A Taste of CVC4
Part 2: Quantified Formulas

Cesare Tinelli
Andrew Reynolds
Clark Barrett
Quantified Formulas in SMT

\( \forall x. P(x) \)

\( P \) is true for all \( x \), where \( P \) is a formula involving some background theory

- Satisfiability problem is undecidable in general
- \( \forall \) are critical for applications:
  - Automated Theorem Proving
  - Software/Hardware verification
  - Synthesis, planning, ...
- \( \forall \) are handled in SMT solvers by a variety of techniques:
  - Complete techniques for certain fragments
  - Heuristic techniques for the general case
Overview

- How do we extend SMT solvers for quantified formulas?

- **Quantifier Instantiation** in CVC4:
  - Heuristic (E-matching)
  - Model-based
  - Conflict-based

- **More advanced techniques** in CVC4:
  - Finite Model Finding
  - Function synthesis
DPLL(T)-based SMT Solver

- DPLL(T)-based SMT solver
  - SAT solver maintains a set of propositional clauses
  - Decision Procedure for T determines satisfiability of conjunctions of T-literals
DPLL(T)-based SMT Solver

- Ground solver = SAT solver + Decision Procedure for T

- DPLL(T)

- Ground solver

- SAT Solver

- Decision Procedure for T

- unsat

- sat
DPLL(T) + Quantifiers

SMT solver consists of:

- **Ground solver** maintains a set of ground (quantifier-free) constraints $G$
- **Quantifiers Module** maintains a set of quantified formulas $Q$

SAT Solver

Decision Procedure for T

Quantifiers Module
DPLL(T) + Quantifier Instantiation

- Primary technique for quantifiers in this talk: **Quantifier Instantiation**
DPLL(T) + Quantifier Instantiation

- If $G$ is T-satisfiable, invoke quantifiers module.
DPLL(T) + Quantifier Instantiation

- Add instances of axioms to $G$

Ground solver

SAT Solver

Decision Procedure for T

Quantifiers Module

$\forall x. P(x)$


- Add instances of axioms to $G$
DPLL(T) + Quantifier Instantiation

- ...and repeat, generally a sound but incomplete procedure
  - Difficult to answer sat (when have we added enough instances of $\forall x . P(x)$?)
Quantifiers Module: Overview

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

• Outputs:
  • $E \land Q$ is T-sat
  • Set of instances of $Q$ to add to $G$
  • ...“unknown” (give up)

• Inputs:
  • Set of ground formulas $G$

• Outputs:
  • “$G$ is T-unsat”, or
  • “$G$ is T-sat”, set of literals $E \models_p G$
Quantifier Instantiation: Design Decisions

• When do we invoke it?
  • Eagerly, during the DPLL(T) search [deMoura/Bjorner CAV07], or
  • Lazily, only after ground solver answers “sat”
Quantifier Instantiation: Design Decisions

• When do we invoke it?
  • Eagerly, during the DPLL(T) search [deMoura/Bjorner CAV07], or
  • Lazily, only after ground solver answers “sat”

• Which instances do we add?
  • …
Quantifier Instantiation: Design Decisions

• When do we invoke it?
  • Eagerly, during the DPLL(T) search \cite{deMoura/Bjorner CAV07}, or
  • Lazily, only after ground solver answers “sat”

• Which instances do we add?
  • …

• Can we terminate?
  i.e. can we ever answer “sat”?
Quantifier Instantiation : in CVC4

• When do we invoke it?
  • Eagerly, during the DPLL(T) search [deMoura/Bjorner CAV09], or
  • Lazily, only after ground solver answers “sat”

• Which instances do we add?
  • …

• Can we terminate?
  i.e. can we ever answer “sat”?
Quantifier Instantiation: in CVC4

• When do we invoke it?
  • Eagerly, during the DPLL(T) search [demMoura/Bjorner CAV09], or
  • Lazily, only after ground solver answers “sat”

• Which instances do we add?
  • E-matching [Detlefs et al 03]
  • Model-based quantifier instantiation [Ge/de Moura CAV09]
  • Conflict-based quantifier instantiation [Reynolds et al FMCAD14]

• Can we terminate?
  i.e. can we ever answer “sat”?
Quantifier Instantiation : in CVC4

• When do we invoke it?
  • Eagerly, during the DPLL(T) search [deMoura/Bjorner CAV09], or
  • Lazily, only after ground solver answers “sat”

• Which instances do we add?
  • E-matching [Detlefs et al 03]
  • Model-based quantifier instantiation [Ge/de Moura CAV09]
  • Conflict-based quantifier instantiation [Reynolds et al FMCAD14]

• Can we terminate?
  i.e. can we ever answer “sat”?  
  • Finite Model Finding [Reynolds et al CADE13]
  • Instantiation for linear arithmetic
Quantifiers Module of CVC4

- CVC4’s quantifiers module contains numerous strategies and techniques

- Ground Solver
  - $\forall$ sat
  - instances, lemmas

- Quantifiers Module
  - Conflict-Based
  - E-matching
  - CE-Guided
  - Conjecture Gen
  - Rewrite Rules
  - FunctionDefs
  - Model Based
  - Enumerative
Quantifiers Module of CVC4

- Core techniques: Conflict-based, Heuristic (e.g. E-matching), Model-based

- Ground Solver:
  - $G, \ldots$
  - $\text{G, …}$
  - $\text{Ground Solver}$
  - $\text{Ground Solver}$

- Conflict-Based
  - $\text{Conflict-Based}$

- E-matching
  - $\text{E-matching}$

- CE-Guided
  - $\text{CE-Guided}$

- Conjecture Gen
  - $\text{Conjecture Gen}$

- Rewrite Rules
  - $\text{Rewrite Rules}$

- Function Defs
  - $\text{Function Defs}$

- Model Based
  - $\text{Model Based}$

- Enumerative
  - $\text{Enumerative}$

- $\forall \text{ sat}$

- $\text{sat}$

- $\text{unsat}$

- $\text{sat}$

- $\text{sat}$
E-matching

• E-matching:
  • Most widely used and successful technique for quantifiers in SMT
  • Implemented in numerous solvers:
    • Z3, CVC3, CVC4, VeriT, Alt-Ergo, ...
E-matching: Example

\[ a, b, c : S \]
\[ f, g : S \rightarrow S \]
\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) \neq a \]
\[ \forall x. \ f(x) = g(x) \]
E-matching: Example

\( a, b, c : S \)
\( f, g : S \rightarrow S \)
\( f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) \neq a \)

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

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

\[ a, b, c : S \]
\[ f, g : S \to S \]
\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) \neq a \]
\[ \{x \mapsto a\} \quad \{x \mapsto b\} \quad \{x \mapsto c\} \]
\[ \forall x. f(x) = g(x) \]
E-matching: Example

\[ a, b, c : S \]
\[ f, g : S \rightarrow S \]
\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) \neq a \]
\[ f(a) = g(a), \ f(b) = g(b), \ f(c) = g(c) \]
\[ \forall x. f(x) = g(x) \]
E-matching: Example

\(a, b, c : S\)

\(f, g : S \rightarrow S\)

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

\(f(a) = g(a), \ f(b) = g(b), \ f(c) = g(c)\)

\(\forall x. \ f(x) = g(x)\)
E-matching: Challenges

• What happens when there too many instances to add?
  • E-matching adds many instances, degrades performance for solver to continue

• What happens when there are no instances to add?
  • E-matching is an incomplete procedure, cannot answer SAT even when saturated
E-matching: Challenges

• What happens when there are too many instances to add?
  • E-matching adds many instances, degrades performance for solver to continue
    ⇒Use conflict-based instantiation [Reynolds/Tinelli/deMoura FMCAD14]

• What happens when there are no instances to add?
  • E-matching is an incomplete procedure, cannot answer SAT even when saturated
    ⇒Use model-based instantiation [Ge/deMoura CAV09]
Model-based Instantiation: Example

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

\[ \forall x. f(x) = g(x) \]
Model-based Instantiation: Example

\[ f(a) = a, \quad f(b) = b, \quad f(c) = c, \quad g(a) = a \]
\[ f(a) = g(a), \quad f(b) = g(b), \quad f(c) = g(c) \]
\[ \forall x. f(x) = g(x) \]

- Add instances by E-matching, as before
Model-based Instantiation: Example

\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) = a \]
\[ f(a) = g(a), \ f(b) = g(b), \ f(c) = g(c) \]
\[ \forall x. f(x) = g(x) \]

• E-matching saturates, but ground constraints are satisfiable
  • Can we check that \( \forall x. f(x) = g(x) \) is also satisfiable?
Model-based Instantiation: Example

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

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

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

• **Idea**: construct candidate model \( M \) for functions \( f \) and \( g \)
  
  • Check if \( \forall x. f(x) = g(x) \) satisfied by \( M \)
Model-based Instantiation: Example

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

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

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

\[ f^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]
Model-based Instantiation: Example

\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) = a \]
\[ f(a) = g(a), \ f(b) = g(b), \ f(c) = g(c) \]
\[ \forall x. f(x) = g(x) \]
\[ f^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]
\[ g^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]
Model-based Instantiation: Example

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

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

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

\[ f^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]

\[ g^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]

• Does \( M \) satisfy \( \forall x. f(x) = g(x) \)?
Model-based Instantiation: Example

\[ f(a) = a, \quad f(b) = b, \quad f(c) = c, \quad g(a) = a \]
\[ f(a) = g(a), \quad f(b) = g(b), \quad f(c) = g(c) \]
\[ \forall x. f(x) = g(x) \]

\[ f^M := \lambda x. \text{ITE}(x = a, a, \text{ITE}(x = b, b, c)) \]
\[ g^M := \lambda x. \text{ITE}(x = a, a, \text{ITE}(x = b, b, c)) \]

\[ f^M \mid g^M \]

• Does \( M \) satisfy \( \forall x. f(x) = g(x) \)?

\[ \Rightarrow \text{If } \exists x. f^M(x) \neq g^M(x) \text{ is unsat, then yes} \]
Model-based Instantiation: Example

\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) = a \]
\[ f(a) = g(a), \ f(b) = g(b), \ f(c) = g(c) \]
\[ \forall x. f(x) = g(x) \]
\[ f^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]
\[ g^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]

• Does \( M \) satisfy \( \forall x. f(x) = g(x) \) ?

\[ \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \neq \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]
Model-based Instantiation: Example

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

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

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

\[ f^M := \lambda x. \text{ite}(x = a, a, \text{ite}(x = b, b, c)) \]

\[ g^M := \lambda x. \text{ite}(x = a, a, \text{ite}(x = b, b, c)) \]

• Does \( M \) satisfy \( \forall x. f(x) = g(x) \)?

\[ \Rightarrow \text{Yes, return } \text{sat} \text{ with model } M \]
Model-based Instantiation: Example

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

\[ f(a) = g(a), \quad f(b) = g(b), \quad f(c) = g(c), \quad d \in \{a, b, c\} \]

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

\[ f^M := \lambda x. \text{ite}(x=a, a, \text{ite}(x=b, b, c)) \]

\[ g^M := \lambda x. \text{ite}(x=a, a, \text{ite}(x=c, c, b)) \]

• If \( M \) does not satisfy \( \forall x. f(x) = g(x) \),
Model-based Instantiation: Example

\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) = a \]
\[ f(a) = g(a), \ f(b) = g(b), \ f(c) = g(c), \ d \notin \{a, b, c\} \]
\[ \forall x. f(x) = g(x) \]

\[ f^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]
\[ g^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=c,c,b)) \]

• If \( M \) does not satisfy \( \forall x. f(x) = g(x) \),
  \[ \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \neq \text{ite}(x=a,a,\text{ite}(x=c,c,b)) \]
  is unsat?
Model-based Instantiation: Example

\[ f(a) = a, \ f(b) = b, \ f(c) = c, \ g(a) = a \]
\[ f(a) = g(a), \ f(b) = g(b), \ f(c) = g(c), \ d \notin \{a, b, c\} \]
\[ \forall x. f(x) = g(x) \]

\[ f^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=b,b,c)) \]
\[ g^M := \lambda x. \text{ite}(x=a,a,\text{ite}(x=c,c,b)) \]

- **If** \( M \) **does not satisfy** \( \forall x. f(x) = g(x) \),
  \[ \text{ite}(d=a,a,\text{ite}(d=b,b,c)) \neq \text{ite}(d=a,a,\text{ite}(d=c,c,b)) \]
  Take \( x=d \)
Model-based Instantiation: Example

\[ f(a) = a, \quad f(b) = b, \quad f(c) = c, \quad g(a) = a \]
\[ f(a) = g(a), \quad f(b) = g(b), \quad f(c) = g(c), \quad d \notin \{a, b, c\} \]
\[ \forall x. f(x) = g(x) \]

\[ f^M := \lambda x. \text{ite}(x=a, a, \text{ite}(x=b, b, c)) \]
\[ g^M := \lambda x. \text{ite}(x=a, a, \text{ite}(x=c, c, b)) \]

- If \( M \) does not satisfy \( \forall x. f(x) = g(x) \),
  \[ c \neq b \]

sat, where \( x=d \)
Model-based Instantiation: Example

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

\[ f(a) = g(a), \quad f(b) = g(b), \quad f(c) = g(c), \quad d \notin \{a, b, c\}, \quad \forall x. f(x) = g(x) \]

\[ f^M := \lambda x. \text{ite}(x=a, a, \text{ite}(x=b, b, c)) \]

\[ g^M := \lambda x. \text{ite}(x=a, a, \text{ite}(x=c, c, b)) \]

• If \( M \) does not satisfy \( \forall x. f(x) = g(x) \),
  \( \Rightarrow \) Add instance \( f(d) = g(d) \), will refine model
Conflict-Based Instantiation: Example

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

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

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

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

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

- E-matching may return with many ground instances
  - In practice, 1000+ instances per invocation
    \[ \Rightarrow \text{Degrades solver performance} \]
Conflict-Based Instantiation: Example

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

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

- **Idea:** find an instance of \( \forall x. f(x) = g(x) \) that is conflicting with ground constraints
  - If so, add *only* that instance
Conflict-Based Instantiation: Example

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

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

**Idea:** find an instance of \( \forall x. f(x) = g(x) \) that is conflicting with ground constraints

\( \Rightarrow f(a) = a, \ g(a) \neq a \models f(x) \neq g(x) \{x \mapsto a\} \)
Conflict-Based Instantiation: Example

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

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

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

- **Idea**: find an instance of \( \forall x. f(x) = g(x) \) that is conflicting with ground constraints

  \[ \Rightarrow f(a) = a, \quad g(a) \neq a \quad \models f(x) \neq g(x) \{ x \rightarrow a \} \]
Putting it Together

Quantifiers Module

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

• Input:
  • Ground literals $E$
  • Quantified formulas $Q$
Putting it Together

Quantifiers Module

E-matching

Model Based

Conflict-Based

\(E \land Q\) is unsat

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

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

Quantifiers Module

E-matching

Model Based

Conflict-Based

E = Q is unsat

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

where E |= ¬P(a)

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

Quantifiers Module

E-matching

Model Based

Conflict-Based

\( E \land Q \) is unsat

\( E \vdash \neg P(a) \)

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

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

model for \( E \)

Model Based
Putting it Together

Quantifiers Module

E-matching

Model Based

Conflict-Based

$E \land Q$ is unsat

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

$P(a)$, where $E \models \neg P(a)$

$M$ is not a model for $Q$

$M$ is not a model for $Q$

$P(a)$, where $M \not\models P(a)$

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

$E \land Q$ is sat, model $M$

$E \models P(a)$

where $\exists x. P(x) \in Q$
Other techniques for Quantified Formulas

• Advanced techniques in CVC4:
  • Rewrite Rules
  • Automated Induction [Reynolds/Kuncak VMCAI15]
  • Finite Model Finding [Reynolds et al CADE13]
  • Synthesis [Reynolds et al CAV15]

⇒ Each target a particular type of quantified formulas
Other techniques for Quantified Formulas

• Advanced techniques in CVC4:
  • Rewrite Rules
  • Automated Induction [Reynolds/Kuncak VMCAI15]
  • Finite Model Finding [Reynolds et al CADE13]
  • Synthesis [Reynolds et al CAV15]

⇒ Each target a particular type of quantified formulas

Focus of the remainder
Property P is verified

Manual Inspection

Candidate Model

Verification Condition for P

SMT solver

UNSAT

Unknown

Finite Model Finding: Motivation

Property P is verified with quantifiers
Property P is verified with quantifiers.
Finite Model Finding in SMT

\[ G \]

**Ground Solver**

\[ \forall xy : S.Q(x, y) \]

**Quantifiers Module**
Finite Model Finding in SMT

\[ G \]

\[ \forall x \, y : S \cdot Q(x, y) \]

- If \( S \) has finite interpretation,
  - use finite model finding

\[ \Rightarrow \text{If } S \text{ has finite interpretation,} \]
\[ \cdot \text{use finite model finding} \]
Finite Model Finding in SMT

$G \quad \forall xy : S. Q(x, y)$

$S = \{a, b, c, d, e\}$
Finite Model Finding in SMT

\[ G \equiv Q(a, a) \land \ldots \land Q(e, a) \land Q(a, b) \land \ldots \land Q(a, e) \land Q(a, e) \land \ldots \land Q(e, e) \]

\[ \forall xy : S. Q(x, y) \]

- Reduction of quantified formulas to ground formulas

\[ S = \{a, b, c, d, e\} \]
Finite Model Finding in SMT

\[ G = \forall x y : S. Q(x, y) \]

S = \{a, b, c, d, e\}

\[ \Rightarrow \text{Ability to answer SAT, assuming decision procedure for } G \land Q(a, a) \land \ldots \land Q(e, e) \]

\[ \text{sat} \]
Finite Model Finding in SMT

Ground Solver

Quantifiers Module

**\(G = \forall xy : S . Q(x, y)\)**

\[ Q(a, a) \land \ldots \land Q(e, a) \land Q(a, b) \land \ldots \land Q(a, c) \land \ldots \land Q(a, e) \land \ldots \land Q(e, e) \]

- Can be very large

\[ S = \{a, b, c, d, e\} \]

unsat \hspace{1cm} sat
Finite Model Finding: Example

\( a, b, c, d, e : S \)

\( P, R : (S, S) \rightarrow \text{Bool} \)

\( a \neq b, b \neq c, c \neq d, d \neq e, e \neq a \)

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

\( \forall xy. P(x, y) \lor R(x, y) \)
Finite Model Finding in SMT

• Address large # instantiations by:
  1. Minimizing model sizes [Reynolds et al CAV13]
     • Find interpretation that minimizes the #elements in S
  2. Only add instantiations that refine model [Reynolds et al CADE13]
     • Model-based quantifier instantiation [Ge/deMoura CAV 2009]
Finite Model Finding : Minimizing Model Sizes

• Minimize model sizes using a theory solver for cardinality constraints

```
|S| \leq 1

Search for models where |S|=1

\neg |S| \leq 1

|S| \leq 2

If none exist, search for models where |S|=2

\neg |S| \leq 2

|S| \leq 3

etc.

\neg |S| \leq 3

[Reynolds/Tinelli/Goel/Krstic CAV13]
```
Finite Model Finding: Minimizing Model Sizes

- Minimize model sizes using a theory solver for cardinality constraints

Search for models where $|S| = 1$

If none exist, search for models where $|S| = 2$

etc.

⇒ If model exists where $|S| \leq 3$, only need $3 \times 3 = 9$ instances instead of $5 \times 5 = 25$ instances

[Reynolds/Tinelli/Goel/Krstic CAV13]
FMF: Example

\begin{align*}
a, b, c, d, e & : S \\
P, R & : (S, S) \rightarrow \text{Bool} \\
a & \neq b, \ b & \neq c, \ c & \neq d, \ d & \neq e, \ e & \neq a \\
\neg P(a, b), \ \neg R(a, c) \\
\forall xy. P(x, y) & \lor R(x, y) \\
\end{align*}
FMF: Example

\[ a, b, c, d, e : S \]
\[ P, R : (S, S) \rightarrow \text{Bool} \]
\[ a \neq b, b \neq c, c \neq d, d \neq e, e \neq a \]
\[ \neg P(a, b), \neg R(a, c) \]
\[ \forall xy. P(x, y) \lor R(x, y) \]

\[ S = \{a, b, c, d, e\} \]
FMF: Example

\[ a, b, c, d, e : S \]
\[ P, R : (S, S) \rightarrow \text{Bool} \]
\[ a \neq b, \ b \neq c, \ c \neq d, \ d \neq e, \ e \neq a \]
\[ \neg P(a, b), \ \neg R(a, c) \]
\[ \forall xy. \ P(x, y) \lor R(x, y) \]
FMF: Example

\[ a,b,c,d,e:S \]
\[ P,R:(S,S) \rightarrow \text{Bool} \]
\[ a \neq b, b \neq c, c \neq d, d \neq e, e \neq a \]
\[ \neg P(a,b), \neg R(a,c), a = d, c = e \]
\[ \forall xy. P(x,y) \lor R(x,y) \]

EXAMPLE...

\[ S = \{a, b, c\} \]
Finite Model Finding: Model-Based Instantiation

\[ \exists xy : S. \neg Q(x, y) \]

- Construct candidate model \( M \)

\[ S = \{a, b, c, d, e\} \]
\[ \{Q \rightarrow Q^M, \ldots\} \]
Finite Model Finding : Model-Based Instantiation

- Evaluate quantified formulas based on $Q^M$

$\forall xy:S.Q(x,y)$

$S=\{a,b,c,d,e\}$

$\{Q\rightarrow Q^M,\ldots\}$

[Reynolds/Tinelli/Goel/Krstic/Barrett/Deters CADE13]
Finite Model Finding: Model-Based Instantiation

- Only add instances that evaluate to F in $Q^M$

$\Rightarrow$ Significantly increased scalability

[$\text{Reynolds/Tinelli/Goel/Krstic/Barrett/Deters CADE13}$]
FMF: Example

a, b, c, d, e : S
P, R : (S, S) → Bool
a ≠ b, b ≠ c, c ≠ d, d ≠ e, e ≠ a
¬P(a, b), ¬R(a, c)
∀xy. P(x, y) ∨ R(x, y)
FMF: Example

\(a, b, c, d, e : S\)

\(P, R : (S, S) \rightarrow \text{Bool}\)

\(a \neq b, b \neq c, c \neq d, d \neq e, e \neq a\)

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

\(\forall xy. P(x, y) \lor R(x, y)\)

\(P := \lambda xy. (x \neq a \lor y \neq b)\)

\(R := \lambda xy. (x \neq a \lor y \neq c)\)
**FMF: Example**

\[ a, b, c, d, e : S \]

\[ P, R : (S, S) \rightarrow \text{Bool} \]

\[ a \neq b, b \neq c, c \neq d, d \neq e, e \neq a \]

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

\[ \forall xy. P(x, y) \lor R(x, y) \]

\[ P := \lambda xy. (x \neq a \lor y \neq b) \]

\[ R := \lambda xy. (x \neq a \lor y \neq c) \]

---

**Truth Tables:**

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>R</th>
<th>(P \lor R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>F</td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example Diagram:**

- **P:**
  - a: F
  - b, c, d, e: T

- **R:**
  - a: F
  - b, c, d, e: T

- **(P \lor R):**
  - a: T
  - b, c, d, e: T
Finite Model Finding in CVC4

• **Sound** for both “sat” and “unsat”

• **Finite-model complete**
  - If there is a finite model, CVC4 will eventually find it
    (when all quantification is over sorts that are interpreted as finite)

• Refutationally **incomplete** in general
  - But regardless, is often able to answer “unsat”
Extension: Bounded Integer Quantification

• \( \forall x: \text{Int. } 0 \leq x < t \Rightarrow P(x) \)

\[
\begin{align*}
& t \leq 0 \quad \neg t \leq 0 \\
& \text{Search for models where } t=0 \\
& \text{If none exist, search for models where } t=1 \\
& t \leq 1 \quad \neg t \leq 1 \\
& \text{If none exist, search for models where } t=1 \\
& t \leq 2 \quad \neg t \leq 2 \\
& \text{etc.}
\end{align*}
\]
Extension: Bounded Length Strings

• Given input \( F[s_1, \ldots, s_n] \) for strings \( s_1 \ldots s_n \):

\[
\Sigma_{i=1 \ldots n} |s_i| \leq 0
\]

Search for models where sum of lengths=0

\[
\Sigma_{i=1 \ldots n} |s_i| \leq 1
\]

Search for models where sum of lengths=1

\[
\Sigma_{i=1 \ldots n} |s_i| \leq 2
\]

Search for models where sum of lengths=2

\[
\neg \Sigma_{i=1 \ldots n} |s_i| \leq 0
\]

\[
\neg \Sigma_{i=1 \ldots n} |s_i| \leq 1
\]

\[
\neg \Sigma_{i=1 \ldots n} |s_i| \leq 2
\]

etc.

EXAMPLE...
Synthesis: Motivation

• Synthesis Problem: \( \exists f. \forall x. P(f, x) \)
  
There exists a function \( f \) such that for all \( x \), \( P(f, x) \)

• Most existing approaches for synthesis
  • Rely on specialized solver that makes subcalls to an SMT Solver

• CVC4 has approach for synthesis, which is entirely inside SMT solver
Example: Max of Two Integers

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

• Specifies that \( f \) computes the maximum of integers \( x \) and \( y \)
• Solution:

\[
f := \lambda xy. \text{ite}(x \geq y, x, y)
\]

\[
\begin{align*}
\exists f. \forall xy. (f(x, y) &\geq x \land f(x, y) \geq y \land \\
(f(x, y) &\leq x \lor f(x, y) = y))
\end{align*}
\]
How does an SMT solver handle Max example?

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

- Challenge: quantification over function \( f \)
  - No SMT solvers directly support second-order quantification
How does an SMT solver handle Max example?

\[ f : \text{Int} \times \text{Int} \to \text{Int} \]
\[ \forall xy. (f(x,y) \geq x \land f(x,y) \geq y \land (f(x,y) = x \lor f(x,y) = y)) \]

• Direct approach:
  • Treat \( f \) as an \textit{uninterpreted function}
  • Succeed if SMT solver can find correct interpretation of \( f \)

\[ \Rightarrow \text{This is challenging} \]
  • How does the solver know the right interpretation for \( f \) to pick?
How does an SMT solver handle Max example?

$$\exists f . \forall x y . (f(x,y) \geq x \land f(x,y) \geq y \land (f(x,y) = x \lor f(x,y) = y))$$
How does an CVC4 handle Max example?

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

• Alternative:
  • This property is **single invocation**
    • All occurrences of \( f \) are of the form \( f(x, y) \)
  ... and thus, can be converted to a first-order quantification
    • Introduce first-order variable \( g \)
    • Push quantification downwards “anti-skolemization”
How does an CVC4 handle Max example?

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

Convert to first-order

\[ \forall xy. \exists g. (g \geq x \land g \geq y \land (g = x \lor g = y)) \]

[Reynolds/Deters/Kuncak/Tinelli/Barrett CAV15]
How does an CVC4 handle Max example?

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

Convert to first-order

\[ \forall xy. \exists g. (g \geq x \land g \geq y \land (g = x \lor g = y)) \]

• Problem is now:
  • First-order, linear (integer) arithmetic, with one quantifier alternation

\[ \Rightarrow \text{CVC4 has specialized instantiation procedure} \]

[Reynolds/Deters/Kuncak/Tinelli/Barrett CAV15]
Max Example

∀xy. ∃g. (g ≥ x ∧ g ≥ y ∧ (g = x ∨ g = y))
Max Example

\[ \forall x y. \, \exists g. \text{isMax}(g, x, y) \]
Max Example

\[ \forall x y. \exists g. \text{isMax}(g,x,y) \]

• Goal: show the above formula is sat
Max Example

\[ \exists x y. \forall g. \neg \text{isMax}(g, x, y) \]

• Since \( F \) is LIA-sat if and only if \( \neg F \) is LIA-unsat, 
  \[ \implies \] Suffices to show that negation is unsat
Max Example

\[ \forall g. \neg \text{isMax}(g, a, b) \]

- Skolemize, for fresh constants \( a \) and \( b \)
Max Example

∀g. ¬isMax(g,a,b)

Ground Solver

Quantifiers Module
Max Example

• Which instances of $\forall g. \neg \text{isMax}(g, a, b)$ do we consider?
Counterexample-Guided Instantiation

- **Idea**: choose instances of $\forall g. \neg \text{isMax}(g,a,b)$ based on models for “counterexample” fresh constant $c$
Counterexample-Guided Instantiation

- $\text{isMax}(c,a,b)$
- $\ldots$
- $\forall g. \neg \text{isMax}(g,a,b)$

- If ground constraints without CE is unsat, answer “unsat”
Counterexample-Guided Instantiation

- $\text{isMax}(c,a,b)$
- $\text{isMax}(c,a,b)$
- $\ldots$
- $\forall g. \neg \text{isMax}(g,a,b)$
- $\text{unsat}$

**Ground Solver**

- $\text{sat}$

- Else, if ground constraints with CE is unsat, answer “sat”
isMax(c,a,b) → sat, where c=a

∀g.¬isMax(g,a,b) → Ground Solver

Counterexample-Guided Instantiation
Counterexample-Guided Instantiation

\[ \text{isMax}(c,a,b) \]
\[ \neg \text{isMax}(a,a,b) \]

\[ \forall g. \neg \text{isMax}(g,a,b) \]

Instance \( \{ g \mapsto a \} \)

sat, where \( c = a \)

Ground Solver

Quantifiers Module
Counterexample-Guided Instantiation

\[ \text{isMax}(c, a, b) \]
\[ \neg \text{isMax}(a, a, b) \]
\[ \forall g. \neg \text{isMax}(g, a, b) \]
Counterexample-Guided Instantiation

\[ \exists g. \neg \text{isMax}(g, a, b) \]
\[ \neg \text{isMax}(a, a, b) \]

\text{sat, where } c = b
Counterexample-Guided Instantiation

\[ \forall g. \neg \text{isMax}(g, a, b) \]

\[ \neg \text{isMax}(a, a, b) \]

\[ \neg \text{isMax}(b, a, b) \]

\[ \text{sat, where } c = b \]

Instance
\[ \{ g \mapsto b \} \]

Ground Solver

Quantifiers Module
Counterexample-Guided Instantiation

\[ \forall g. \neg \text{isMax}(g, a, b) \]

\[ \neg \text{isMax}(c, a, b) \]

\[ \neg \text{isMax}(a, a, b) \]

\[ \neg \text{isMax}(b, a, b) \]

Ground Solver

Quantifiers Module

unsat
Counterexample-Guided Instantiation

\( \text{isMax}(c, a, b) \)

\( \ldots \)

\( \forall g. \neg \text{isMax}(g, a, b) \)

Ground Solver

Quantifiers Module
Counterexample-Guided Instantiation

\[ c \geq a, c \geq b, c = a \lor c = b \]

\[ \ldots \]

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

Ground Solver

Quantifiers Module
Counterexample-Guided Instantiation

\[ c \geq a, c \geq b, c = a \lor c = b \]

\[ \ldots \]

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

Ground Solver

Quantifiers Module

sat
Counterexample-Guided Instantiation

\[ c \geq a, c \geq b, c = a \lor c = b \]

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

- Take maximal lower bound for \( c \) in model \( M \)
Counterexample-Guided Instantiation

\[ c \geq a, c \geq b, c = a \lor c = b \]
\[ a < a \lor a < b \lor (a \neq a \land a \neq b) \]

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

Instance
\{g \mapsto a\}

sat, model \( M \), \( \max(a^M, b^M) = a^M \)
Counterexample-Guided Instantiation

c ≥ a, c ≥ b, c = a ∨ c = b
a < a ∨ a < b ∨ (a ≠ a ∧ a ≠ b)

Ground Solver

Quantifiers Module

∀g. (g < a ∨ g < b ∨ (g ≠ a ∧ g ≠ b))
Counterexample-Guided Instantiation

\[ c \geq a, c \geq b, c = a \lor c = b \]
\[ a \leq a \lor a < b \lor (a \neq a \land a \neq b) \]

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

Ground Solver

Quantifiers Module
Counterexample-Guided Instantiation

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

\[ c \geq a, c \geq b, c = a \lor c = b \]

Ground Solver

Quantifiers Module

\[ a < b \]
Counterexample-Guided Instantiation

\[
\begin{align*}
\forall g. & (g < a \vee g < b \vee (g \neq a \land g \neq b)) \\
\text{Ground Solver} & : c \geq a, c \geq b, c = a \lor c = b \\
a < b & \\
\text{Quantifiers Module} & : \text{sat}
\end{align*}
\]
Counterexample-Guided Instantiation

- \( c \geq a, c \geq b, c = a \lor c = b \)
- \( a < b \)
- \( \forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b)) \)

- Ground Solver
- Quantifiers Module

- sat, model \( M \), \( \max(a^M, b^M) = b^M \)
- Take maximal lower bound for \( c \) in model \( M \)
Counterexample-Guided Instantiation

Ground Solver

Instance \{g \mapsto b\}

Quantifiers Module

\[ c \geq a, c \geq b, c = a \lor c = b \]
\[ a < b \]
\[ b < a \lor b < b \lor (b \neq a \land b \neq b) \]

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

sat, model \( M \), \( \max(a^M, b^M) = b^M \)
Counterexample-Guided Instantiation

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

\[ c \geq a, c \geq b, c = a \lor c = b \]

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

Ground Solver

Quantifiers Module

unsat
Synthesis: Solutions

$\exists f. \forall x y. \text{isMax}(f(x, y), x, y)$

Ground Solver

Quantifiers Module
Synthesis: Solutions

Negate, convert to FO

∀g. ¬isMax(g, a, b)
Synthesis: Solutions

\[ \neg \text{isMax}(a, a, b) \]
\[ \neg \text{isMax}(b, a, b) \]

\[ \exists f. \forall xy. \text{isMax}(f(x, y), x, y) \]

\[ \forall g. \neg \text{isMax}(g, a, b) \]

Ground Solver

Quantifiers Module

unsat
Synthesis: Solutions

- $\neg \text{isMax}(a, a, b)$
- $\neg \text{isMax}(b, a, b)$

$\exists f. \forall xy. \text{isMax}(f(x, y), x, y)$

$\forall g. \neg \text{isMax}(g, a, b)$

$\neg \text{isMax}(a, a, b)$
$\neg \text{isMax}(b, a, b)$

$\text{unsat}$ $\Rightarrow$ implies original synthesis conjecture has a solution
Synthesis: Solutions

Ground Solver

- \neg isMax(a, a, b)
- \neg isMax(b, a, b)

Quantifiers Module

\forall g. \neg isMax(g, a, b)

\exists f. \forall xy. isMax(f(x, y), x, y)

\Rightarrow Solution can be extracted from unsatisfiable core of instantiations a/g, b/g

\[ f := \lambda xy. \text{ite}( \text{isMax}(a, a, b), a, b)[x/a][y/b] \]
Desired function, after simplification

Synthesis: Solutions
Ground Solver
Quantifiers Module

\[ \neg \text{isMax}(a, a, b) \]
\[ \neg \text{isMax}(b, a, b) \]

\[ \exists f. \forall xy. \text{isMax}(f(x, y), x, y) \]

\[ f := \lambda xy. \text{ite}(x \geq y, x, y) \]

\[ \Rightarrow \text{unsat} \]

\[ \Rightarrow \text{Desired function, after simplification} \]

EXAMPLE...
Counterexample-Guided Instantiation

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

• Consider example: \( \forall g . (g > a \lor g < b) \)
Counterexample-Guided Instantiation

\[ c \leq a, c \geq b \]

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

Ground Solver

Quantifiers Module

sat
Counterexample-Guided Instantiation

\[ c \leq a, \, c \geq b \]

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

- Take maximal lower bound for \( c \) in model \( M \)

sat, model \( M \), \( \max (b^M) = b^M \)

- Take maximal lower bound for \( c \) in model \( M \)
Counterexample-Guided Instantiation

\[ c \leq a, \ c \geq b \]

\[ b > a \lor b < b \]

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

Instance \{g → b\}

Ground Solver

Quantifiers Module

sat, model \( M \), \( \max(b^M) = b^M \)
Counterexample-Guided Instantiation

\begin{align*}
c \leq a, c \geq b \\
b > a \lor b < b
\end{align*}

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

Ground Solver

Quantifiers Module
Counterexample-Guided Instantiation

\[ c \leq a, c \geq b \]

\[ b > a \]

\[ \forall g \cdot (g > a \lor g < b) \]

\[ \{b > a\} \text{ is sat} \]
Counterexample-Guided Instantiation

- \{b>a\} is sat
- ...but \{c≤a, c≥b, b>a\} is unsat
  \[⇒ \text{In other words, there is no model for counterexample } c\]
Counterexample-Guided Instantiation

\[ c \leq a, c \geq b \]
\[ b > a \]
\[ \text{Ground Solver} \]
\[ \forall g. (g > a \lor g < b) \]
\[ \text{Quantifiers Module} \]
\[ \text{sat} \]
Counterexample-Guided Instantiation

\[ c \leq a, \ c \geq b \]

\[ b > a \]

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

⇒ All models satisfying \( b > a \) also satisfy \( \forall g. \ (g > a \lor g < b) \)
Counterexample-Guided Instantiation

• For linear real and integer arithmetic:
  • With one quantifier alternation:
    • **Sound** and **complete** (terminating) [Reynolds/King/Kuncak, draft 2015]
  • With arbitrary quantifier alternations:
    • Effective in practice, for both “sat” and “unsat”
Counterexample-Guided Instantiation in CVC4

- Highly competitive for synthesis applications
  - Won, GENERAL/LIA divisions of SygusComp 2015

- Applicable to arbitrary quantified formulas as well
  - Won, LIA/LRA divisions of SMT COMP 2015
  - Won, first-order theorems division of CASC J7
  - 2nd place, first-order theorems division of CASC 25
  - Won, first-order non-theorems division of CASC 25
Counterexample-Guided Instantiation in CVC4

• Highly competitive for synthesis applications
  • Won, GENERAL/LIA divisions of SygusComp 2015

• Applicable to arbitrary quantified formulas as well
  • Won, LIA/LRA divisions of SMT COMP 2015
  • Won, first-order theorems division of CASC J7
  • 2nd place, first-order theorems division of CASC 25
  • Won, first-order non-theorems division of CASC 25
Conclusion

• **CVC4 + quantified formulas** can be used for:
  • Theorem proving and verification
  • Finite model finding (**finite-model-find**)
  • Function synthesis (**cegqi**, on *.sl)
  • ...and more:
    • Inductive Theorem Proving (**quant-ind**)
    • Model finding for recursive functions (**fmf-fun**)
    • ...

⇒ All techniques work in combination with the wide array of ground theories CVC4 supports
Thanks!

• CVC4 is publicly available at:

  http://cvc4.cs.nyu.edu/web/