

Challenges for Fast Synthesis Procedures in SMT

Andrew Reynolds

ARCADE Workshop

August 6, 2017



THE UNIVERSITY
OF IOWA

Synthesis

- SMT solvers act as *subroutines* for automated synthesis
 - For program snippets, planning, digital circuits, programming by examples, ...
- More recently, SMT solvers act as *stand-alone tools* for synthesis
 - Leveraging their support for first-order quantification

[Reynolds et al CAV2015]



Synthesis Conjectures

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

There exists a function f for which property P holds for all x

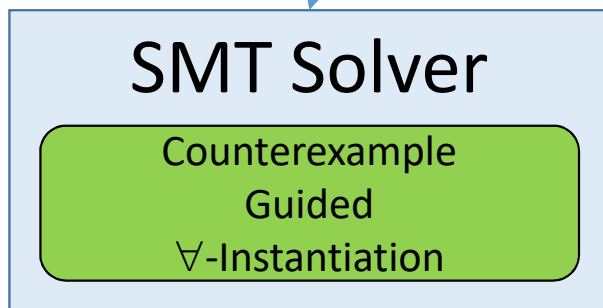
Refutation-Based Synthesis in SMT

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

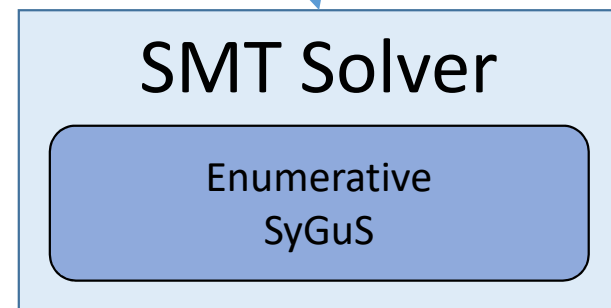
(**negated** synthesis conjecture)

Refutation-Based Synthesis in SMT

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



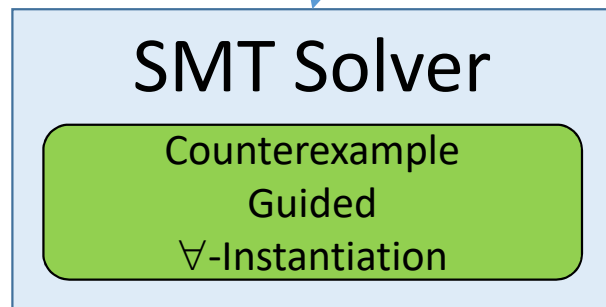
or



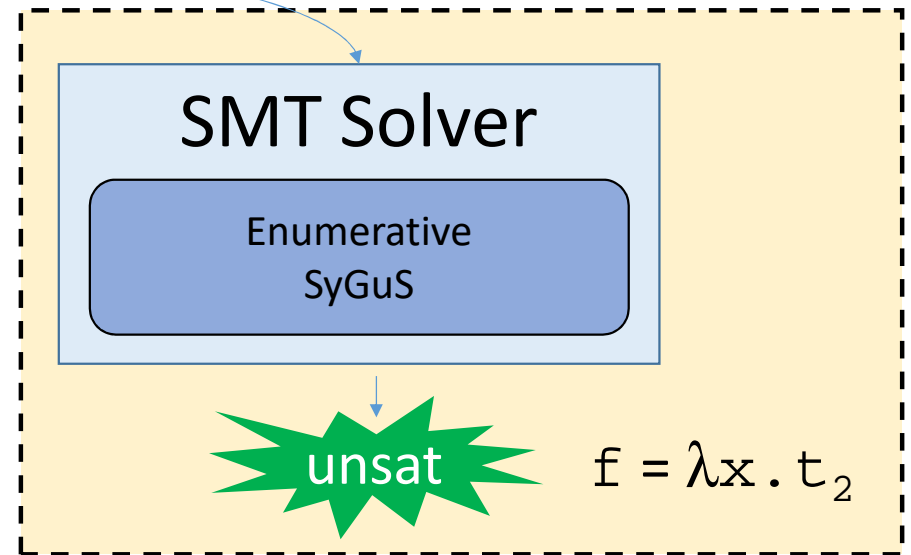
- Two approaches for refutation-based synthesis in SMT solvers [Reynolds et al CAV2015]

Refutation-Based Synthesis in SMT

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



or



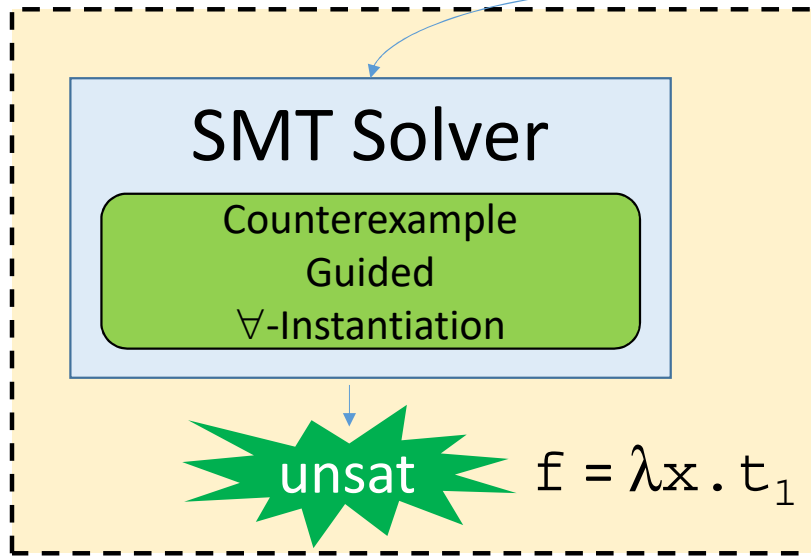
unsat $f = \lambda x . t_1$

unsat $f = \lambda x . t_2$

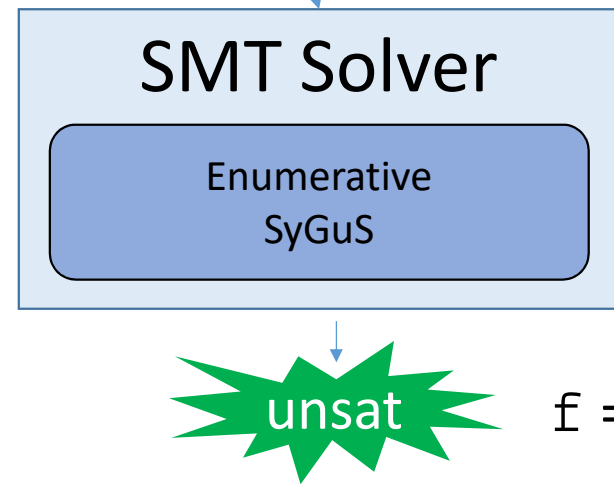
\Rightarrow Based on enumerative search (via syntax-guided synthesis) [Alur et al 2013]

Refutation-Based Synthesis in SMT

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



or



⇒ Based on first-order quantifier instantiation (**focus of this talk**)

Single Invocation Conjectures

- Some synthesis conjectures are *essentially first-order*:

$$\neg \exists f . \forall x y . \mathbf{f}(x, y) \geq x \wedge \mathbf{f}(x, y) \geq y \wedge (\mathbf{f}(x, y) = x \vee \mathbf{f}(x, y) = y)$$

“ $\mathbf{f}(x, y)$ is the maximum of x and y ”

Single Invocation Conjectures

$$\neg \exists f . \forall xy . \underline{f(x,y)} \geq x \wedge \underline{f(x,y)} \geq y \wedge (\underline{f(x,y)} = x \vee \underline{f(x,y)} = y)$$

Int × Int → Int

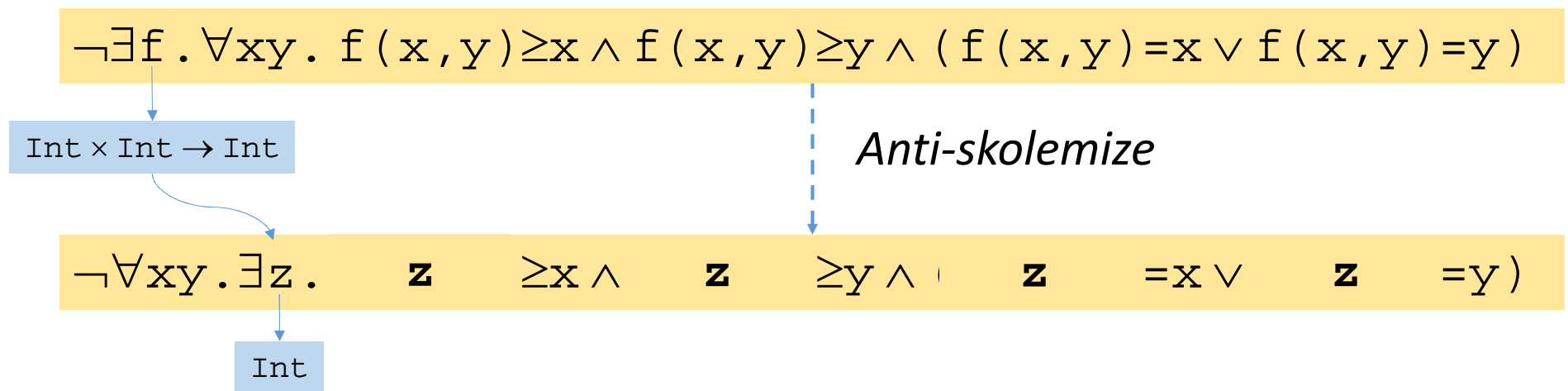
All occurrence of f are in terms of the form $f(x,y)$
∅ “single invocation” synthesis conjectures

Single Invocation Conjectures

$$\neg \exists f. \forall xy. f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$$

$\text{Int} \times \text{Int} \rightarrow \text{Int}$

Single Invocation Conjectures



Single Invocation Conjectures

$\neg \exists f. \forall x y. f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$

Int × Int → Int

$\neg \forall x y. \exists z. z \geq x \wedge z \geq y \wedge (z = x \vee z = y)$

Int

“for each x, y , there exists a return value z that is the maximum of x and y ”

Single Invocation Conjectures

$\neg \exists f. \forall xy. f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$

Int × Int → Int

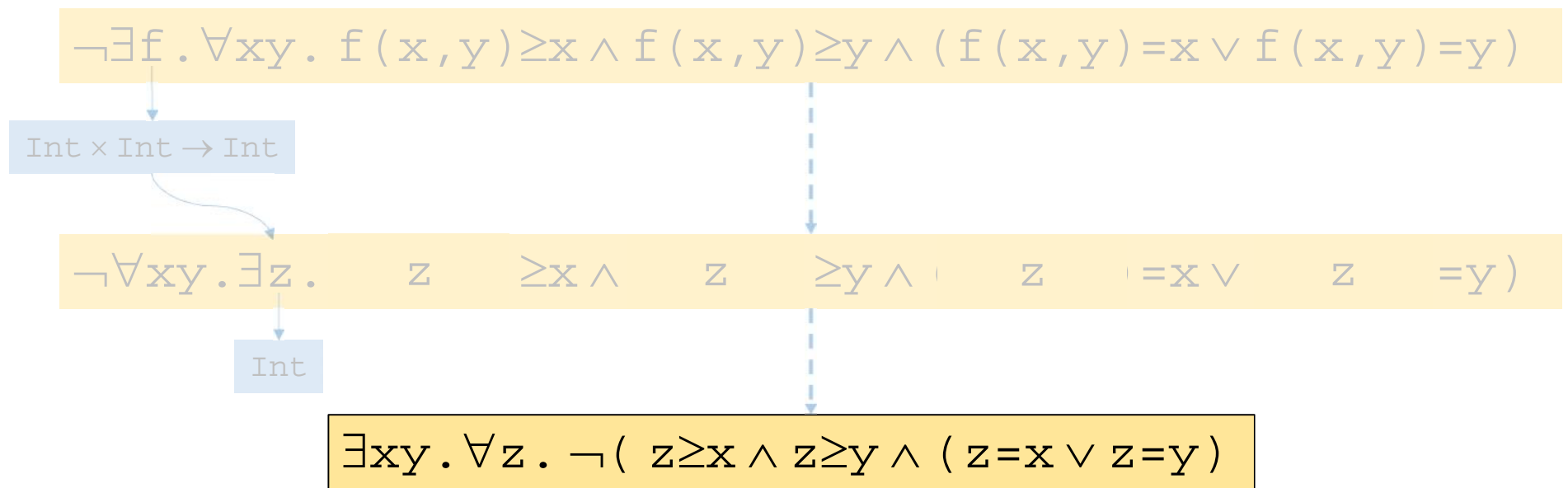
$\neg \forall xy. \exists z. z \geq x \wedge z \geq y \wedge (z = x \vee z = y)$

Int

Simplify

$\exists xy. \forall z. \neg (z \geq x \wedge z \geq y \wedge (z = x \vee z = y))$

Single Invocation Conjectures



First-order linear arithmetic \emptyset Solvable by first-order \exists -instantiation
 [Reynolds et al CAV2015]

Single Invocation Synthesis in SMT

$\neg \exists f. \forall xy. f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$

SAT Solver

LIA
 \forall -instantiation

Single Invocation Synthesis in SMT

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

SAT Solver

LIA
 \forall -instantiation

Single Invocation Synthesis in SMT

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

Translate to first-order

$\forall z. \neg \text{isMax}(z, x, y)$

SAT Solver

LIA

\forall -instantiation

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(\mathbf{x}, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(\mathbf{y}, x, y)$

SAT Solver

Instantiate $z \rightarrow \mathbf{x}, z \rightarrow \mathbf{y}$

LIA
 \forall -instantiation

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow x < y$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow y < x$

SAT Solver

Simplify

LIA
 \forall -instantiation

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow x < y$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow y < x \dots$

SAT Solver

LIA

\forall -instantiation

unsat

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow x < y$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow y < x$

SAT Solver

LIA

\forall -instantiation

unsat

\Rightarrow Solution for f can be constructed from unsatisfiable core of instantiations

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(x, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(y, x, y)$

SAT Solver

LIA

\forall -instantiation

unsat

$\lambda xy. ?$

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(x, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(y, x, y)$

SAT Solver

LIA
 \forall -instantiation

unsat

$\lambda xy. \text{ite}(\text{isMax}(x, x, y), x, ?)$

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(x, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(y, x, y)$

SAT Solver

LIA
 \forall -instantiation

unsat

$\lambda xy. \text{ite}(\text{isMax}(x, x, y), x, y)$

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(x, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(y, x, y)$

SAT Solver

LIA

\forall -instantiation

unsat

$\lambda xy. \text{ite}((x \geq x \wedge x \geq y \wedge (x = x \vee x = y)), x, y)$

\Rightarrow Expand

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(x, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(y, x, y)$

SAT Solver

LIA

\forall -instantiation

unsat

$\lambda xy. \text{ite}(x \geq y, x, y)$

\Rightarrow Simplify

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(x, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(y, x, y)$

SAT Solver

LIA

\forall -instantiation

unsat

$\lambda xy. \text{ite}(x \geq y, x, y)$

Desired function

Single Invocation Synthesis in SMT

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

$\forall z. \neg \text{isMax}(z, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(x, x, y)$
 $\forall z. \neg \text{isMax}(z, x, y) \Rightarrow \neg \text{isMax}(y, x, y)$

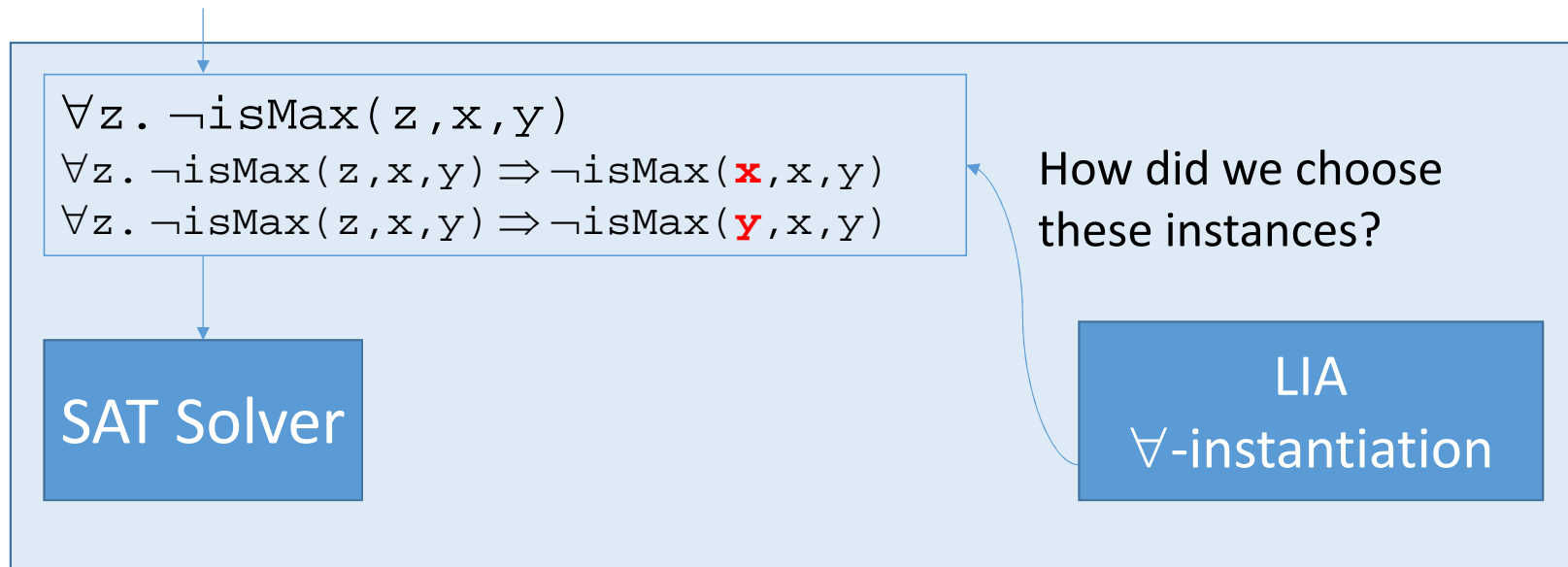
SAT Solver

**How did we choose
these instances?**

LIA
 \forall -instantiation

Single Invocation Synthesis in SMT

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



\Rightarrow Use *counterexample-guided quantifier instantiation* (CEGQI)

Variants used in [Monniaux 2010, Komuravelli et al 2014, Reynolds et al 2015, Dutertre 2015, Bjorner/Janota 2016, Fedukovich et al 2016, Preiner et al 2017]

Counterexample-Guided \forall -Instantiation

Quantifier Elimination Procedures

$\Leftarrow(\Rightarrow)?$

Instantiation-Based procedures for $\exists\forall$ formulas

\Leftrightarrow

Synthesis procedures for single-invocation properties

Counterexample-Guided \forall -Instantiation

- SMT+ \forall linear arithmetic [Monniaux 2010, Reynolds et al 2015, Dutertre 2015, Bjorner/Janota 2016]
 - Based on maximal lower (minimal upper) bounds
Analogous to [Loos+Wiespfenning 93]
 - Based on interior point method:
Analogous to [Ferrante+Rackoff 79]
 - For integers: based on maximal lower (minimal upper) bounds (+c)
Analogous to [Cooper 72]
- SMT + \forall BV, QBF, finite domains [Wintersteiger et al 2013, Rabe et al 2016, Preiner et al 2017]
 - Based on model value, SyGuS, others?
- SMT + Strings, sets, floating points, datatypes
 - ???

Finite instantiation strategy \tilde{O} sound and complete synthesis procedure for s.i.

Counterexample-Guided \forall -Instantiation

- SMT+ \forall linear arithmetic [Monniaux 2010, Reynolds et al 2015, Dutertre 2015, Bjorner/Janota 2016]
 - Based on maximal lower (minimal upper) bounds
Analogous to [Loos+Wiespfenning 93]
 - Based on interior point method:
Analogous to [Ferrante+Rackoff 79]
 - For integers: based on maximal lower (minimal upper) bounds (+c)
Analogous to [Cooper 72]
- SMT + \forall BV, QBF, finite domains [Wintersteiger et al 2013, Rabe et al 2016, Preiner et al 2017]
 - Based on model value, SyGuS, others?
- SMT + Strings, sets, floating points, datatypes
 - ???

CHALLENGE #1:

How do we develop instantiation procedures for new SMT *theories*?

Finite instantiation strategy \tilde{O} *sound* and *complete* synthesis procedure for *s.i.*

Comparison of Synthesis Approaches

- *SMT + \exists -instantiation*

Pro: Very fast

Pro: Complete for (in)feasibility

Con: Non-optimal solutions

Con: Only for single-invocation conjectures

- Enumerative Search

Con: Typically very slow

Con: Cannot show infeasibility

Pro: Optimal (shortest) solutions

Pro: Applies to all second-order conjectures

Comparison of Synthesis Approaches

- *SMT + \exists -instantiation*

Pro: Very fast

Pro: Complete for (in)feasibility

Con: Non-optimal solutions

Con: Only for single-invocation conjectures

CHALLENGES

- Enumerative Search

Con: Typically very slow

Con: Cannot show infeasibility

Pro: Optimal (shortest) solutions

Pro: Applies to all second-order conjectures

Shorter Solutions via Proof Analysis

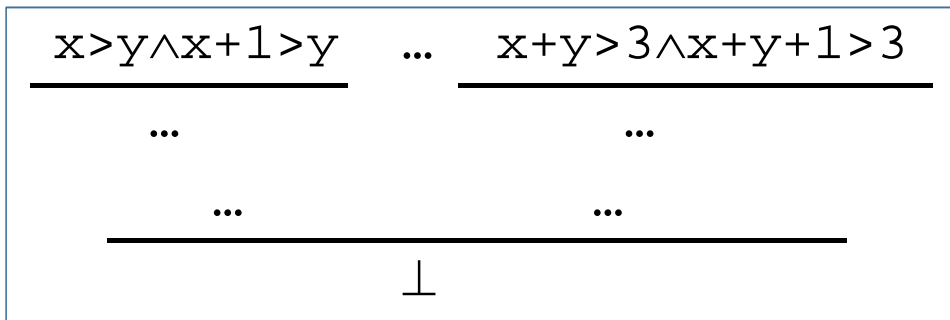
unsat

$$\frac{\frac{x > y \wedge x + 1 > y \quad \dots \quad x + y > 3 \wedge x + y + 1 > 3}{\dots}}{\dots} \perp$$

$f = \lambda x. \text{ite}(x > y \wedge x + 1 > y, t_1, t_2)$

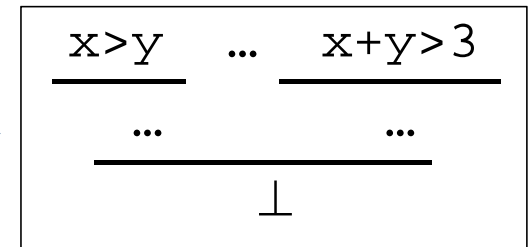
Shorter Solutions via Proof Analysis

unsat



unsat

Simplify



$f = \lambda x. \text{ite}(x > y \wedge x + 1 > y, t_1, t_2)$

$f = \lambda x. \text{ite}(x > y, t_1, t_2)$

CHALLENGE #2:

What *proofs/analysis* techniques are relevant for solution minimization?

What if conjecture is *Partially Single Invocation*?

$$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$$

- Can single invocation techniques be leveraged beyond *single-invocation* conjectures?

What if conjecture is *Partially Single Invocation*?

$$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$$

E.g. invariant synthesis problem for I w.r.t pre , T , post

What if conjecture is *Partially Single Invocation*?

$$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$$

Partition into...

$$\exists I. \forall x. (\text{pre}(x) \Rightarrow I(x)) \wedge (I(x) \Rightarrow \text{post}(x))$$

Single-invocation portion

$$\exists I. \forall x x'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$$

Non-single-invocation portion

What if conjecture is *Partially Single Invocation*?

$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$

$\exists I. \forall x. (\text{pre}(x) \Rightarrow I(x)) \wedge (I(x) \Rightarrow \text{post}(x))$

$\exists I. \forall x x'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$

SMT Solver

Counterexample
Guided
 \forall -Instantiation

unsat

$\lambda x. \text{ite}((\text{pre}(x) \Rightarrow \mathbf{T}) \wedge (\mathbf{T} \Rightarrow \text{post}(x)), \mathbf{T}, \perp)$

What if conjecture is *Partially Single Invocation*?

$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$

$\exists I. \forall x. (\text{pre}(x) \Rightarrow I(x)) \wedge (I(x) \Rightarrow \text{post}(x))$

$\exists I. \forall x x'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$

SMT Solver

Counterexample
Guided
 \forall -Instantiation

unsat

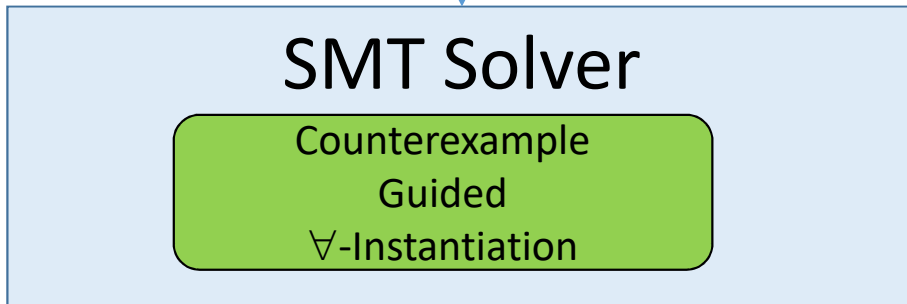
$\lambda x. \text{post}(x)$

What if conjecture is *Partially Single Invocation*?

$$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$$

$$\exists I. \forall x. (\text{pre}(x) \Rightarrow I(x)) \wedge (I(x) \Rightarrow \text{post}(x))$$

$$\exists I. \forall x x'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$$



unsat $\lambda x. \text{post}(x)$

Candidate invariant
 \Rightarrow check against non-single invocation portion

What if conjecture is *Partially Single Invocation*?

$$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$$

$$\exists I. \forall x. (\text{pre}(x) \Rightarrow I(x)) \wedge (I(x) \Rightarrow \text{post}(x)) \wedge S'$$

$$\exists I. \forall x x'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$$

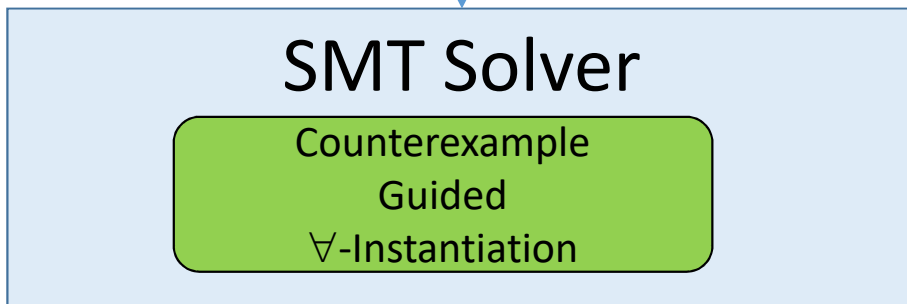
$\lambda x. \text{post}(x)$ solution?

Refine?

No

Yes

$\lambda x. \text{post}(x)$



\Rightarrow Related to property-directed reachability (PDR) [Bradley 2011]

What if conjecture is *Partially Single Invocation*?

$$\exists I. \forall x x'. (\text{pre}(x) \Rightarrow I(x)) \wedge ((I(x) \wedge T(x, x')) \Rightarrow I(x')) \wedge (I(x) \Rightarrow \text{post}(x))$$

$$\exists I. \forall x. (\text{pre}(x) \Rightarrow I(x)) \wedge (I(x) \Rightarrow \text{post}(x)) \wedge S'$$

$$\exists I. \forall x x'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$$

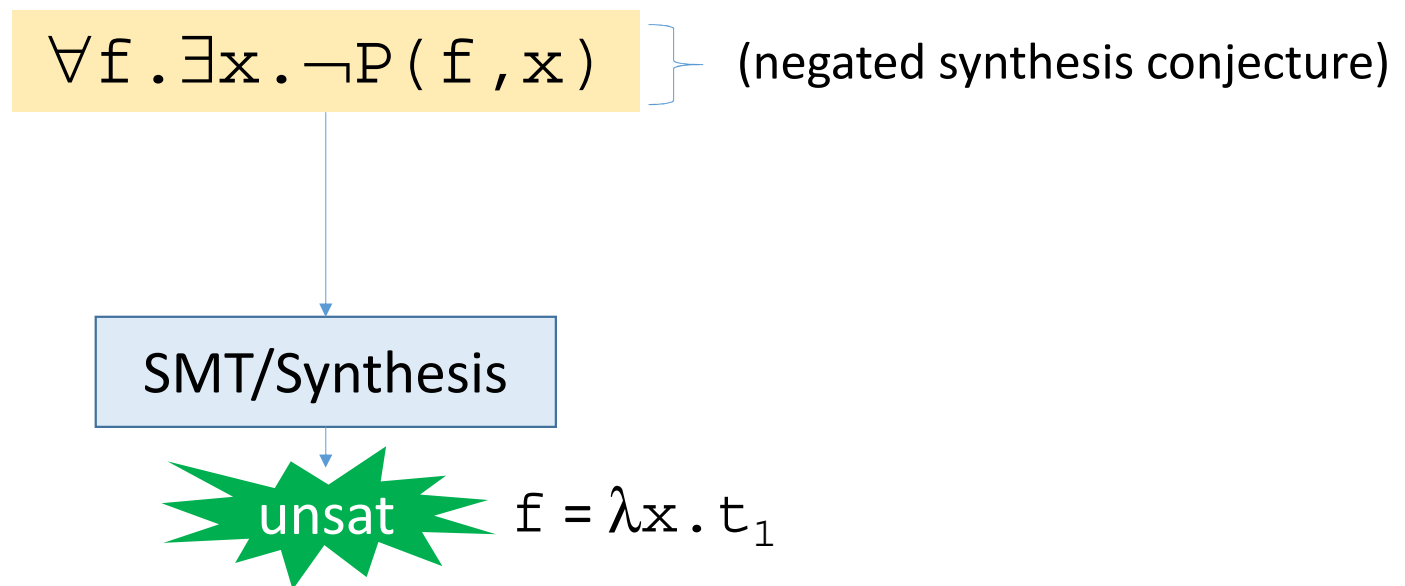
SMT Solver

Counterexample
Guided
 \forall -Instantiation

CHALLENGE #3:

How can we combine first-order and second-order techniques for *function synthesis*?

Synthesis for Higher-Order Theorem Proving?



Synthesis for Higher-Order Theorem Proving?

