]

\[ b^M = 0 \]

\[ c^M = -4 \]

\[ k^M = 3 \]

in M

\[ k \geq b = 0 \]

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

Compute their value in M
Counterexample-Guided Instantiation

Ground Solver

CE-GUI

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

Add instance for lower bound that is maximal in $M$

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

$a = b + 5$

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

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

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

$C, a = b + 5, c + 3 < b,$

$k \leq a$

$k \geq b$

$k \geq c + 3$

$a^M = 5$

$b^M = 0$

$c^M = -4$

$k^M = 3$

in $M$

$k \geq b = 0$

$k \geq c + 3 = -1$
Counterexample-Guided Instantiation

- **Ground Solver**
  - $\forall x. (x > a \lor x < b \lor x - c < 3)$
  - $k \leq a$
  - $k \geq b$
  - $k \geq c + 3$

- **CEGQI**
  - $a = b + 5$
  - $b - c < 3$
  - $k \leq a \land k \geq b \land k \geq c + 3$

- **In $M$**
  - $k \geq b = 0$
  - $k \geq c + 3 = -1$

- **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

Ground Solver

CE-GQI

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

\[ a = b + 5 \]

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

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

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

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

**Ground Solver**

**CEGQI**

- \( b \)
- \( a \)
- \( c + 3 \)
- \( k \leq a \)
- \( k \geq b \)
- \( k \geq c + 3 \)
Counterexample-Guided Instantiation

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

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

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

CEGQI

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

\[
\\begin{align*}
\forall x \ (x > a & \lor x < b \lor x - c < 3) \\
\neg C & \\
a = b + 5 & \\
c + 3 < a & \\
b < c + 3 & \\
\Rightarrow \exists abc \ (a = b + 5 \land \forall x \ (x > a \lor x < b \lor x - c < 3)) \\
\text{is LIA-satisfiable}
\end{align*}
\]
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, ...

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

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

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

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

$ F[k] \rightarrow \{ x \mapsto k^M \}$

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

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

- Can be used for:
  - Quantifier elimination
    \( \psi[t_1] \land ... \land \psi[t_n] \text{ is (un)sat} \)
    - \( \exists x. \neg \psi[x] \) is equivalent to \( \neg \psi[t_1] \lor ... \lor \neg \psi[t_n] \)
  - Function Synthesis
    \( \psi[t_1] \land ... \land \psi[t_n] \text{ is unsat} \)
    - \( \lambda x. \text{ite} (\psi[t_1], t_1, \ldots, \text{ite} (\psi[t_{n-1}], t_{n-1}, t_n) \ldots) \) is a solution for \( f \) in \( \forall x. \psi[f(x)] \)
Counterexample-Guided Instantiation

• Challenge:
Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!
Counterexample-Guided Instantiation

• Challenge: does not work in presence of uninterpreted functions!

\[
\forall x. x < a \lor x < b \lor P(x) \\
\neg C \lor (k \geq a \land k \geq b \land \neg P(k))
\]
Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!

\[ \forall x (x < a \lor x < b \lor P(x)) \]

\[ \neg C \lor (k \geq a \land k \geq b \land \neg P(k)) \]
Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!
Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!
Counterexample-Guided Instantiation

• Challenge: does not work in presence of uninterpreted functions!
Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!

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

\[ \forall x. x < a \lor x < b \lor P(x) \]

\[ \neg C \lor (k \geq a \land k \geq b \land \neg P(k)) \]

\[ a^M = 1 \]
\[ b^M = 0 \]
\[ k^M = 2 \]

in \( M \)

\[ k \geq a = 1 \]
\[ k \geq b = 0 \]

\( \Rightarrow a \) is still the maximal lower bound in \( M \)!
Counterexample-Guided Instantiation

• Challenge: does not work in presence of uninterpreted functions!

\[ \forall x. (x < a \lor x < b \lor P(x)) \Rightarrow a < b \lor P(a) \]
\[ \forall x. x < a \lor x < b \lor P(x) \]
\[ \neg C \lor (k \geq a \land k \geq b \land \neg P(k)) \]

\[ \Rightarrow \text{Unlike the pure arithmetic case:} \]
• Instance does not suffice to rule out \( a \) as maximal lower bound
Summary

SMT solvers handle quantifiers+theories via combination of:
  - DPLL(T)-based ground solver
  - Instantiation via:
    - Conflict-based, E-matching, Model-Based Instantiation
      - Effective in practice for \(\forall+\text{UF}, \forall+\text{UFLIA}, \forall+\text{UFLRA}, \ldots\)
      - Can be decision procedure for limited fragments, e.g. Bernays-Shonfinkel
      - Conflict-Based, E-matching are useful for “unsat”
      - Model-Based is useful for “sat”
    - Counterexample-guided Instantiation
      - Decision procedure for \(\forall+\text{LRA}, \forall+\text{LIA}, \forall+\text{BV}, \ldots\)
In practice: Distribute $\forall$ to proper strategy

Quantifiers Module

$\forall x. \psi[x]$

$\psi$ contains UF

- Conflict-Based
- E-matching
- Model Based

$\psi$ contains no UF

- CE-Guided
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
- **ground literals** $E$
- **$\forall$ formulas** $Q$
- **unsat** → **sat**
Summary: DPLL(T)+Instantiation

Ground Solver
unsat
T-clauses \( F \)
Quantifiers Module

Lemmas

SMT Solver

Conflict-Based
CE-Guided

Model-Based

T-Decision Procedures

SAT Solver

ground literals \( \exists \)
\( \forall \) formulas \( Q \)

sat
Other Important Aspects of $\forall$ Not Covered

- Eager Quantifier Instantiation
- Relevancy
- Preprocessing
- Rewriting
Future Challenges

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

• Develop new approaches for $\forall + UF + \text{theories}$ that:
  • Are efficient in practice
    • E-matching is efficient for $\forall + UF$, 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

• ....Questions?