From Counter-Model-based Quantifier Instantiation to Quantifier Elimination in SMT

CADE 27

Andrew Reynolds Cesare Tinelli

August 29, 2019

The University of Iowa

Acknowledgments

Collaborators:

Kshitij Bansal*, **Haniel Barbosa***, **Clark Barrett***, François Bobot*, **Martin Brain***, Morgan Deters*, Pascal Fontaine, Amit Goel, Liana Hadarean*, Dejan Jovanovič*, Guy Katz*, **Tim King***, Sava Krstič, **Viktor Kunčak**, Tianyi Liang*, Paul Meng*, Mudathir Mohammed*, **Leonardo de Moura**, **Aina Niemetz***, Andres Noetzli*, **Mathias Preiner***, **Arjun Viswanathan**, **Yoni Zohar***

(*) past or present CVC4 developer

Outline

Introduction

Satisfiability Modulo Theories (SMT)

- Subfield of automated deduction focussing on specialized reasoning in certain logical theories
- Used in large and diverse number of applications
- Traditionally, strong on quantifier-free reasoning
- However, many applications require a mix of built-in **and** axiomatically defined symbols

Need for Quantifiers in SMT Applications

Automated Theorem Proving

Background axioms: $\forall x.\, g(e,x) = g(x,e) = x,\ \forall x.\, g(x,i(x)) = e$
$\forall x.\, g(x,g(y,z)) = g(g(x,y),x)$

Software Verification

Unfolding: $\forall x.\, \mathit{foo}(x) = \mathit{bar}(x+1)$
Code contracts: $\forall x.\, \mathit{pre}(x) \Rightarrow \mathit{post}(f(x))$
Frame axioms: $\forall x.\, x \neq t \Rightarrow A'(x) = A(x)$

Function Synthesis

Synthesis conjectures: $\forall i{:}\mathit{input}.\, \exists o{:}\mathit{output}.\, R[o,i]$

Planning

Specifications: $\exists p{:}\mathit{plan}\, \forall t{:}\mathit{time}\, F[p,t]$

Grand Challenge in Automated Deduction

Reasoning efficiently about

theory symbols and quantifiers

First-order theorem provers focus mostly on reasoning with quantifiers but some have been extended to theory reasoning:

Vampire, E, SPASS, Beagle

- First-order resolution/superposition [Nieuwenhuis&Rubio 1999, Prevosto&Waldman 2006, Althaus et al. 2009, Baumgartner&Waldman 2013]
- AVATAR [Voronkov 2014, Reger et al. 2015]

iProver

- InstGen calculus [Ganzinger&Korovin 2003]

Princess

- Sequent calculus [Rümmer 2008]

Reasoning with Theories and Quantifiers in FOL — ATP case

First-order theorem provers focus mostly on reasoning with quantifiers but some have been extended to theory reasoning:

Vampire, E, SPASS, Beagle

- First-order resolution/superposition [Nieuwenhuis&Rubio 1999, Prevosto&Waldman 2006, Althaus et al. 2009, Baumgartner&Waldman 2013]
- AVATAR [Voronkov 2014, Reger et al. 2015]

iProver

- InstGen calculus [Ganzinger&Korovin 2003]

Princess

- Sequent calculus [Rümmer 2008]

Reasoning with Theories and Quantifiers in FOL — SMT case

SMT solvers focus mostly on quantifier-free theory reasoning but some have been extended to reasoning with quantifiers:

Alt-Ergo, CVC3, CVC4, veriT, Z3

- Some superposition-based [deMoura et al. 2009]
- Most instantiation-based [Detlefs et al. 2005, deMoura et al. 2007, Ge et al. 2007, ...]

Reasoning with Theories and Quantifiers in FOL — SMT case

SMT solvers focus mostly on quantifier-free theory reasoning but some have been extended to reasoning with quantifiers:

Alt-Ergo, CVC3, CVC4, veriT, Z3

- Some superposition-based [deMoura et al. 2009]
- Most instantiation-based [Detlefs et al. 2005, deMoura et al. 2007, Ge et al. 2007, ...]

SMT Solvers using Quantifier Instantiation

Traditionally:

- E-matching [Detlefs et al. 2005, Bjørner et al. 2007, CADE 2007]

More recently:

- Model-Based Instantiation [Ge et al. 2009, CADE 2013]
- Conflict-Based Instantiation [FMCAD 2014, TACAS 2017]
- Theory-specific Approaches
  - Linear arithmetic [Bjørner 2012, CAV 2015, Janota et al. 2015]
  - Bit-Vectors [Wintersteiger et al. 2013, Dutertre 2015]

SMT Solvers using Quantifier Instantiation

Implemented in

Alt-Ergo, CVC3-4, FX7, Simplify, veriT, Z3

Traditionally:

- E-matching [Detlefs et al. 2005, … 2007]

More recently:

- Model-Based Instantiation [Ge et al. 2009, CADE 2013]
- Conflict-Based Instantiation [FMCAD 2014, TACAS 2017]
- Theory-specific Approaches
  - Linear arithmetic [Bjørner 2012, CAV 2015, Janota et al. 2015]
  - Bit-Vectors [Wintersteiger et al. 2013, Dutertre 2015]

SMT Solvers using Quantifier Instantiation

Implemented in

Traditionally:

- E-matching [Detlefs et al. 2005, ... 2007]

Alt-Ergo, CVC3-4, FX7, Simplify, veriT, Z3

More recently:

- Model-Based Instantiation [G...

CVC4, Z3

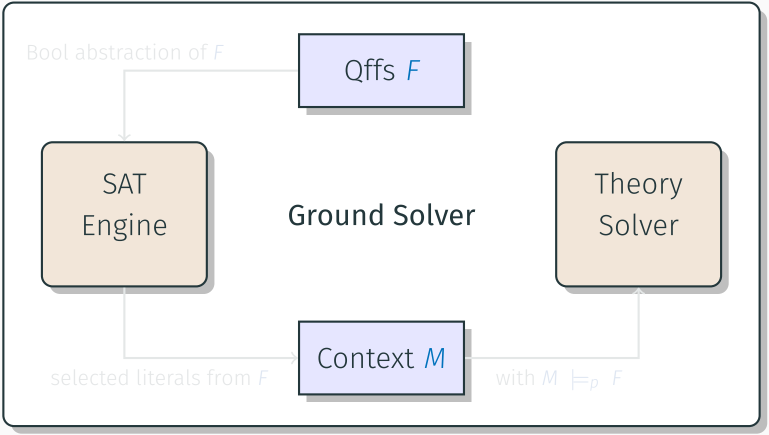
- Conflict-Based Instantiation [...

CVC4, veriT

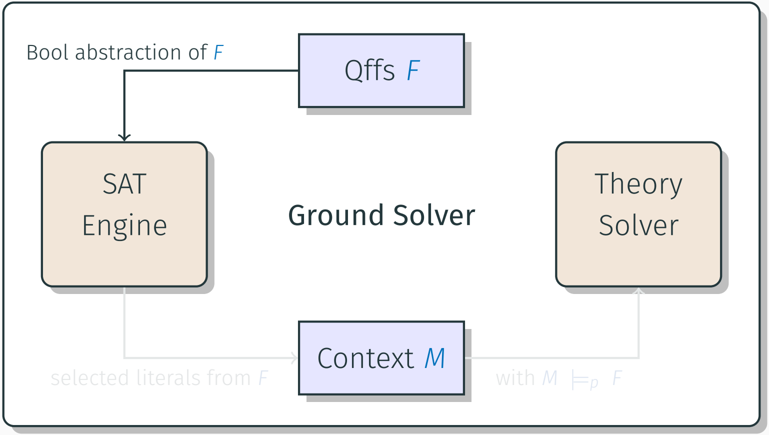
- Theory-specific Approaches
  - Linear arithmetic [Bjørner 2... . 2015]
  - Bit-Vectors [Wintersteiger e...

CVC4, Yices, veriT, Z3

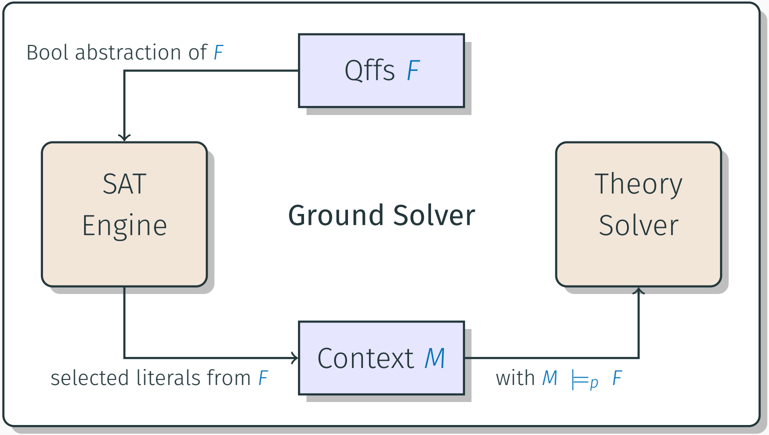
SMT Solvers for Ground Formulas



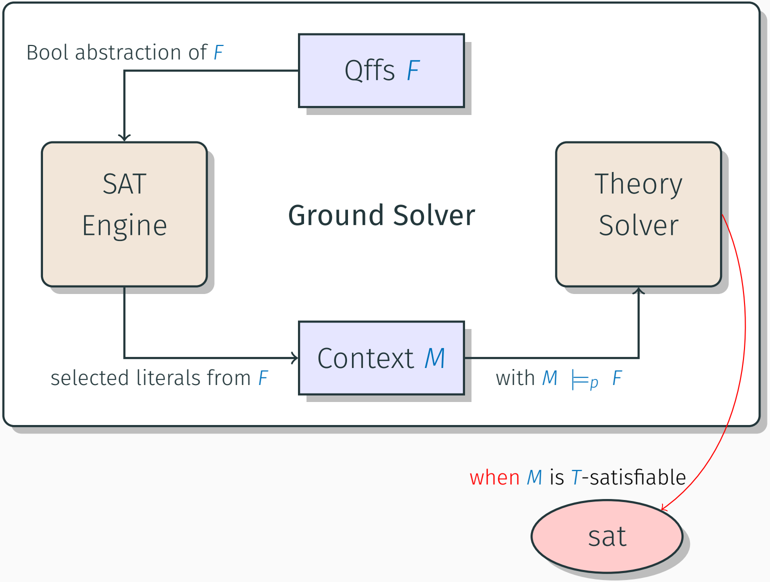
SMT Solvers for Ground Formulas



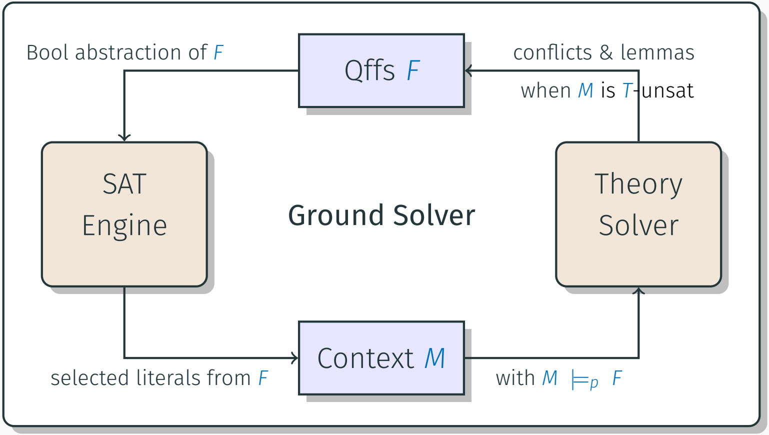
SMT Solvers for Ground Formulas



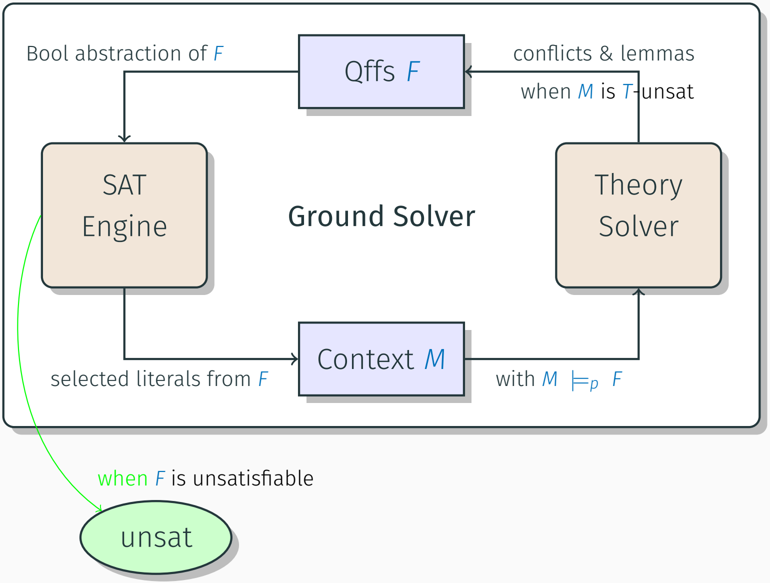
SMT Solvers for Ground Formulas



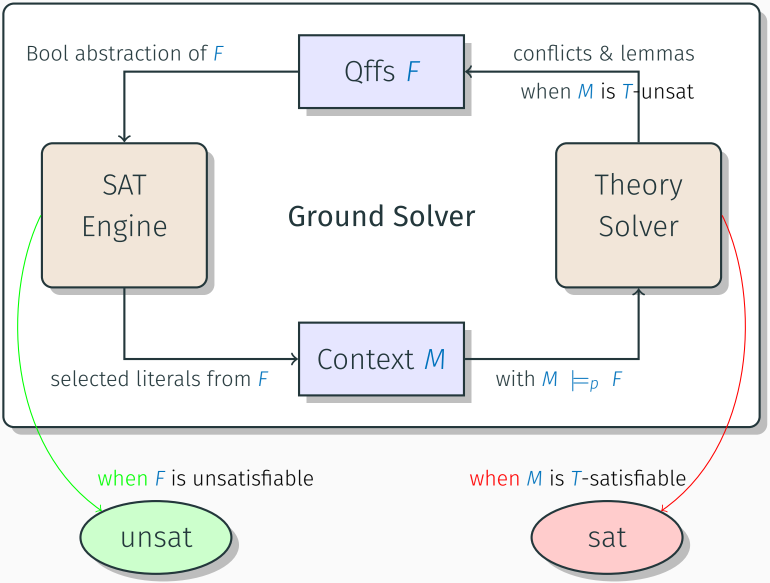
SMT Solvers for Ground Formulas



SMT Solvers for Ground Formulas

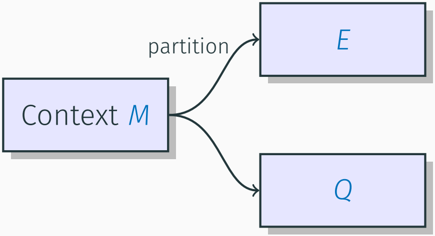


SMT Solvers for Ground Formulas



Adding Quantifier Instantiation to SMT Solvers

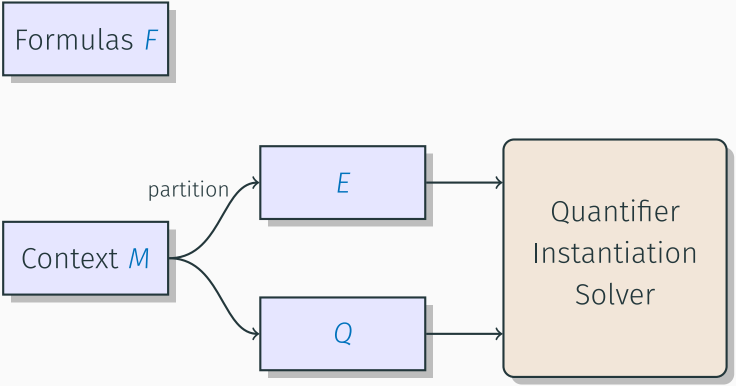
Formulas F $\quad \{a \approx b \Rightarrow \underbrace{\forall x\, \forall y\, P(x) \vee Q(x,y)}_{q}, \ldots\}$



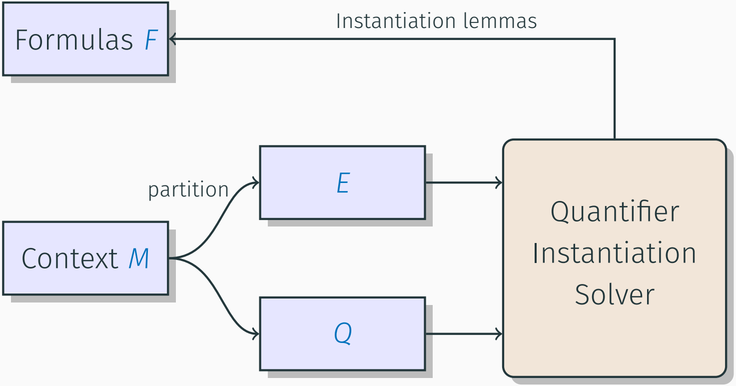
$\{a \approx b,\ f(a) \neq b,\ P(a) \approx \bot,\ \ldots\}$

$\{\forall x\, \forall y\, P(x) \vee Q(x,y), \ldots\}$

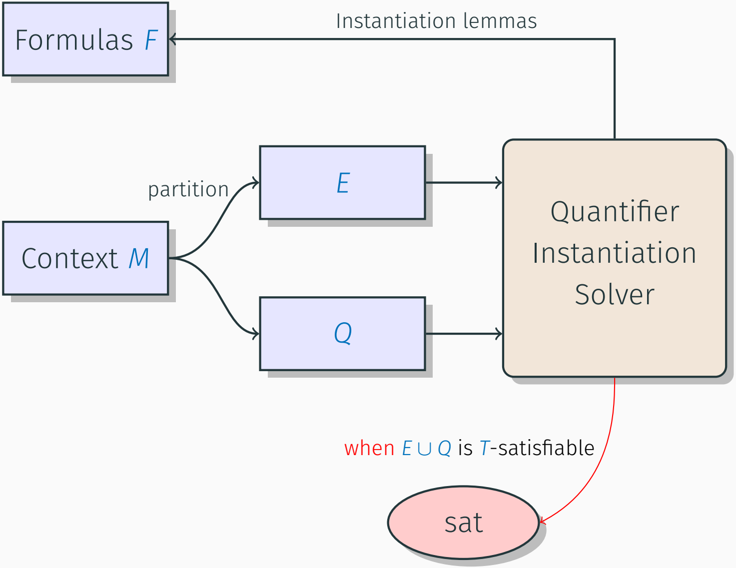
Adding Quantifier Instantiation to SMT Solvers



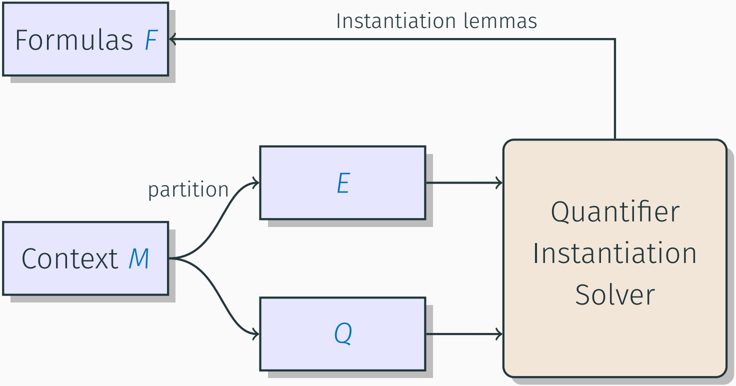
Adding Quantifier Instantiation to SMT Solvers



Adding Quantifier Instantiation to SMT Solvers



Adding Quantifier Instantiation to SMT Solvers



Main Questions:

- Which instantiations likely lead to unsat?
- When can we answer sat?

Quantifier Instantiation

Outline

Instantiations by E-matching

Basic Idea: Choose instances based on pattern matching over *E-graph* of asserted ground (dis-)equalities [Nelson 80]

Most widely used technique for refuting quantified problems in SMT

Exploited in:

- Software Verification
  (Boogie, Dafny, Leon, SPARK, Why3, ...)
- ...
- Automated Theorem Proving
  (Sledgehammer)

Instantiations by E-matching

Basic Idea: Choose instances based on pattern matching over *E-graph* of asserted ground (dis-)equalities [Nelson 80]

Most widely used technique for refuting quantified problems in SMT

Exploited in:

- Software Verification
  (Boogie, Dafny, Leon, SPARK, Why3, ...)
- ...
- Automated Theorem Proving
  (Sledgehammer)

Instantiations by E-matching

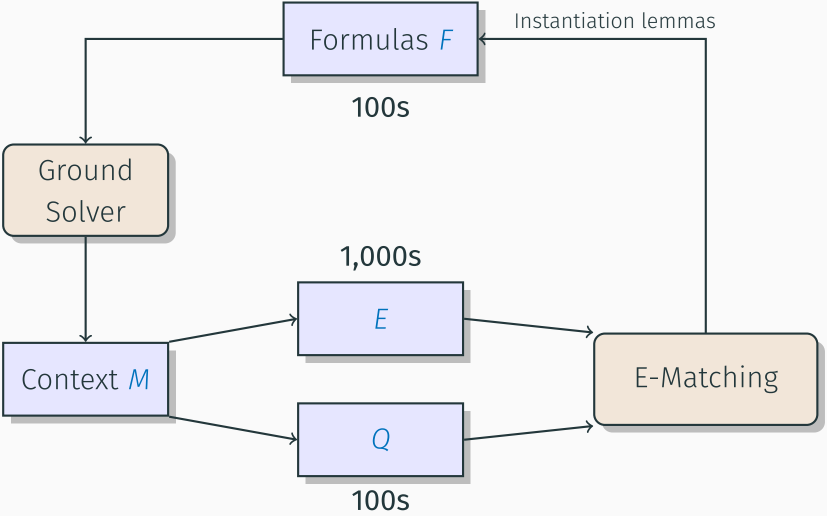
Basic Idea: Choose instances based on pattern matching over *E-graph* of asserted ground (dis-)equalities [Nelson 80]

Most widely used technique for refuting quantified problems in SMT

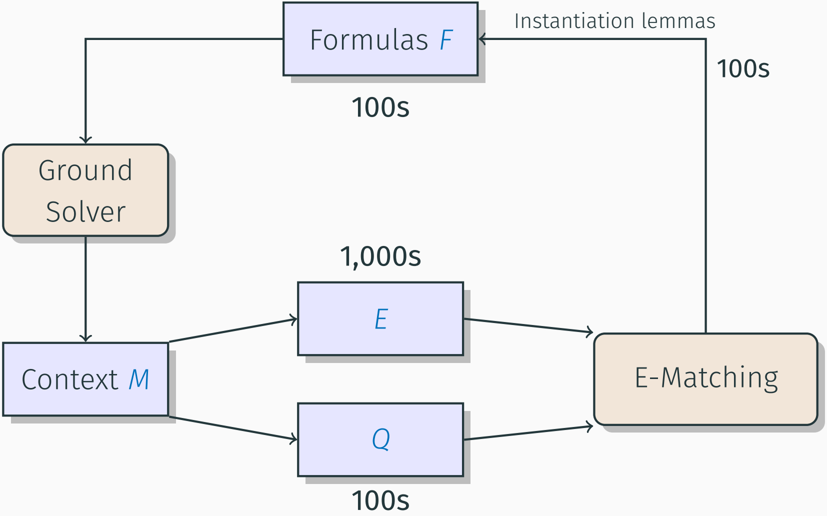
Exploited in:

- Software Verification
  (Boogie, Dafny, Leon, SPARK, Why3, ...)
- ...
- Automated Theorem Proving
  (Sledgehammer)

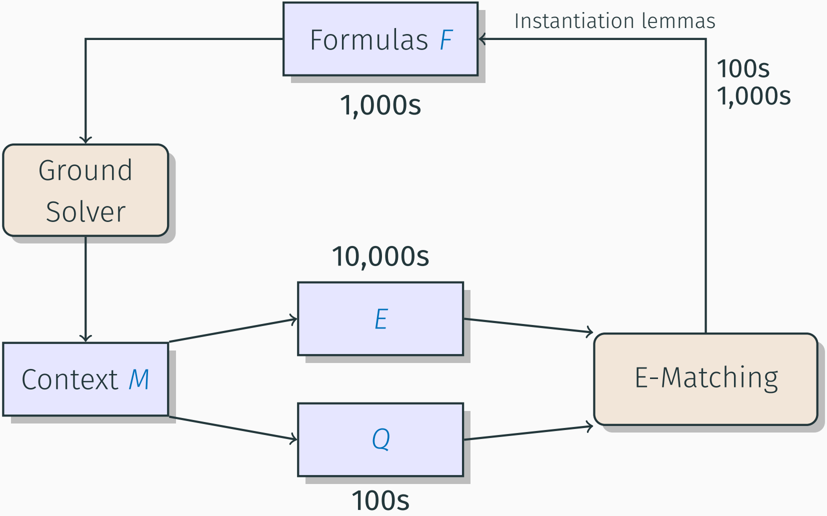
E-matching's Challenge #1 : Too Many Instances



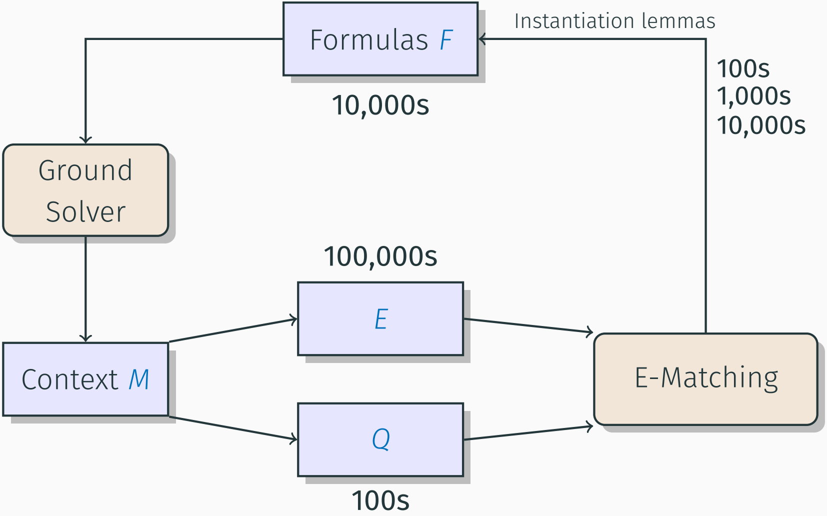
E-matching's Challenge #1 : Too Many Instances



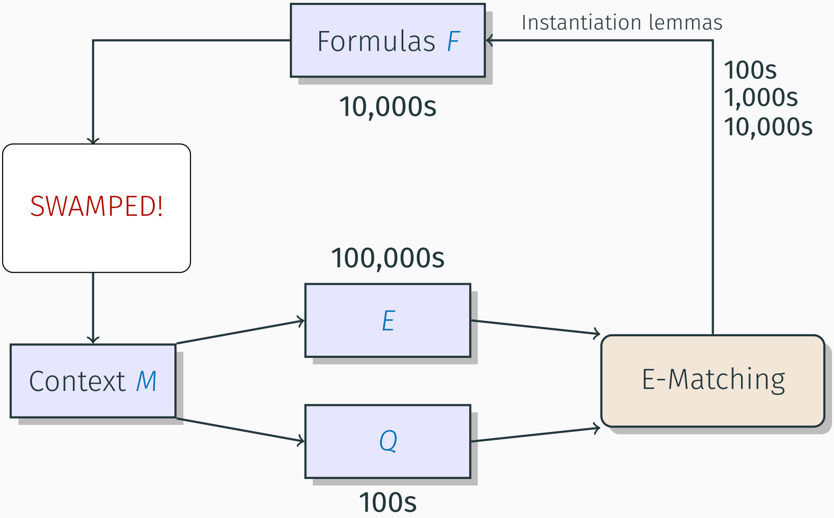
E-matching's Challenge #1 : Too Many Instances



E-matching's Challenge #1 : Too Many Instances

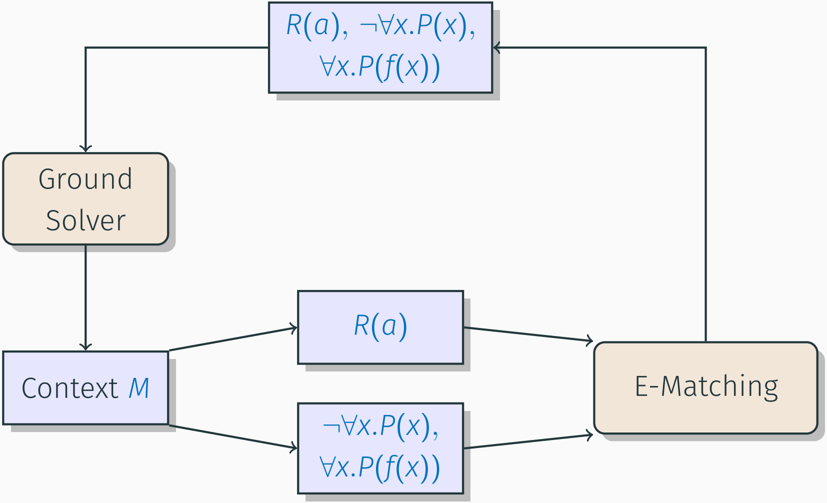


E-matching's Challenge #1 : Too Many Instances

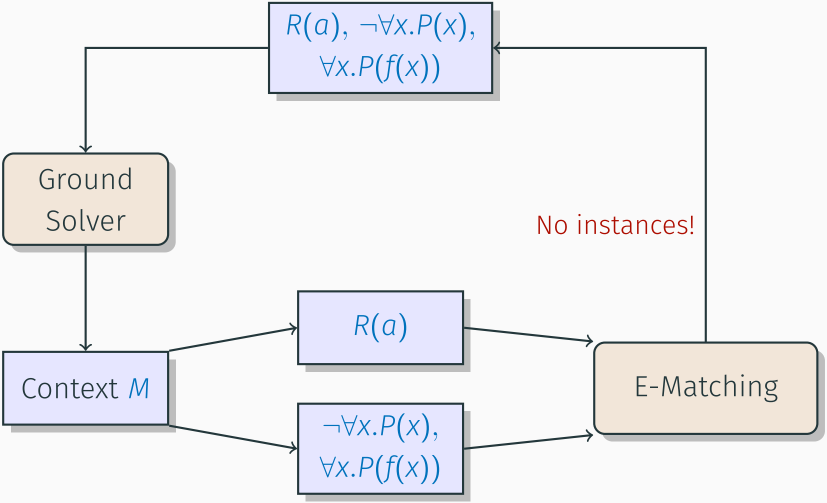


Ground solver gets overloaded and times out

E-matching's Challenge #2 : Incompleteness



E-matching's Challenge #2 : Incompleteness



Unsatisfiability goes undetected

Addressing E-matching's Challenges

Too many instances?

- ▷ Try *conflict-based instantiation* first [FMCAD 2014]

Apply E-matching

No instances and input may be satisfiable?

- ▷ Try *model-based instantiation* next [Ge&deMoura 2009]

Addressing E-matching's Challenges

Too many instances?

- ▷ Try *conflict-based instantiation* first [FMCAD 2014]

Apply E-matching

No instances and input may be satisfiable?

- ▷ Try *model-based instantiation* next [Ge&deMoura 2009]

Addressing E-matching's Challenges

Too many instances?

- ▷ Try *conflict-based instantiation* first [FMCAD 2014]

Apply E-matching

No instances and input may be satisfiable?

- ▷ Try *model-based instantiation* next [Ge&deMoura 2009]

Outline

Conflict-Based QI

Basic idea: Given $E \cup \{\forall \mathbf{x}.\, \varphi[\mathbf{x}], \ldots\}$,

- Try to find one *conflict instance* $\forall \mathbf{x}.\, \varphi[\mathbf{x}] \Rightarrow \varphi[\mathbf{t}]$ such that
$$E, \varphi[\mathbf{t}] \models_T \bot$$
- If this is possible, E-matching is not needed

Leads to fewer instances, improving ability to answer **unsat**

Conflict-Based QI

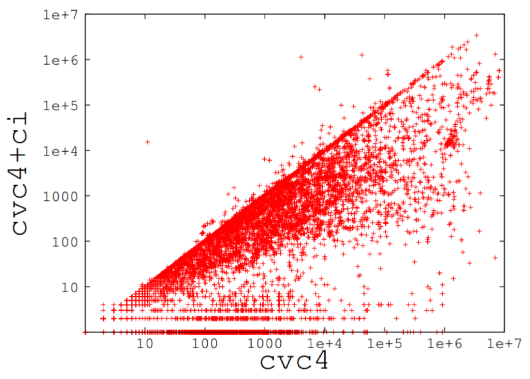
Basic idea: Given $E \cup \{\forall \mathbf{x}.\, \varphi[\mathbf{x}], \ldots\}$,

- Try to find one *conflict instance* $\forall \mathbf{x}.\, \varphi[\mathbf{x}] \Rightarrow \varphi[\mathbf{t}]$ such that
$$E, \varphi[\mathbf{t}] \models_T \bot$$
- If this is possible, E-matching is not needed

Leads to fewer instances, improving ability to answer **unsat**

Impact of Conflict-Based QI in CVC4

CBQI (cvc4+ci) needs 10^{-1}x instances to show **unsat** vs. E-matching alone



(evaluation on SMT-LIB, TPTP, and Isabelle benchmarks [FMCAD 2014])

CBQI's Challenge #1: Finding Conflicting Instances

Our solution: Construct instances via a stronger version of matching [FMCAD 2014]

Intuition: with $\forall \mathbf{x}.\, P[\mathbf{x}] \vee Q[\mathbf{x}]$ only match on $P[\mathbf{t}]$ where

$$P[\mathbf{t}] \equiv_{\mathrm{EUF}} \perp$$

Formalized as calculus based on ground E-(dis)unification [TACAS 2017]

CBQI's Challenge #2: Theory Symbols

Difficulty of finding conflicting instances in the presence of theory symbols:

$$E = \{f(1) \approx 5, \ldots\} \quad Q = \{\underbrace{\forall x, y. f(x+y) > x + 2 \cdot y}_{q}, \ldots\}$$

Generally, use fast and incomplete procedure for quantifiers + theories

CBQI's Challenge #2: Theory Symbols

Difficulty of finding conflicting instances in the presence of theory symbols:

$$E = \{f(1) \approx 5, \ldots\} \quad Q = \{\underbrace{\forall x, y. f(x+y) > x + 2 \cdot y}_{q}, \ldots\}$$

$$\downarrow$$

$$q \Rightarrow f(1) > 5$$

Generally, use fast and incomplete procedure for quantifiers + theories

CBQI's Challenge #2: Theory Symbols

Difficulty of finding conflicting instances in the presence of theory symbols:

$$E = \{f(1) \approx 5, \ldots\} \quad Q = \{\underbrace{\forall x, y.\, f(x+y) > x + 2 \cdot y}_{q}, \ldots\}$$

$$\downarrow$$

$$q \Rightarrow f(-3+4) > -3 + 2 \cdot 4$$

Generally, use fast and incomplete procedure for quantifiers + theories

CBQI's Challenge #2: Theory Symbols

Difficulty of finding conflicting instances in the presence of theory symbols:

$$E = \{f(1) \approx 5, \ldots\} \quad Q = \{\underbrace{\forall x, y.\, f(x+y) > x + 2 \cdot y}_{q}, \ldots\}$$

$$\downarrow$$

$$q \Rightarrow f(-3 + 4) > -3 + 2 \cdot 4$$

Generally, use fast and incomplete procedure for quantifiers + theories

Model-based Quantifier Instantiation

$E = \{\text{ground literals}\} \qquad Q = \{\text{quantified formulas}\}$

Basic idea:

If E-matching saturates, build a *candidate model* $\mathcal{I}$ satisfying E

1. Check if $\mathcal{I}$ also satisfies Q
   (using a ground satisfiability query)
2. If not, add instance of formula in Q falsified by $\mathcal{I}$
3. Repeat

Gives ability to answer **sat**

Model-based Quantifier Instantiation

$E = \{\text{ground literals}\} \qquad Q = \{\text{quantified formulas}\}$

Basic idea:

If E-matching saturates, build a *candidate model* $\mathcal{I}$ satisfying E

1. Check if $\mathcal{I}$ also satisfies Q
   (using a ground satisfiability query)
2. If not, add instance of formula in Q falsified by $\mathcal{I}$
3. Repeat

Gives ability to answer **sat**

Model-based Quantifier Instantiation

$$E = \{\text{ground literals}\} \qquad Q = \{\text{quantified formulas}\}$$

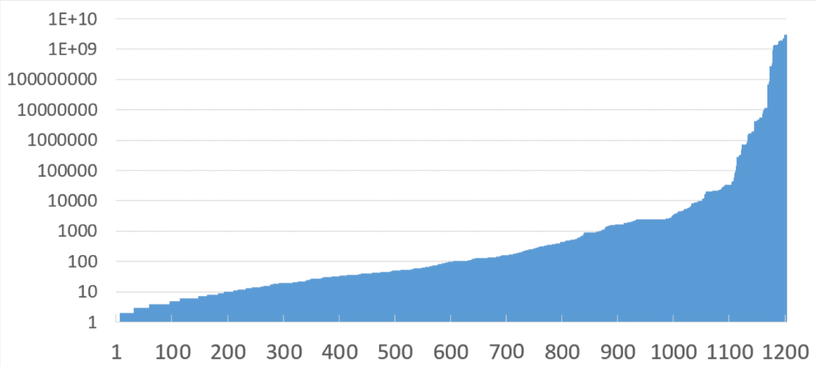
Basic idea:

If E-matching saturates, build a *candidate model* $\mathcal{I}$ satisfying E

1. Check if $\mathcal{I}$ also satisfies Q
   (using a ground satisfiability query)
2. If not, add instance of formula in Q falsified by $\mathcal{I}$
3. Repeat

Gives ability to answer **sat**

Impact of Model-Based QI in CVC4

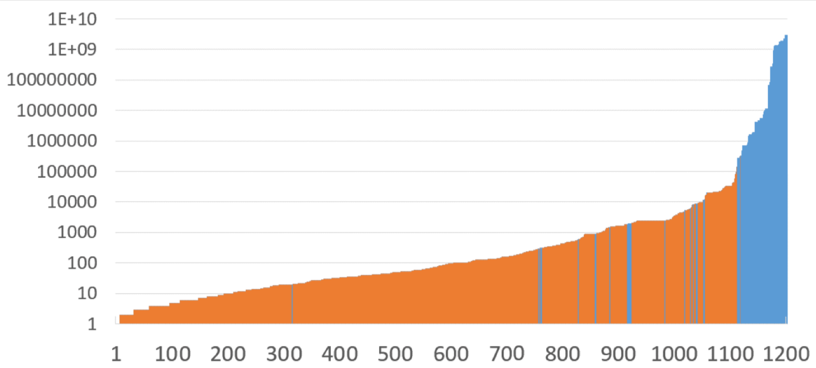


x = 1,203 satisfiable TPTP benchmarks

y = # of instances potentially generated by exhaustive instantiation

E.g. $4^3 = 64$ instances for $\forall x, y, z : \mathbf{A}. P(x, y, z)$ when $|A| = 4$

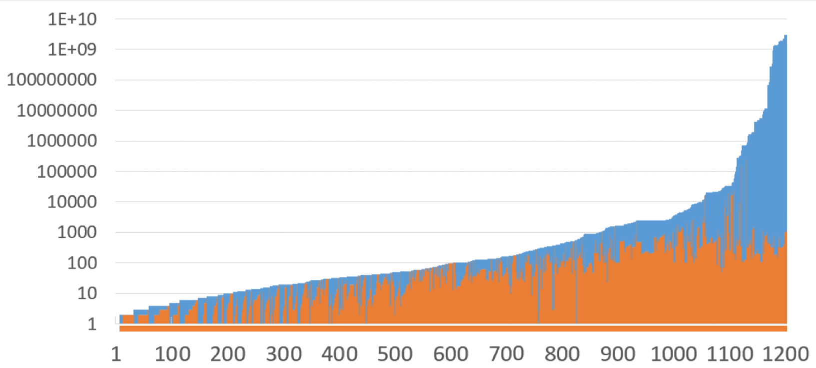
Impact of Model-Based QI in CVC4



CVC4 Finite Model Finding + Model-Based instantiation [CADE 2013]

Scales only up to $\sim$150K instances with a 30s timeout

Impact of Model-Based QI in CVC4



CVC4 Finite Model Finding + Model-Based instantiation [CADE 2013]

Scales to >2B instances with a 30s timeout,
generates only a fraction of possible instances

Model-Based QI: Challenges

How do we build interpretations $\mathcal{I}$?

Typically, build $\mathcal{I}$ where every function is *almost constant*:

$$f^{\mathcal{I}} := \lambda x.\, \mathsf{ite}(x = t_1, v_1, \mathsf{ite}(x = t_2, v_2, \ldots, \mathsf{ite}(x = t_n, v_n, v_{\mathrm{def}}) \ldots))$$

This works well in EUF

Model-Based QI: Challenges

How do we build interpretations $\mathcal{I}$?

However, more sophisticated models are needed when other theories are involved:

$\forall x, y : \mathbf{Int}.\,(f(x, y) \geq x \wedge f(x, y) \geq y)$ $\qquad$ $f^{\mathcal{I}} := \lambda x, y : \mathbf{Int}.\,\mathbf{ite}(x \geq y, x, y)$

Model-Based QI: Challenges

How do we build interpretations $\mathcal{I}$?

However, more sophisticated models are needed when other theories are involved:

$\forall x, y : \textbf{Int}.\, (f(x, y) \geq x \wedge f(x, y) \geq y)$

$f^{\mathcal{I}} := \lambda x, y : \textbf{Int}.\, \textbf{ite}(x \geq y, x, y)$

$\forall x : \textbf{Int}.\, 3 \cdot g(x) + 5 \cdot h(x) = x$

$g^{\mathcal{I}} := \lambda x.\, x - 3 \cdot x$
$h^{\mathcal{I}} := \lambda x.\, 2 \cdot x$

Model-Based QI: Challenges

How do we build interpretations $\mathcal{I}$?

However, more sophisticated models are needed when other theories are involved:

$\forall x, y : \mathbf{Int}.\, (f(x, y) \geq x \wedge f(x, y) \geq y)$	$f^{\mathcal{I}} := \lambda x, y : \mathbf{Int}.\, \mathbf{ite}(x \geq y, x, y)$
$\forall x : \mathbf{Int}.\, 3 \cdot g(x) + 5 \cdot h(x) = x$	$g^{\mathcal{I}} := \lambda x.\, x - 3 \cdot x$ $h^{\mathcal{I}} := \lambda x.\, 2 \cdot x$
$\forall x, y : \mathbf{Int}.\, u(x + y) + 11 \cdot v(w(x)) = x$	??

Model-Based QI: Challenges

How do we build interpretations $\mathcal{I}$?

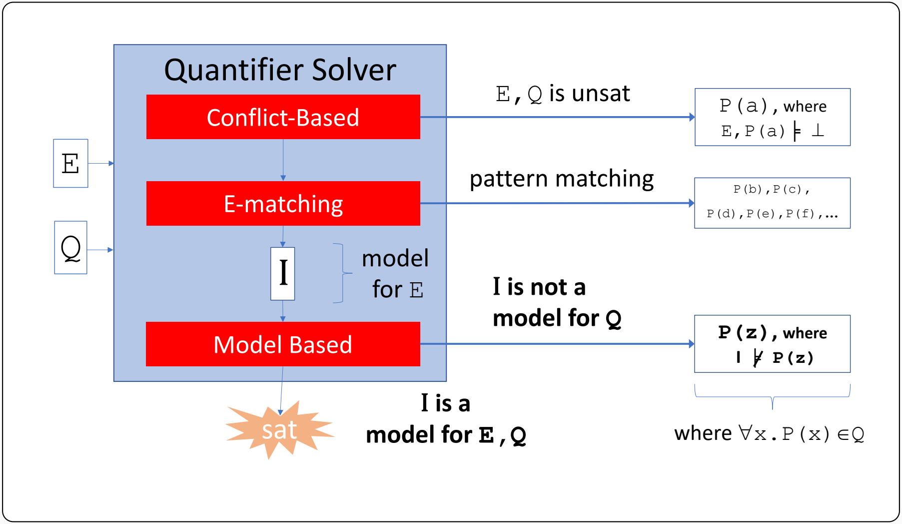
However, more sophisticated models are needed when other theories are involved:

$\forall x, y : \mathbf{Int}.\, (f(x,y) \geq x \wedge f(x,y) \geq y)$	$f^{\mathcal{I}} := \lambda x, y : \mathbf{Int}.\, \mathbf{ite}(x \geq y, x, y)$
$\forall x : \mathbf{Int}.\, 3 \cdot g(x) + 5 \cdot h(x) = x$	$g^{\mathcal{I}} := \lambda x.\, x - 3 \cdot x$ $h^{\mathcal{I}} := \lambda x.\, 2 \cdot x$
$\forall x, y : \mathbf{Int}.\, u(x+y) + 11 \cdot v(w(x)) = x$	??

More research is needed!

(may leverage recent advantages in syntax-guided synthesis?)

Putting It All Together in CVC4



General Challenge

Reasoning efficiently about quantifiers + EUF + other theories is still hard!

E-matching: Pattern selection, matching modulo theories

Conflict-based: Matching is incomplete, entailment tests are expensive

Model-based: Models are complex, interpreted domains may be infinite

General Challenge

Reasoning efficiently about quantifiers + EUF + other theories is still hard!

E-matching: Pattern selection, matching modulo theories

Conflict-based: Matching is incomplete, entailment tests are expensive

Model-based: Models are complex, interpreted domains may be infinite

Bright Spot

Reasoning efficiently about quantifiers + EUF + other theories is not as bad

- Classic QE algorithms are decision procedures for LRA [Ferrante&Rackoff 79, Loos&Wiespfenning 93], LIA [Cooper 72], datatypes [Maher 1988], ...
- Some have been leveraged successfully in SMT applications [Monniaux 2010, Bjorner 2012, Reynolds et al. 2015, Bjorner&Janota 2016, ...]

Bright Spot

Reasoning efficiently about quantifiers + EUF + other theories is not as bad

- Classic QE algorithms are decision procedures for LRA [Ferrante&Rackoff 79, Loos&Wiespfenning 93], LIA [Cooper 72], datatypes [Maher 1988], ...
- Some have been leveraged successfully in SMT applications [Monniaux 2010, Bjorner 2012, Reynolds et al. 2015, Bjorner&Janota 2016, ...]

Bright Spot

Reasoning efficiently about quantifiers + EUF + other theories is not as bad

- Classic QE algorithms are decision procedures for LRA [Ferrante&Rackoff 79, Loos&Wiespfenning 93], LIA [Cooper 72], datatypes [Maher 1988], ...
- Some have been leveraged successfully in SMT applications [Monniaux 2010, Bjorner 2012, Reynolds et al. 2015, Bjorner&Janota 2016, ...]

Counter-Example-Guided Quantifier Instantiation

Counterexample-Guided QI

Counterexample-Guided Quantifier Instantiation

Variants implemented in number of tools:

- Z3 [Bjorner 2012, Bjorner&Janota 2016]
- SPACER [Komuravelli et al. 2014][1]
- Yices [Dutertre 2015]
- CVC4 [CAV 2015, CAV 2018]
- UFO [Fedyukovich et al. 2016][2]
- Boolector [Preiner et al. 2017]

[1] Originally using Z3 as backend, now integrated in Z3
[2] Using Z3 as backend

Counterexample-Guided Instantiation

Basic idea: Derived from quantifier elimination (e.g., for LIA):

$$\exists x.\, \psi[x, \mathbf{y}] \equiv_T \psi[t_1, \mathbf{y}] \vee \cdots \vee \psi[t_n, \mathbf{y}] \text{ for some } t_1, \ldots, t_n$$

Counterexample-Guided Instantiation

Basic idea: Derived from quantifier elimination (e.g., for LIA):

$$\forall x.\, \neg\psi[x, \mathbf{y}] \equiv_T \neg\psi[t_1, \mathbf{y}] \wedge \cdots \wedge \neg\psi[t_n, \mathbf{y}] \quad \text{for some } t_1, \ldots, t_n$$

Counterexample-Guided Instantiation

Basic idea: Derived from quantifier elimination (e.g., for LIA):

$$\forall x.\, \neg\psi[x, \mathbf{y}] \equiv_T \neg\psi[t_1, \mathbf{y}] \wedge \cdots \wedge \neg\psi[t_n, \mathbf{y}] \text{ for some } t_1, \ldots, t_n$$

Enumerate instances via a counterexample-guided loop that is

1. terminating: generate a finite set $S \supseteq \{t_1, \ldots, t_n\}$
2. efficient in practice: typically terminates after $<< n$ instances

basic-CEGQI($\forall \mathbf{x}.\, \psi[\mathbf{x}, \mathbf{y}]$)

    $G := \emptyset$ (instances of $\forall \mathbf{x}.\, \psi$)

    repeat

        if G is T-unsatisfiable

            return **unsat** (because $\forall \mathbf{x}.\, \psi \models_T G$)

        else

            let $G' = G \cup \{\neg \psi\}$

            if is G' is T-unsatisfiable

                return **sat** (because $G \models_T \forall \mathbf{x}.\, \psi$)

            else

                let $\mathcal{I}$ be a T-model of G'

                let $\mathbf{t}[\mathbf{y}] = \mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G)$

                $G := G \cup \{\psi[\mathbf{t}, \mathbf{y}]\}$

basic-CEGQI($\forall \mathbf{x}.\, \psi[\mathbf{x}, \mathbf{y}]$)

$G := \emptyset$ (instances of $\forall \mathbf{x}.\, \psi$)

repeat

if G is T-unsatisfiable

return **unsat** (because $\forall \mathbf{x}.\, \psi \models_T G$)

else

let $G' = G \cup \{\neg\psi\}$

if is G' is T-unsatisfiable

return **sat** (because $G \models_T \forall \mathbf{x}.\, \psi$)

else

let $\mathcal{I}$ be a T-model of G'

let $\mathbf{t}[\mathbf{y}] = \mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G)$

$G := G \cup \{\psi[\mathbf{t}, \mathbf{y}]\}$

High-level View of Basic Procedure

basic-CEGQI($\forall \mathbf{x}.\, \psi[\mathbf{x}, \mathbf{y}]$)

$G := \emptyset$ (instances of $\forall \mathbf{x}.\, \psi$)

repeat

if G is T-unsatisfiable

return **unsat** (because $\forall \mathbf{x}.\, \psi \models_T G$)

else

let $G' = G \cup \{\neg\psi\}$

if is G' is T-unsatisfiable

return **sat** (because $G \models_T \forall \mathbf{x}.\, \psi$)

else

let $\mathcal{I}$ be a T-model of G'

let $\mathbf{t}[\mathbf{y}] = \mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G)$

$G := G \cup \{\psi[\mathbf{t}, \mathbf{y}]\}$

High-level View of Basic Procedure

basic-CEGQI($\forall \mathbf{x}.\, \psi[\mathbf{x}, \mathbf{y}]$)

$G := \emptyset$ (instances of $\forall \mathbf{x}.\, \psi$)

repeat

if G is T-unsatisfiable

return **unsat** (because $\forall \mathbf{x}.\, \psi \models_T G$)

else

let $G' = G \cup \{\neg \psi\}$

if is G' is T-unsatisfiable

return **sat** (because $G \models_T \forall \mathbf{x}.\, \psi$)

else

let $\mathcal{I}$ be a T-model of G'

let $\mathbf{t}[\mathbf{y}] = \mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G)$

$G := G \cup \{\psi[\mathbf{t}, \mathbf{y}]\}$

High-level View of Basic Procedure

basic-CEGQI($\forall \mathbf{x}.\, \psi[\mathbf{x}, \mathbf{y}]$)

$G := \emptyset$ (instances of $\forall \mathbf{x}.\, \psi$)

repeat

if G is T-unsatisfiable

return **unsat** (because $\forall \mathbf{x}.\, \psi \models_T G$)

else

let $G' = G \cup \{\neg\psi\}$

if is G' is T-unsatisfiable

return **sat** (because $G \models_T \forall \mathbf{x}.\, \psi$)

else

let $\mathcal{I}$ be a T-model of G'

let $\mathbf{t}[\mathbf{y}] = \mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G)$

$G := G \cup \{\psi[\mathbf{t}, \mathbf{y}]\}$

Relies on *selection function* Sel

Selection Function

Right selection functions make CEGQI a decision procedure for various theories T

Termination Requirements:

1. Quantifier-free fragment of T is decidable
2. For all qffs $\psi[\mathbf{x}, \mathbf{y}]$, selection function Sel is
   2.1 *finite*:
   there is a finite set $S_{\psi,\mathbf{x}}$ s.t. $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \in S_{\psi,\mathbf{x}}$ for all legal $\mathcal{I}, G$
   2.2 *monotonic*:
   if $G \models_T \psi[\mathbf{t}, \mathbf{y}]$ then $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \neq \mathbf{t}$ for all legal $\mathcal{I}, G$

Selection Function

Right selection functions make CEGQI a decision procedure for various theories T

Termination Requirements:

1. Quantifier-free fragment of T is decidable
2. For all qffs $\psi[\mathbf{x}, \mathbf{y}]$, selection function Sel is

   2.1 *finite*:
   there is a finite set $S_{\psi,\mathbf{x}}$ s.t. $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \in S_{\psi,\mathbf{x}}$ for all legal $\mathcal{I}, G$

   2.2 *monotonic*:
   if $G \models_T \psi[\mathbf{t}, \mathbf{y}]$ then $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \neq \mathbf{t}$ for all legal $\mathcal{I}, G$

Selection Function

Right selection functions make CEGQI a decision procedure for various theories T

Termination Requirements:

1. Quantifier-free fragment of T is decidable
2. For all qffs $\psi[\mathbf{x}, \mathbf{y}]$, selection function Sel is
   2.1 *finite*:
   there is a finite set $S_{\psi,\mathbf{x}}$ s.t. $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \in S_{\psi,\mathbf{x}}$ for all legal $\mathcal{I}, G$
   2.2 *monotonic*:
   if $G \models_T \psi[\mathbf{t}, \mathbf{y}]$ then $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \neq \mathbf{t}$ for all legal $\mathcal{I}, G$

Right selection functions make CEGQI a decision procedure for various theories T

Termination Requirements:

1. Quantifier-free fragment of T is decidable
2. For all qffs $\psi[\mathbf{x}, \mathbf{y}]$, selection function Sel is
   2.1 *finite*:
   there is a finite set $S_{\psi,\mathbf{x}}$ s.t. $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \in S_{\psi,\mathbf{x}}$ for all legal $\mathcal{I}, G$
   2.2 *monotonic*:
   if $G \models_T \psi[\mathbf{t}, \mathbf{y}]$ then $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \neq \mathbf{t}$ for all legal $\mathcal{I}, G$

Right selection functions make CEGQI a decision procedure for various theories T

Termination Requirements:

1. Quantifier-free fragment of T is decidable

2. For all qffs $\psi[\mathbf{x}, \mathbf{y}]$, selection function Sel is

   2.1 *finite*:
   there is
   $\mathcal{I}, G$

   2.2 *monotonic*:
   if $G \models_T \psi[\mathbf{t}, \mathbf{y}]$ then $\mathrm{Sel}(\mathbf{x}, \psi, \mathcal{I}, G) \neq \mathbf{t}$ for all legal $\mathcal{I}, G$

Theorem. Under (1), procedure **basic-CEGQI** always terminates if *sel* is finite and monotonic

From CEGQI ...

basic-CEGQI($\forall x.\, \psi[x, y]$)

$G := \emptyset$

repeat

if G is T-unsatisfiable

return **unsat**

else

let $G' = G \cup \{\neg\psi\}$

if is G' is T-unsatisfiable

return **sat**

else

let $\mathcal{I}$ be a T-model of G'

let $t[y] = \mathrm{Sel}(x, \psi, \mathcal{I}, G)$

$G := G \cup \{\psi[t, y]\}$

From CEGQI to Quantifier Elimination

project($x, \psi[x, y]$)
  $G := \emptyset$
  repeat
    if G is T-unsatisfiable
      return $\perp$
    else
      let $G' = G \cup \{\neg\psi\}$
      if is G' is T-unsatisfiable
        return $\bigwedge G$
      else
        let $\mathcal{I}$ be a T-model of G'
        let $t[y] = \mathrm{Sel}(x, \psi, \mathcal{I}, G)$
        $G := G \cup \{\psi[t, y]\}$

From CEGQI to Quantifier Elimination

$\textbf{project}(x, \psi[x, y])$

$G := \emptyset$

repeat

if G is T-unsatisfiable

return $\bot$

else

let $G' = G \cup \{\neg\psi\}$

if is G' is T-unsatisfiable

return $\bigwedge G$

else

let $\mathcal{I}$ be a T-model of G'

let $\boldsymbol{t}[y] = \mathrm{Sel}(x, \psi, \mathcal{I}, G)$

$G := G \cup \{\psi[t, y]\}$

Note:

Let $\varphi[y] = \textbf{project}(x, \psi[x, y])$

Then $\varphi \equiv_T \forall \boldsymbol{x}.\, \psi$

From CEGQI to QE: General Case

Assumption: Consider only NNF formulas φ containing a subformula $\forall \mathbf{x}.\, \varphi_1 \vee \varphi_2$ (resp. $\exists \mathbf{x}.\, \varphi_1 \wedge \varphi_2$) only if $\varphi_1 \vee \varphi_2$ (resp. $\varphi_1 \wedge \varphi_2$) is quantifier-free[3]

[3] For simplicity and wlog by (lazy) PNF transformation

From CEGQI to QE: General Case

$$
\begin{aligned}
\mathsf{qe}(x, \varphi) \quad &:= \quad \text{if } \varphi \text{ is quantifier-free then} \\
&\qquad\quad \mathsf{project}(x, \varphi) \\
&\qquad \text{else} \\
&\qquad\quad \text{match } \varphi \text{ with} \\
&\qquad\qquad\qquad \varphi_1 \wedge \varphi_2 : \quad \mathsf{qe}(x, \varphi_1) \wedge \mathsf{qe}(x, \varphi_2) \\
&\qquad\qquad\qquad \exists z.\, \psi : \quad \neg\, \mathsf{qe}(x, \forall z.\, \mathsf{nnf}(\neg\psi)) \\
&\qquad\qquad\qquad \forall z.\, \psi : \quad \mathsf{qe}(x, \mathsf{qe}(z, \psi)) \\
\mathsf{nnf}(\varphi) \quad &:= \quad \text{negation normal form of } \varphi
\end{aligned}
$$

From CEGQI to QE: General Case

$\mathbf{qe}(\boldsymbol{x}, \varphi)$:= if φ is quantifier-free then

$\quad \mathbf{project}(\boldsymbol{x}, \varphi)$

else

$\quad$ match φ with

Note:

1. Avoiding full prenex normal form transformation increases scalability in practice
2. Implementation of general CEBQI in CVC4 is similar in spirit to **qe** but is fully integrated into SMT loop [FMSD 2017]

Selection Functions

Linear real arithmetic (LRA) [FMSD 2017]

- Maximal lower (minimal upper) bounds [Loos+Wiespfenning 1993]

$$l_1 < k, \ldots, l_n < k \quad \Longrightarrow \quad \{x \mapsto l_{\max} + d\}$$

  (may involve virtual terms δ, ∞)

- Interior point method [Ferrante&Rackoff 1979]

$$l_{\max} < k < u_{\min} \quad \Longrightarrow \quad \{x \mapsto (l_{\max} + u_{\min})/2\}$$

Linear integer arithmetic (LIA) [FMSD 2017]

- Maximal lower (minimal upper) bounds [Cooper 1972]

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

Selection Functions

Linear real arithmetic (LRA) [FMSD 2017]

- Maximal lower (minimal upper) bounds [Loos+Wiespfenning 1993]

$$l_1 < k, \ldots, l_n < k \quad \Longrightarrow \quad \{x \mapsto l_{\max} + d\}$$

  (may involve virtual terms δ, ∞)

- Interior point method [Ferrante&Rackoff 1979]

$$l_{\max} < k < u_{\min} \quad \Longrightarrow \quad \{x \mapsto (l_{\max} + u_{\min})/2\}$$

Linear integer arithmetic (LIA) [FMSD 2017]

- Maximal lower (minimal upper) bounds [Cooper 1972]

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

Selection Functions

Linear real arithmetic (LRA) [FMSD 2017]

- Maximal lower (minimal upper) bounds [Loos+Wiespfenning 1993]

$$l_1 < k, \ldots, l_n < k \quad \Longrightarrow \quad \{x \mapsto l_{\max} + d\}$$

(may involve virtual terms δ, ∞)

- In

Common termination argument:
a finite number of instances cover all cases

Linear integer arithmetic (LIA) [FMSD 2017]

- Maximal lower (minimal upper) bounds [Cooper 1972]

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

Selection Functions

Finite domains

- Model-based value instantiations [Wintersteiger et al. 2013]

$$D = \{d_1, \ldots, d_n\} \quad \Longrightarrow \quad \{x \mapsto d_i\}$$

Fixed-size Bit vectors

- Value instantiations [Neimetz et al. 2016]

$$0 \leq i < w \quad \Longrightarrow \quad \{x \mapsto 2^i\}$$

- Invertibility conditions [CAV 2018]

(next slides)

Datatypes

- Stay tuned ...

Selection Functions

Finite domains

- Model-based value instantiations [Wintersteiger et al. 2013]

$$D = \{d_1, \ldots, d_n\} \quad \Longrightarrow \quad \{x \mapsto d_i\}$$

Fixed-size Bit vectors

- Value instantiations [Neimetz et al. 2016]

$$0 \leq i < w \quad \Longrightarrow \quad \{x \mapsto 2^i\}$$

- Invertibility conditions [CAV 2018]

(next slides)

Datatypes

- Stay tuned ...

Selection Functions

Finite domains

- Model-based value instantiations [Wintersteiger et al. 2013]

$$D = \{d_1, \ldots, d_n\} \quad \Longrightarrow \quad \{x \mapsto d_i\}$$

Fixed-size Bit vectors

- Value instantiations [Neimetz et al. 2016]

$$0 \leq i < w \quad \Longrightarrow \quad \{x \mapsto 2^i\}$$

- Invertibility conditions [CAV 2018]

(next slides)

Datatypes

- Stay tuned ...

Outline

Motivation

Example: Prove unsatisfiability of

$$\psi = \forall x.\, x + s \not\approx t$$

with x, s, t bit vectors of size n

It is crucial to find good set of instantiation candidates for x

Motivation

Example: Prove unsatisfiability of

$$\psi = \forall x.\, x + s \not\approx t$$

with x, s, t bit vectors of size n

Naive approach: Enumerate 2^n possible values for x

Example: Prove unsatisfiability of

$$\psi = \forall x.\, x + s \not\approx t$$

with x, s, t bit vectors of size n

Better approach:

1. Try to solve $\neg(x + s \not\approx t)$ for x (yielding $x = t - s$)
2. Instantiate ψ with computed symbolic solution

$$\underbrace{\overbrace{t - s}^{x} + s \not\approx t}_{\text{UNSAT}}$$

Quantifier Instantiation for Bit Vectors

Idea: Compute symbolic solutions of bit vector constraints

Quantifier Instantiation for Bit Vectors

Idea: Compute symbolic solutions of bit vector constraints

Problem: hard or impossible in general

Quantifier Instantiation for Bit Vectors

Idea: Compute symbolic solutions of bit vector constraints

Problem: hard or impossible in general

- **Example:** $2 \cdot x \approx 3$ is unsolvable

Quantifier Instantiation for Bit Vectors

Idea: Compute symbolic solutions of bit vector constraints

Problem: hard or impossible in general

Our Answer:

1. Consider restricted case where φ has the form

   $$x \diamond s \bowtie t \quad \text{or} \quad s \diamond x \bowtie t$$

   with $\bowtie$ relational operator and x not in s or t

2. Consider *conditional* symbolic solutions

   (e.g., identify conditions under which $s \cdot x \approx t$ is solvable)

Invertibility Condition

Exact condition under which a bit vector operation is solvable for some x

Example: $x \cdot s \approx t$

- Invertibility condition: $(-s \mid s) \mathbin{\&} t \approx t$
- $(-s \mid s) \mathbin{\&} t \approx t \;\equiv_{\mathrm{BV}}\; \exists x.\, x \cdot s \approx t$

Invertibility Conditions

- 162 IC's for: $\{\approx, \not\approx, <_u, \leq_u, >_u, \geq_u, <_s, \leq_s, >_s, \geq_s\} \;\times$ $\{\sim, \&, |, \ll, \gg, \gg_a, -, +, \cdot, \text{mod}, \div, \circ, [:]\}$
- 83 crafted manually
- 79 generated automatically with syntax-guided synthesizer

Invertibility Condition

Exact condition under which a bit vector operation is solvable for some x

Example: $x \cdot s \approx t$

- Invertibility condition: $(-s \mid s)\ \&\ t \approx t$
- $(-s \mid s)\ \&\ t \approx t \ \equiv_{\mathrm{BV}}\ \exists x.\, x \cdot s \approx t$

Invertibility Conditions

- 162 IC's for: $\{\approx, \not\approx, <_u, \leq_u, >_u, \geq_u, <_s, \leq_s, >_s, \geq_s\} \times \{\sim, \&, |, \ll, \gg, \gg_a, -, +, \cdot, \mathrm{mod}, \div, \circ, [:]\}$
- 83 crafted manually
- 79 generated automatically with syntax-guided synthesizer

Invertibility Condition

Exact condition under which a bit vector operation is solvable for some x

Example: $x \cdot s \approx t$

- Invertibility condition: $(-s \mid s) \mathbin{\&} t \approx t$
- $(-s \mid s) \mathbin{\&} t \approx t \;\; \equiv_{\text{BV}} \;\; \exists x.\, x \cdot s \approx t$

$x \cdot s = t$ is solvable for x

iff

s has fewer trailing zeroes than t

Invertibility

- 162 IC's
- 83 craft
- 79 generated automatically with syntax-guided synthesizer

Invertibility Condition

Exact condition under which a bit vector operation is solvable for some x

Example: $x \cdot s \approx t$

- Invertibility condition: $(-s \mid s) \,\&\, t \approx t$
- $(-s \mid s) \,\&\, t \approx t \;\equiv_{\mathrm{BV}}\; \exists x.\, x \cdot s \approx t$

Invertibility Conditions

- 162 IC's for: $\{\approx, \not\approx, <_u, \leq_u, >_u, \geq_u, <_s, \leq_s, >_s, \geq_s\} \;\times$ $\{\sim, \&, |, \ll, \gg, \gg_a, -, +, \cdot, \mathrm{mod}, \div, \circ, [:]\}$
- 83 crafted manually
- 79 generated automatically with syntax-guided synthesizer

A Few Invertibility Conditions

$\ell[x]$	$\approx$	$\not\approx$
$x \cdot s \bowtie t$	$(-s \mid s)\ \&\ t \approx t$	$s \not\approx 0 \lor t \not\approx 0$
$x \bmod s \bowtie t$	$\sim(-s) \geq_u t$	$s \not\approx 1 \lor t \not\approx 0$
$s \bmod x \bowtie t$	$(t + t - s)\ \&\ s \geq_u t$	$s \not\approx 0 \lor t \not\approx 0$
$x \div s \bowtie t$	$(s \cdot t) \div s \approx t$	$s \not\approx 0 \lor t \not\approx \sim 0$
$s \div x \bowtie t$	$s \div (s \div t) \approx t$	$\begin{cases} s\ \&\ t \approx 0 & \text{for } \kappa(s) = 1 \\ \top & \text{otherwise} \end{cases}$
$x\ \&\ s \bowtie t$	$t\ \&\ s \approx t$	$s \not\approx 0 \lor t \not\approx 0$
$x \mid s \bowtie t$	$t \mid s \approx t$	$s \not\approx \sim 0 \lor t \not\approx \sim 0$
$x \gg s \bowtie t$	$(t \ll s) \gg s \approx t$	$t \not\approx 0 \lor s <_u \kappa(s)$
$s \gg x \bowtie t$	$\bigvee_{i=0}^{\kappa(s)} s \gg i \approx t$	$s \not\approx 0 \lor t \not\approx 0$
$x \gg_a s \bowtie t$	$(s <_u \kappa(s) \Rightarrow (t \ll s) \gg_a s \approx t) \land$ $(s \geq_u \kappa(s) \Rightarrow (t \approx \sim 0 \lor t \approx 0))$	$\top$
$s \gg_a x \bowtie t$	$\bigvee_{i=0}^{\kappa(s)} s \gg_a i \approx t$	$(t \not\approx 0 \lor s \not\approx 0) \land$ $(t \not\approx \sim 0 \lor s \not\approx \sim 0)$

A Few More Invertibility Conditions

$\ell[x]$	$<_s$	$>_s$
$x \cdot s \bowtie t$	$\sim(-t)\ \&\ (-s \mid s) <_s t$	$t <_s t - ((s \mid t) \mid -s)$
$x \bmod s \bowtie t$	$\sim t <_s (-s \mid -t)$	$(s >_s 0 \Rightarrow t <_s \sim(-s)) \wedge$ $(s \leq_s 0 \Rightarrow t \not\approx \max_s) \wedge$ $(t \not\approx 0 \vee s \not\approx 1)$
$s \bmod x \bowtie t$	$s <_s t \vee 0 <_s t$	$(s \geq_s 0 \Rightarrow s >_s t) \wedge$ $(s <_s 0 \Rightarrow ((s-1) \gg 1) >_s t)$
$x \div s \bowtie t$	$t \leq_s 0 \Rightarrow \min_s \div s <_s t$	$\sim 0 \div s >_s t \vee \max_s \div s >_s t$
$s \div x \bowtie t$	$s <_s t \vee t \geq_s 0$	$\begin{cases} s >_s t & \text{for } \kappa(s) = 1 \\ (s \geq_s 0 \Rightarrow s >_s t) \wedge & \text{otherwise} \\ (s <_s 0 \Rightarrow s \gg 1 >_s t) & \end{cases}$
$x\ \&\ s \bowtie t$	$\sim(-t)\ \&\ s <_s t$	$t <_s s\ \&\ \max_s$
$x \mid s \bowtie t$ $s \mid x \bowtie t$	$\sim(s-t) \mid s <_s t$	$t <_s (s \mid \max_s)$
$x \gg s \bowtie t$	$\sim(-t) \gg s <_s t$	$t <_s (\max_s \ll s) \gg s$
$s \gg x \bowtie t$	$s <_s t \vee 0 <_s t$	$(s <_s 0 \Rightarrow s \gg 1 >_s t) \wedge$ $(s \geq_s 0 \Rightarrow s >_s t)$

From Invertibility Conditions to Symbolic Instantiations

Hilbert choice functions $\varepsilon x.\, \varphi$

- Represents a solution for φ if there is one
- Represents arbitrary value, otherwise

Embed invertibility conditions into choice functions

BV literal:	$l[x] = x \diamond s \bowtie t$
Inv. condition:	IC_x
Symbolic solution:	$\varepsilon y.\,(IC_x \Rightarrow l[y])$

From Invertibility Conditions to Symbolic Instantiations

Hilbert choice functions $\varepsilon x.\, \varphi$

- Represents a solution for φ if there is one
- Represents arbitrary value, otherwise

Embed invertibility conditions into choice functions

BV literal:	$l[x] = x \diamond s \bowtie t$
Inv. condition:	IC_x
Symbolic solution:	$\varepsilon y.\, (IC_x \;\Rightarrow\; l[y])$

From Invertibility Conditions to Symbolic Instantiations

Hilbert choice functions $\varepsilon x.\, \varphi$

- Represents a solution for φ if there is one
- Represents arbitrary value, otherwise

Embed invertibility conditions into choice functions

BV literal:	$l[x] = x \diamond s \bowtie t$
Inv. condition:	IC_x
Symbolic solution:	$\varepsilon y.\, (IC_x \;\Rightarrow\; l[y])$

Note 1: Choice function expresses all conditional solutions with a single term

From Invertibility Conditions to Symbolic Instantiations

Hilbert choice functions $\varepsilon x.\, \varphi$

- Represents a solution for φ if there is one
- Represents arbitrary value, otherwise

Embed invertibility conditions into choice functions

BV literal: $l[x] = x \diamond s \bowtie t$

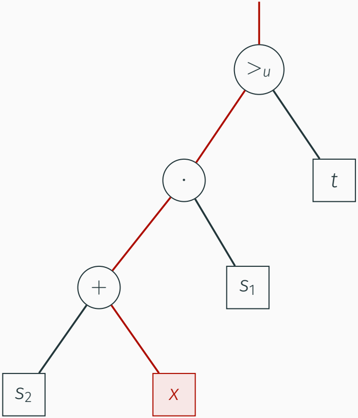
Inv. condition: IC_x

Symbolic solution: $\varepsilon y.\, (IC_x \;\Rightarrow\; l[y])$

Note 2: The ε binder can be later eliminated from instances by Skolemization:

$$\varphi[\varepsilon y.\, (IC_x \;\Rightarrow\; l[y])] \;\longrightarrow\; \varphi[k] \land (IC_x \;\Rightarrow\; l[k])$$

More General Case by Example: $\forall x.\, (s_2 + x) \cdot s_1 \leq_u t$



1. Pick variable to solve for (x)
2. Compute inverse/IC's along path to x
3. Solve $z \cdot s_1 >_u t$ for z

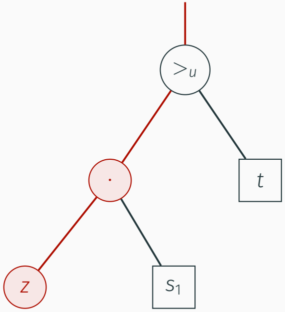
$$\begin{aligned} IC_z &= t <_u -s \mid s \\ z &= \varepsilon y.\, IC_z \Rightarrow y \cdot s_1 >_u t \end{aligned}$$

4. Solve $s_2 + x \approx z$ for x

$$\begin{aligned} IC_x &= \top \\ x &= z - s_2 \end{aligned}$$

Instantiation for x: $\varepsilon y.\, (t <_u -s \mid s \Rightarrow s_1 \cdot y >_u t) - s_2$

More General Case by Example: $\forall x.\, (s_2 + x) \cdot s_1 \leq_u t$



1. Pick variable to solve for (x)
2. Compute inverse/IC's along path to x
3. Solve $z \cdot s_1 >_u t$ for z

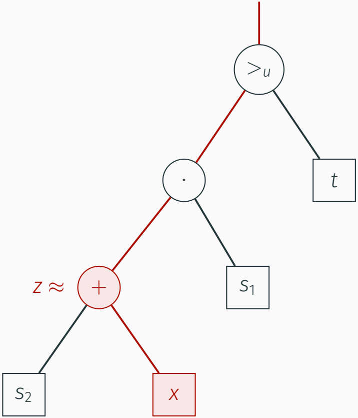
$$
\begin{aligned}
IC_z &= t <_u -s \mid s \\
z &= \varepsilon y.\, IC_z \Rightarrow y \cdot s_1 >_u t
\end{aligned}
$$

4. Solve $s_2 + x \approx z$ for x

$$
\begin{aligned}
IC_x &= \top \\
x &= z - s_2
\end{aligned}
$$

Instantiation for x: $\varepsilon y.\, (t <_u -s \mid s \Rightarrow s_1 \cdot y >_u t) - s_2$

More General Case by Example: $\forall x. (s_2 + x) \cdot s_1 \leq_u t$



1. Pick variable to solve for (x)
2. Compute inverse/IC's along path to x
3. Solve $z \cdot s_1 >_u t$ for z

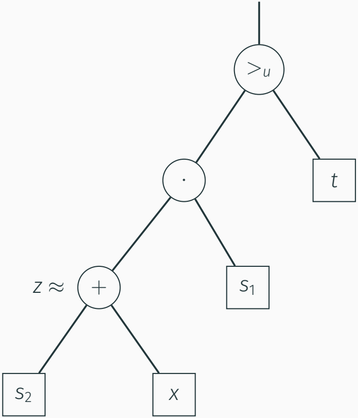
$$\begin{aligned} IC_z &= t <_u -s \mid s \\ z &= \varepsilon y.\, IC_z \Rightarrow y \cdot s_1 >_u t \end{aligned}$$

4. Solve $s_2 + x \approx z$ for x

$$\begin{aligned} IC_x &= \top \\ x &= z - s_2 \end{aligned}$$

Instantiation for x: $\varepsilon y. (t <_u -s \mid s \Rightarrow s_1 \cdot y >_u t) - s_2$

More General Case by Example: $\forall x. (s_2 + x) \cdot s_1 \leq_u t$



1. Pick variable to solve for (x)
2. Compute inverse/IC's along path to x
3. Solve $z \cdot s_1 >_u t$ for z

$$\begin{aligned} IC_z &= t <_u -s \mid s \\ z &= \varepsilon y.\, IC_z \Rightarrow y \cdot s_1 >_u t \end{aligned}$$

4. Solve $s_2 + x \approx z$ for x

$$\begin{aligned} IC_x &= \top \\ x &= z - s_2 \end{aligned}$$

Instantiation for x: $\varepsilon y. (t <_u -s \mid s \Rightarrow s_1 \cdot y >_u t) - s_2$

Multiple Variable Occurrences

Non-linear constraints (multiple occurrences of a variable)

- Try to linearize with rewriting/normalization
  e.g., $x + x + s \approx t \longrightarrow 2 \cdot x + s \approx t$
- Otherwise, replace extra occurrences of x with value in current model $\mathcal{I}$
  e.g., $x \cdot x + s \approx t \longrightarrow x \cdot x^{\mathcal{I}} + s \approx t$

► **Future work:** Use SyGuS to synthesize IC's for non-linear cases

Multiple Variable Occurrences

Non-linear constraints (multiple occurrences of a variable)

- Try to linearize with rewriting/normalization
  e.g., $x + x + s \approx t \longrightarrow 2 \cdot x + s \approx t$
- Otherwise, replace extra occurrences of x with value in current model $\mathcal{I}$
  e.g., $x \cdot x + s \approx t \longrightarrow x \cdot x^{\mathcal{I}} + s \approx t$

► **Future work:** Use SyGuS to synthesize IC's for non-linear cases

Multiple Variable Occurrences

Non-linear constraints (multiple occurrences of a variable)

- Try to linearize with rewriting/normalization
  e.g., $x + x + s \approx t \;\longrightarrow\; 2 \cdot x + s \approx t$
- Otherwise, replace extra occurrences of x with value in current model $\mathcal{I}$
  e.g., $x \cdot x + s \approx t \;\longrightarrow\; x \cdot x^{\mathcal{I}} + s \approx t$

► **Future work:** Use SyGuS to synthesize IC's for non-linear cases

Experimental Results

	$\text{CVC4}_{\text{base}}$	Q3B	Boolector	Z3	CVC4_{ic}
keymaera (4035)	3823	3805	4025	**4031**	3993
psyco (194)	**194**	99	193	193	190
scholl (374)	239	214	**289**	271	246
tptp (73)	**73**	**73**	72	**73**	**73**
uauto (284)	112	256	180	190	**274**
wintersteiger (191)	168	**184**	154	162	168
Total (5151)	4609	4631	4913	4920	**4944**

Limits: 300 seconds CPU time limit, 100G memory limit

CVC4_{ic} won division BV at SMT-COMP 2018

Outline

Invertibility Conditions for Floating Point Arithmetic

Similar approach can be applied to Floating Points [CAV 2019]

However, invertibility conditions are much more complex

Invertibility Conditions for Floating Point Arithmetic

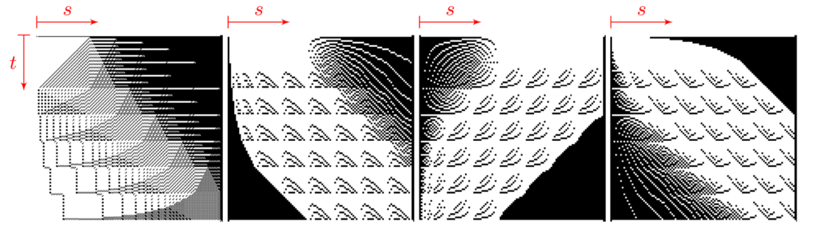
Similar approach can be applied to Floating Points [CAV 2019]

However, invertibility conditions are much more complex (found 167/188 IC's so far)

Invertibility Conditions for Floating Point Arithmetic

Similar approach can be applied to Floating Points [CAV 2019]

However, invertibility conditions are much more complex



(a) $x + s \approx t$ (b) $x \cdot s \approx t$ (c) $s \div x \approx t$ (d) $x \div s \approx t$

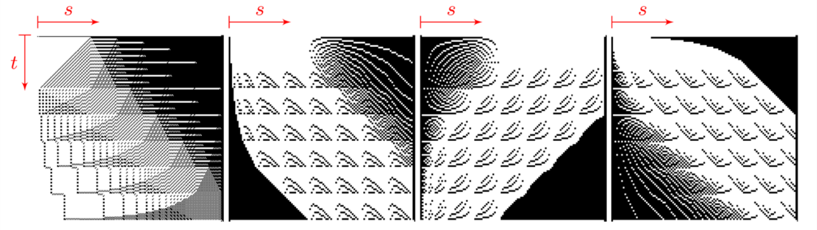
(Shown for FP[3,5])

(white dot = IC is **sat**, black dot = IC is **unsat**)

Invertibility Conditions for Floating Point Arithmetic

Similar approach can be applied to Floating Points [CAV 2019]

However, invertibility conditions are much more complex



(a) $x + s \approx t$ (b) $x \cdot s \approx t$ (c) $s \div x \approx t$ (d) $x \div s \approx t$

(a) $(t \overset{RTN}{-} s) \overset{R}{+} s \approx t \ \vee \ t \approx s \ \vee \ (t \overset{RTP}{-} s) \overset{R}{+} s \approx t$

(b) $t \approx (t \overset{RTP}{\div} s) \overset{R}{\cdot} s \vee t \approx (t \overset{RTN}{\div} s) \overset{R}{\cdot} s \vee (s \approx \pm\infty \wedge t \approx \pm\infty) \vee (s \approx \pm 0 \wedge t \approx \pm 0)$

...

Invertibility Conditions for Floating Point Arithmetic

$$\exists x.\, x \overset{R}{+} s \approx t$$

$$\equiv_{\mathrm{BV}}$$

$$(t \overset{RTN}{-} s) \overset{R}{+} s \approx t \quad \vee \quad t \approx s \quad \vee \quad (t \overset{RTP}{-} s) \overset{R}{+} s \approx t$$

rounding towards negative	corner case (zero)	rounding towards positive
$x = t \overset{RTP}{-} s$	$x = \pm 0$	$x = t \overset{RTN}{-} s$

Invertibility Conditions for Floating Point Arithmetic

$$\exists x.\, x \overset{R}{+} s \approx t$$

$$\equiv_{\mathrm{BV}}$$

$$(t \overset{RTN}{-} s) \overset{R}{+} s \approx t \quad \vee \quad t \approx s \quad \vee \quad (t \overset{RTP}{-} s) \overset{R}{+} s \approx t$$

rounding towards negative	corner case (zero)	rounding towards positive
$x = t \overset{RTP}{-} s$	$x = \pm 0$	$x = t \overset{RTN}{-} s$

Conclusion

Conclusion

- SMT solvers do not operate just on ground formulas
- There has been considerable progress on reasoning with quantified formulas in SMT
- Still, a lot more needs to be done
- The quest of combining theory and quantifier reasoning efficiently is still on
- The CVC4 team is at the forefront of this quest
- CVC4 is available at `http://cvc4.cs.stanford.edu/`
- Join our quest!
- We are hiring PhD students and postdocs and welcome collaborations with other groups

Related and Future Work

- Symmetry elimination
- Proofs
- Synthesis
- Abduction

Thank you