

Fast and Slow Synthesis Procedures in SMT

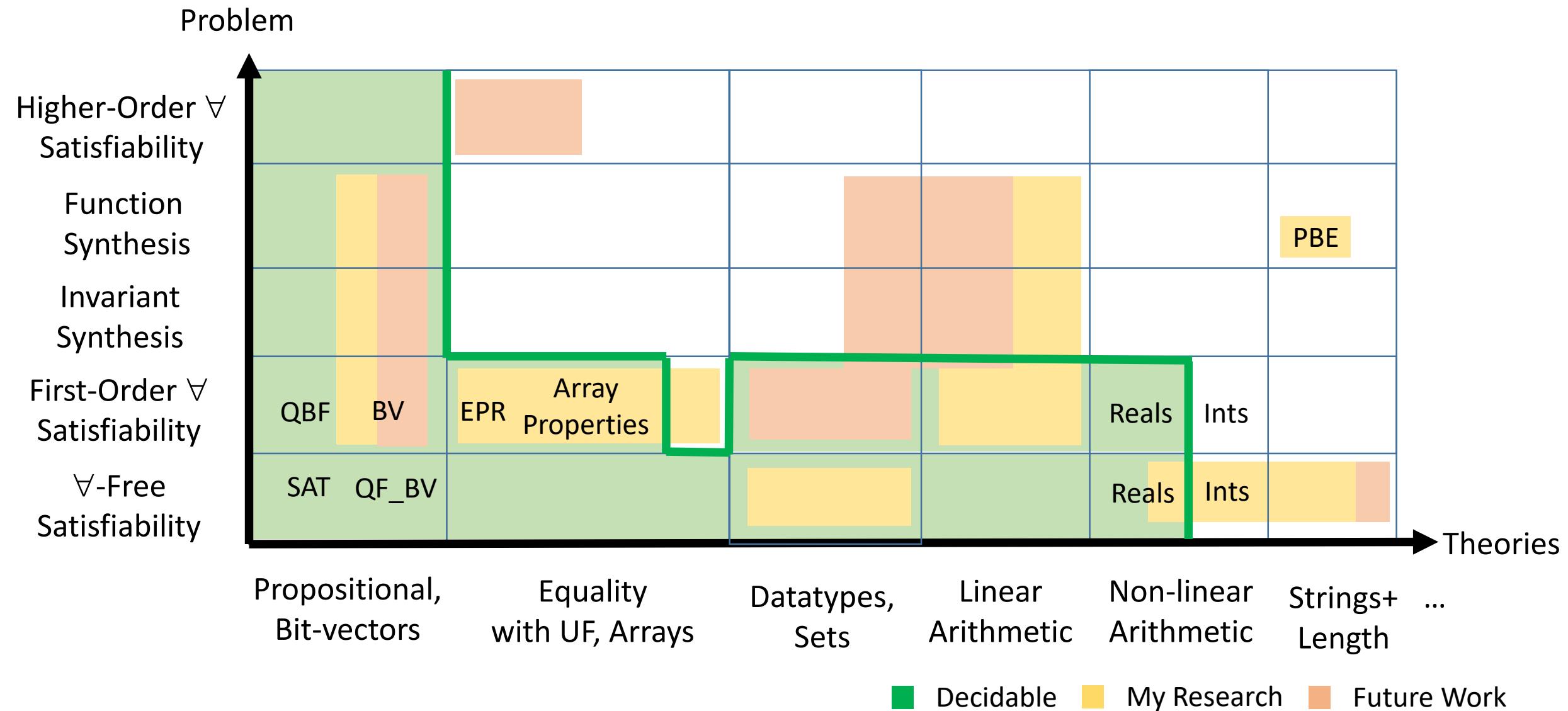
Andrew Reynolds

Dagstuhl Seminar on Higher-Order Logic

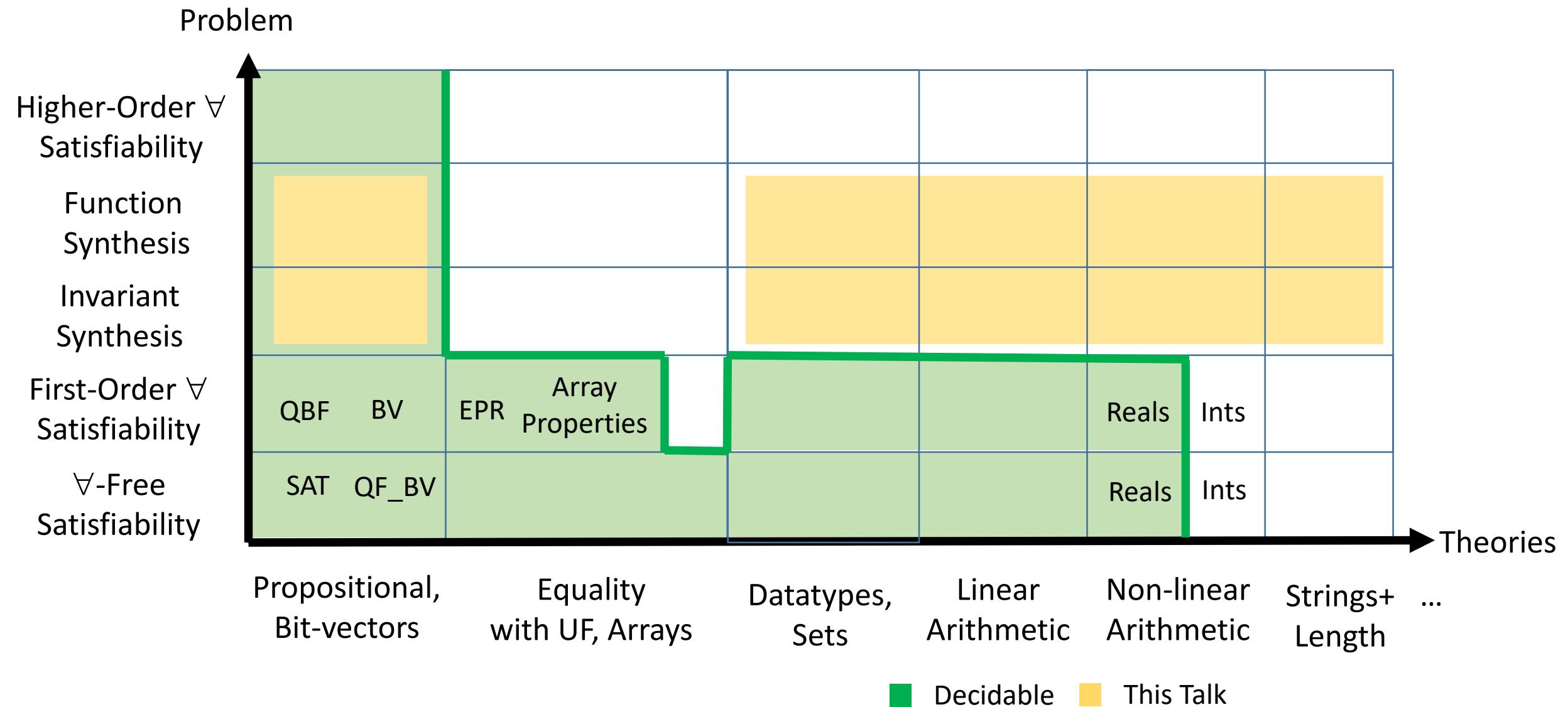
September 14, 2017



My Research in SMT



This Talk: Synthesis for (Interpreted) Theories



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

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



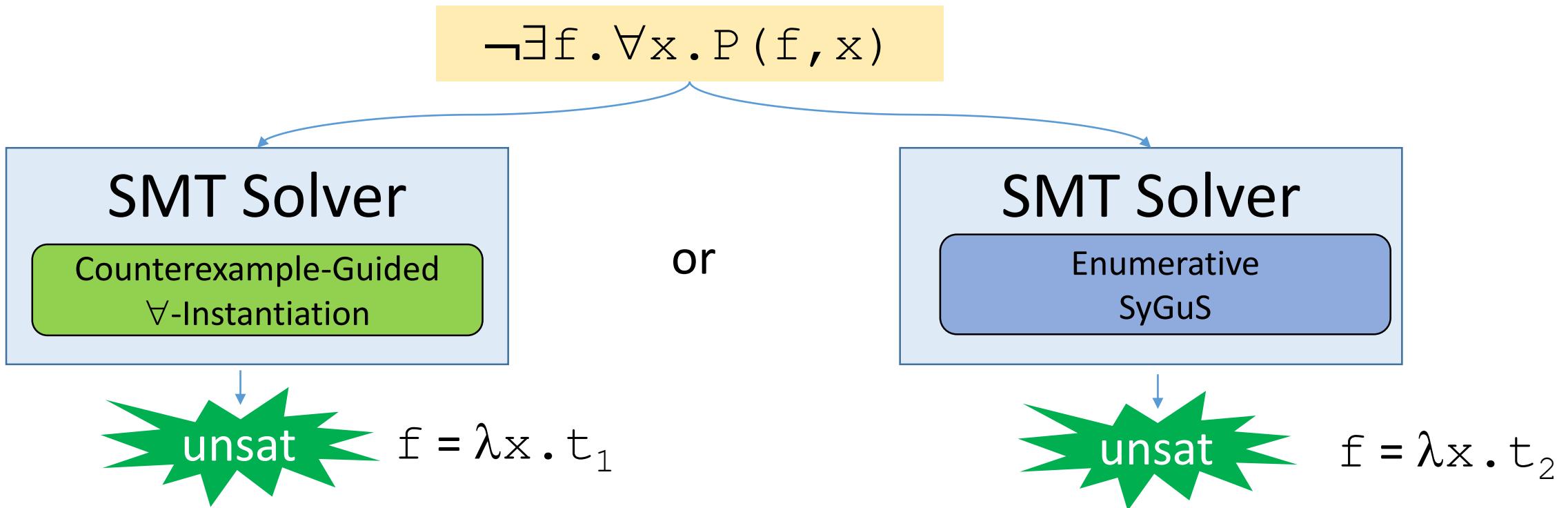
There exists a function f for which property P holds for all x

Refutation-Based Synthesis in SMT

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

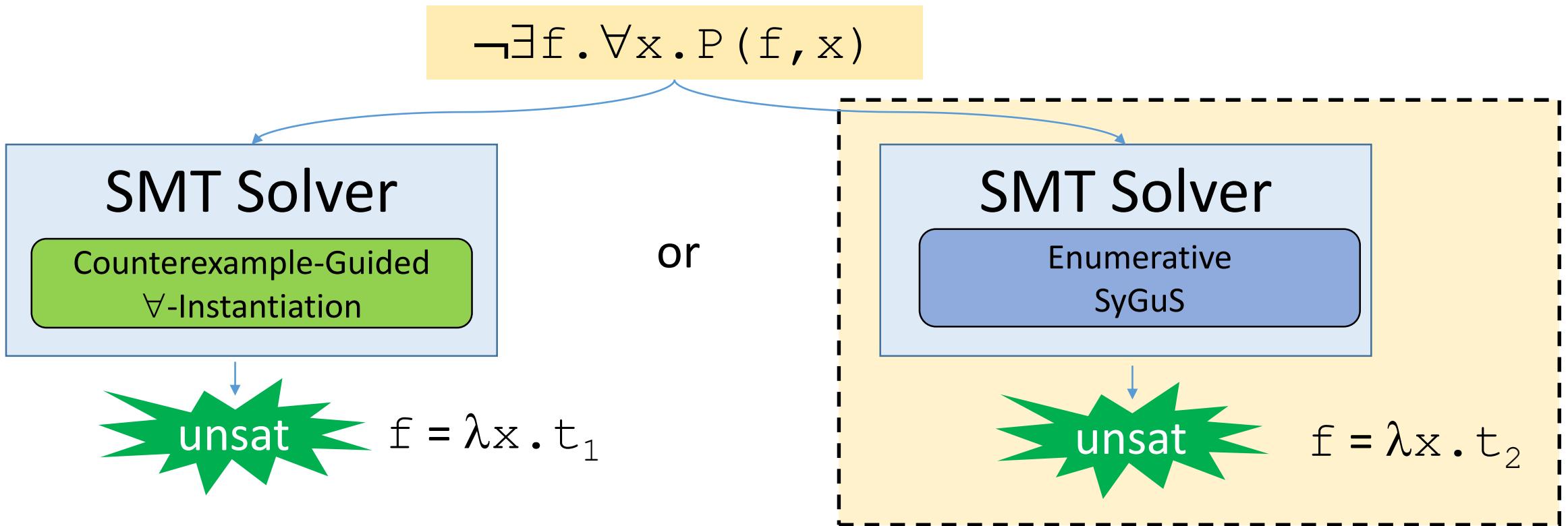
(**negated** synthesis conjecture)

Refutation-Based Synthesis in SMT



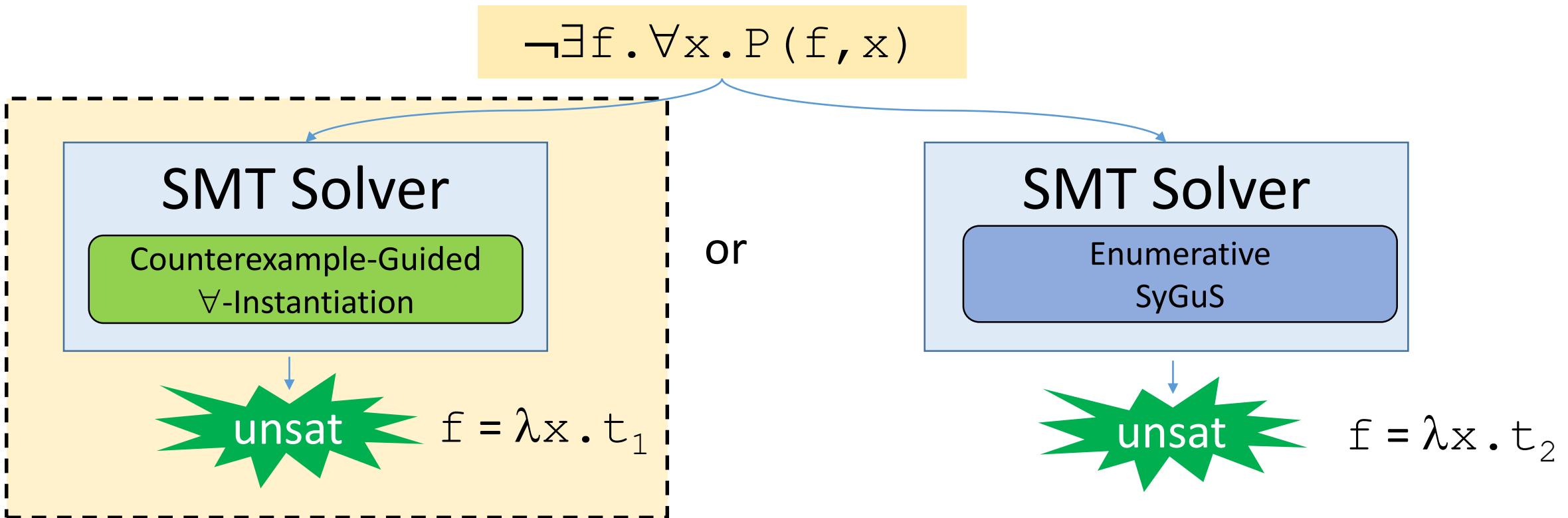
- Two approaches for refutation-based synthesis in SMT solvers [Reynolds et al CAV2015]

Refutation-Based Synthesis in SMT



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

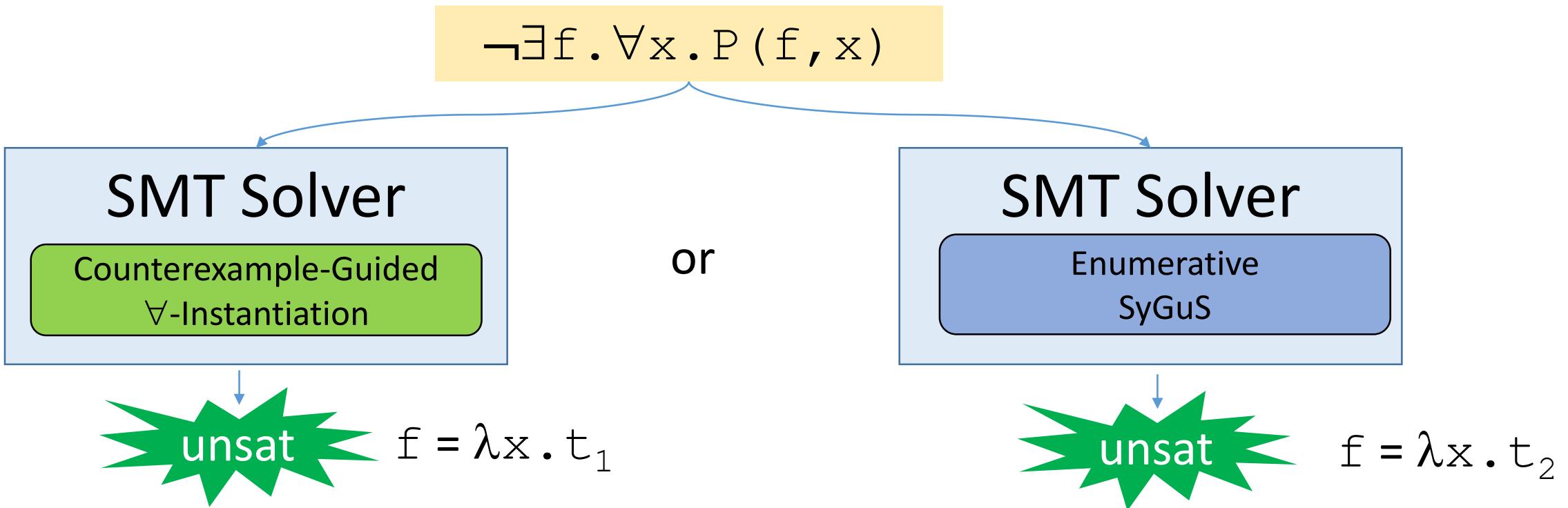
Refutation-Based Synthesis in SMT



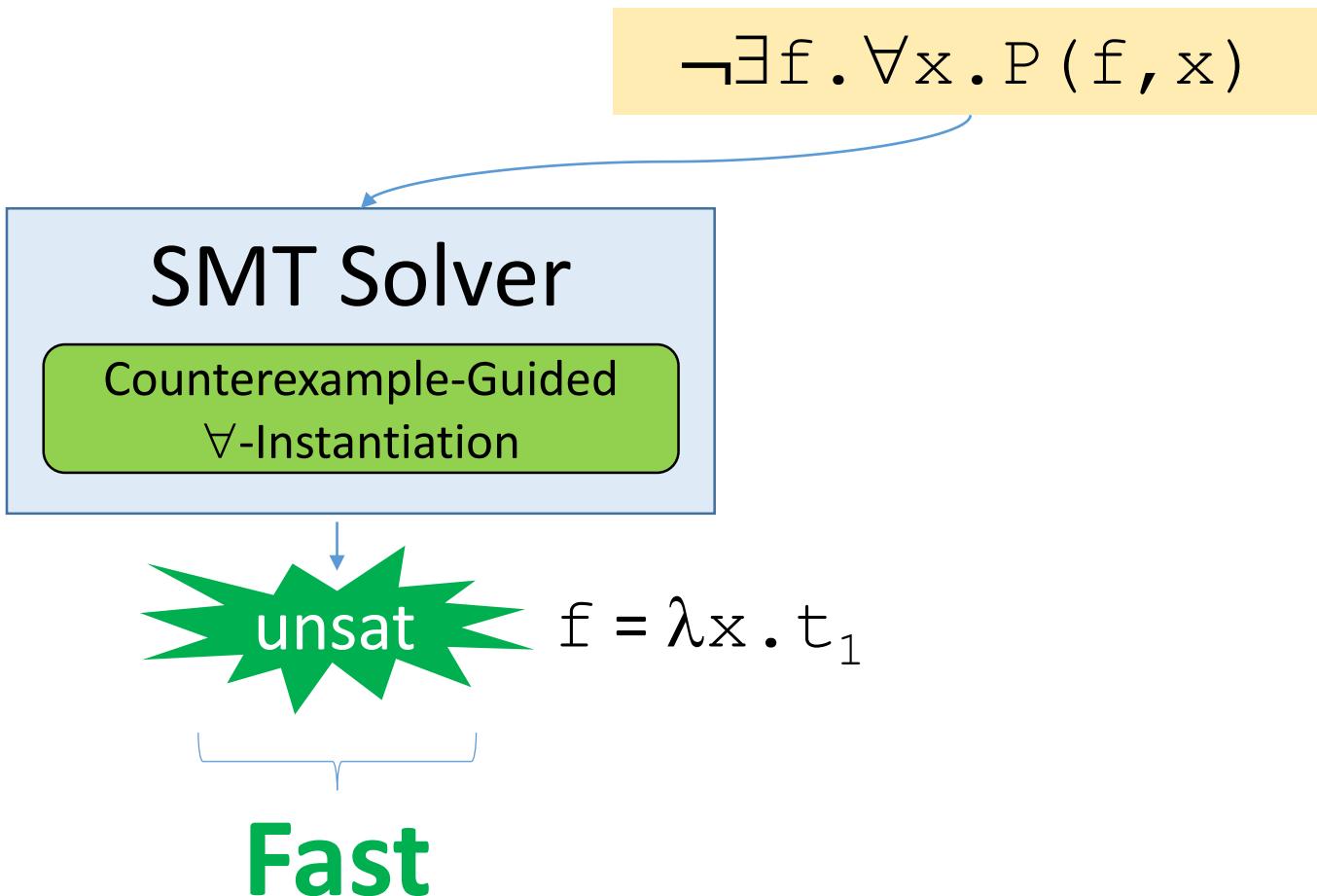
⇒ Based on first-order quantifier instantiation in SMT

[Monniaux 2010, Bjorner 2012, Komuravelli et al 2014, Dutertre 2015...]

Refutation-Based Synthesis in SMT



Refutation-Based Synthesis in SMT



Fast Synthesis in SMT Solvers

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

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

“ $f(x, y)$ is the maximum of x and y ”

Fast Synthesis in SMT Solvers

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

Int \times Int \rightarrow Int



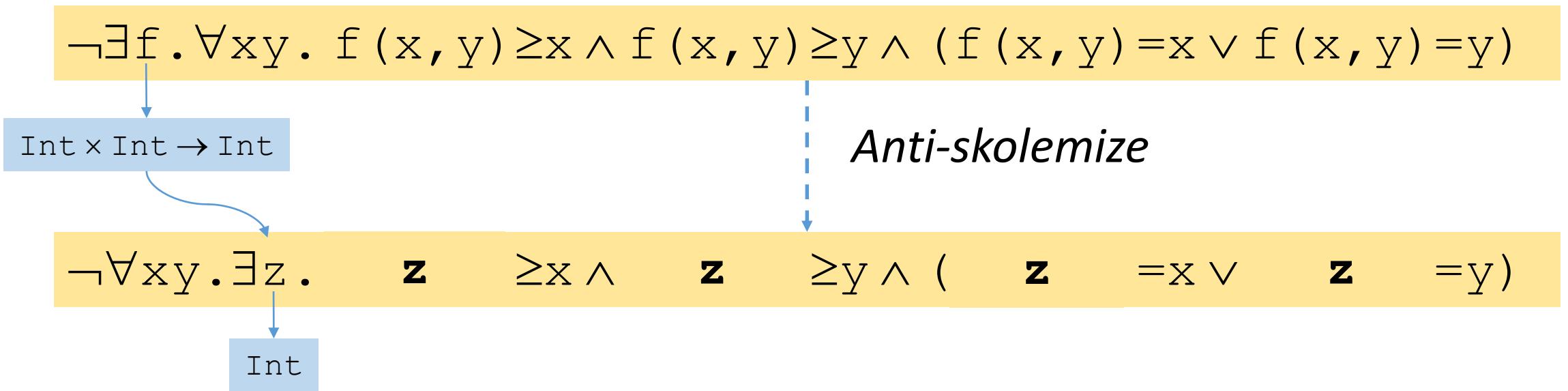
All occurrence of f are in terms of the form $f(x,y)$
⇒ “single invocation” synthesis conjectures

Fast Synthesis in SMT Solvers

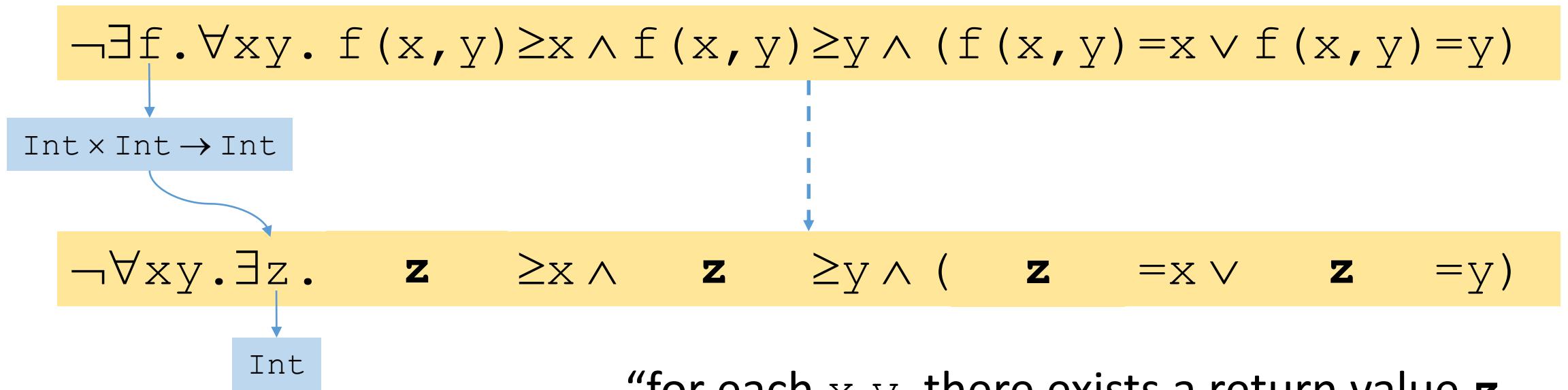
$$\neg \exists f. \forall xy. f(x, y) \geq x \wedge f(x, y) \geq y \wedge (f(x, y) = x \vee f(x, y) = y)$$


Int × Int → Int

Fast Synthesis in SMT Solvers

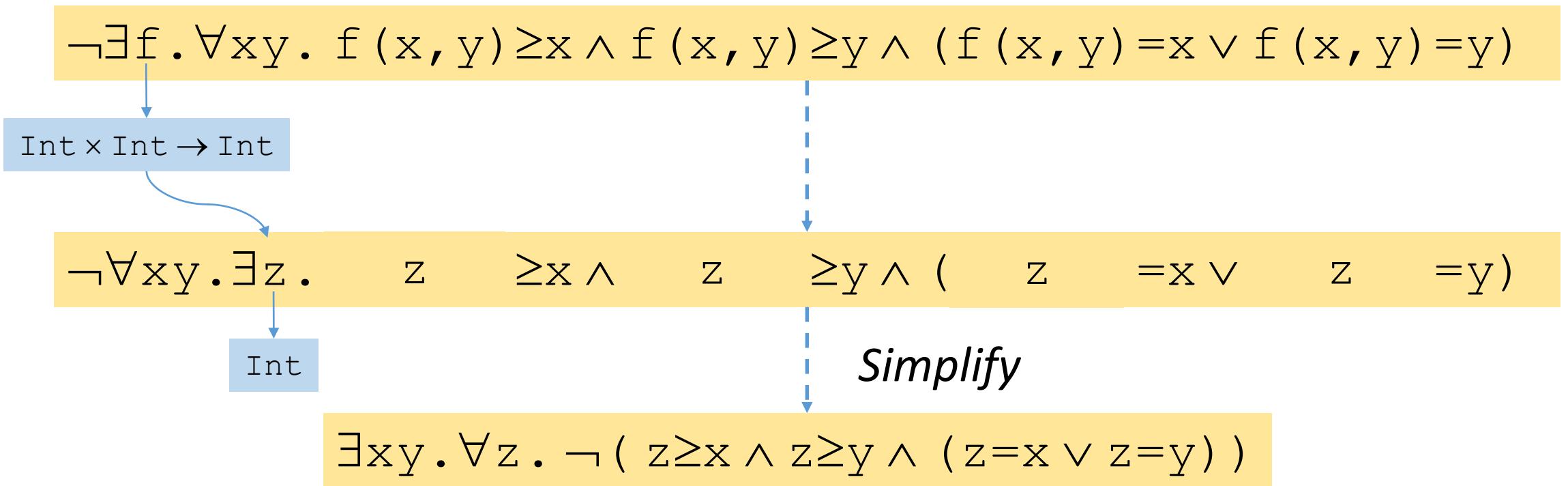


Fast Synthesis in SMT Solvers

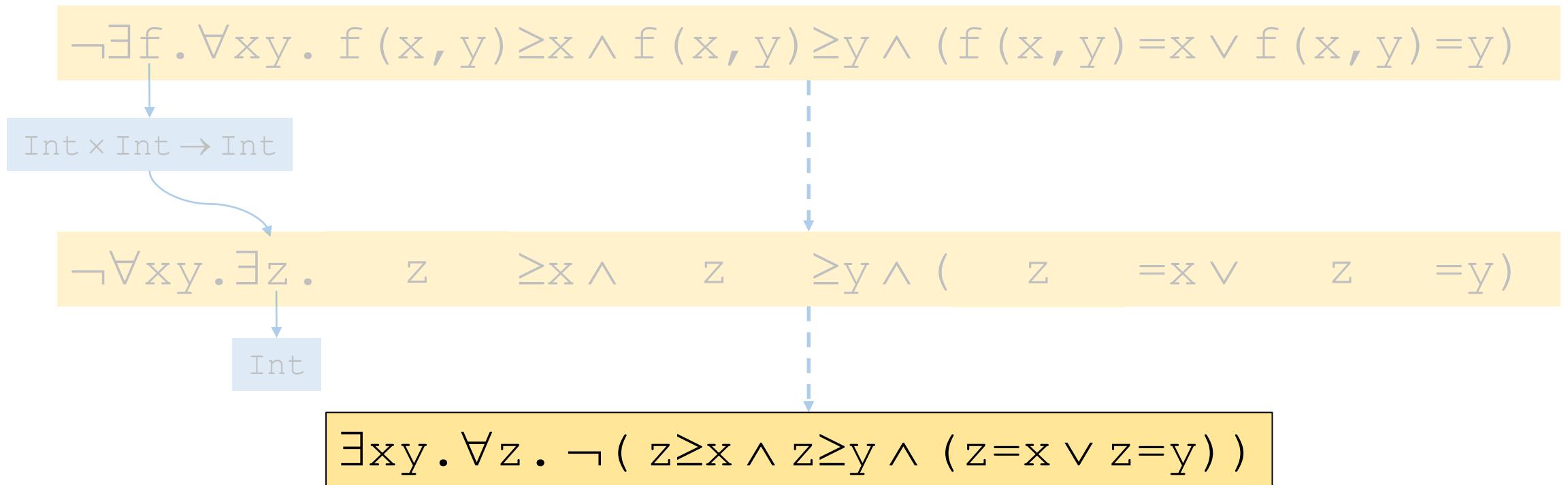


“for each x, y , there exists a return value z that is the maximum of x and y ”

Fast Synthesis in SMT Solvers



Fast Synthesis in SMT Solvers



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

Fast Synthesis in SMT Solvers

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

SAT Solver

LIA
 \forall -instantiation

Fast Synthesis in SMT Solvers

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

SAT Solver

LIA
 \forall -instantiation

Fast Synthesis in SMT Solvers

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

Translate to first-order

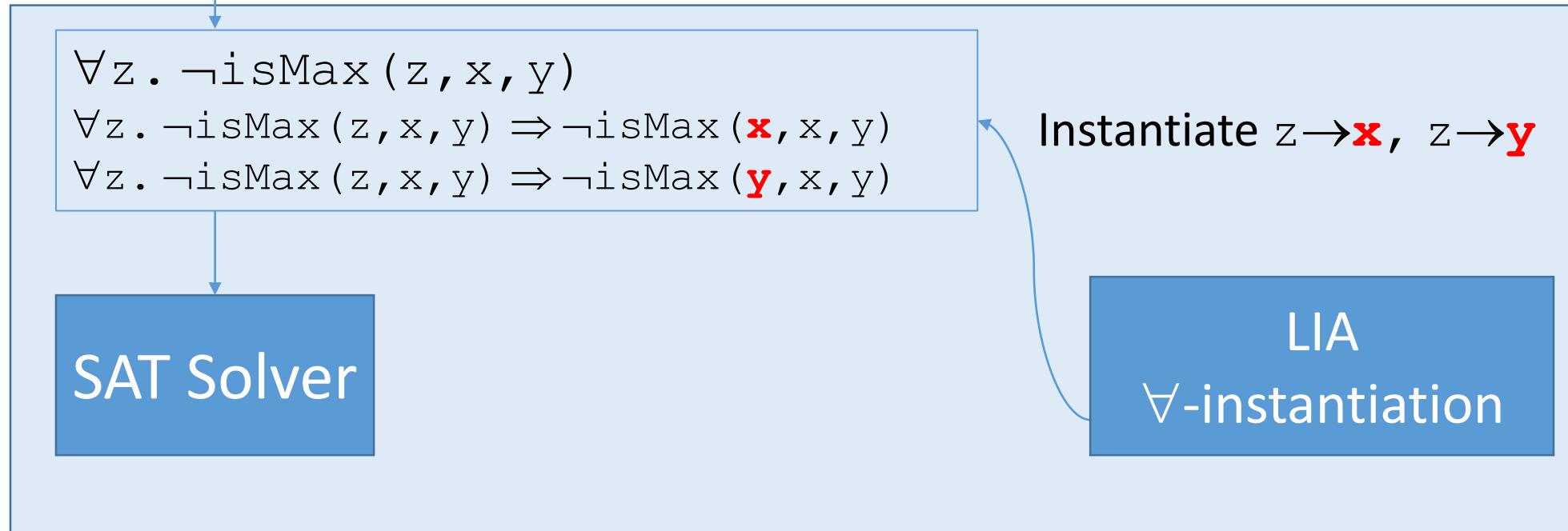
$$\forall z. \neg \text{isMax}(z, x, y)$$

SAT Solver

LIA
 \forall -instantiation

Fast Synthesis in SMT Solvers

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



Fast Synthesis in SMT Solvers

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

SAT Solver

LIA
 \forall -instantiation



⇒ Solution for f can be constructed from unsatisfiable core of instantiations

Fast Synthesis in SMT Solvers

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

SAT Solver

LIA
 \forall -instantiation

unsat

$$\lambda xy. ?$$

Fast Synthesis in SMT Solvers

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

SAT Solver

LIA
 \forall -instantiation


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

Fast Synthesis in SMT Solvers

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

SAT Solver

LIA
 \forall -instantiation

unsat

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

Fast Synthesis in SMT Solvers

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

SAT Solver

LIA
 \forall -instantiation


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

Fast Synthesis in SMT Solvers

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

SAT Solver

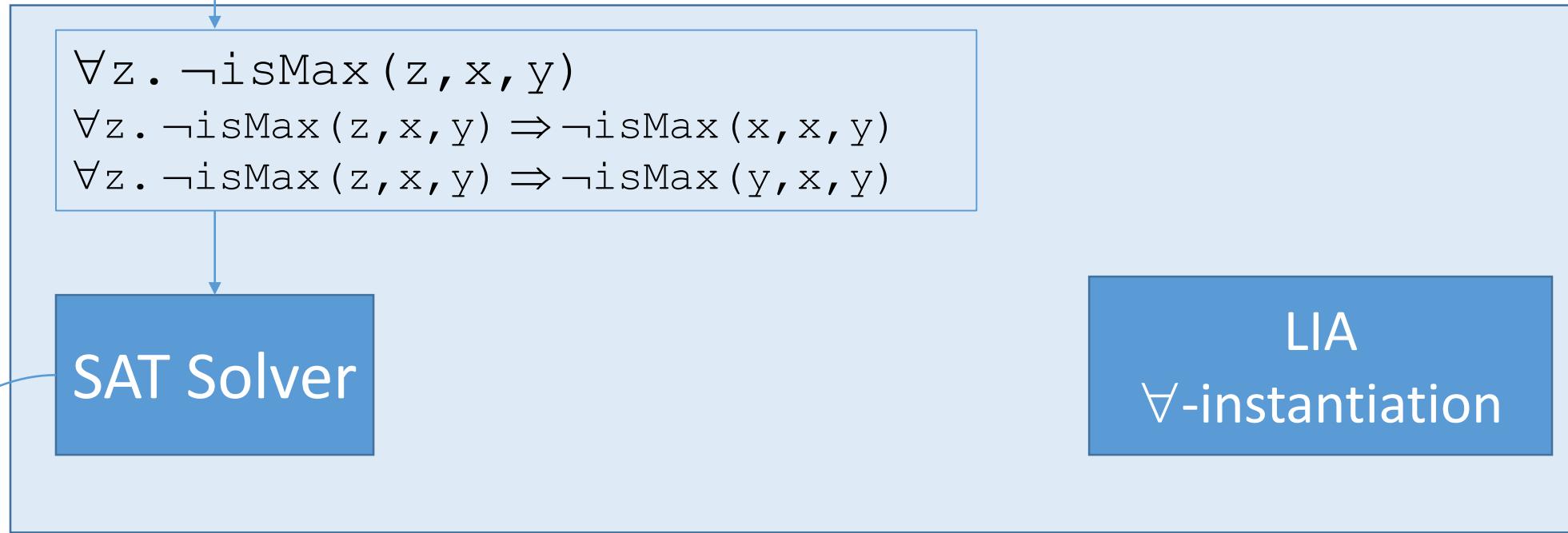
LIA
 \forall -instantiation

unsat

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

\Rightarrow Simplify

Fast Synthesis in SMT Solvers

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

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

Desired function

Fast Synthesis in SMT Solvers

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

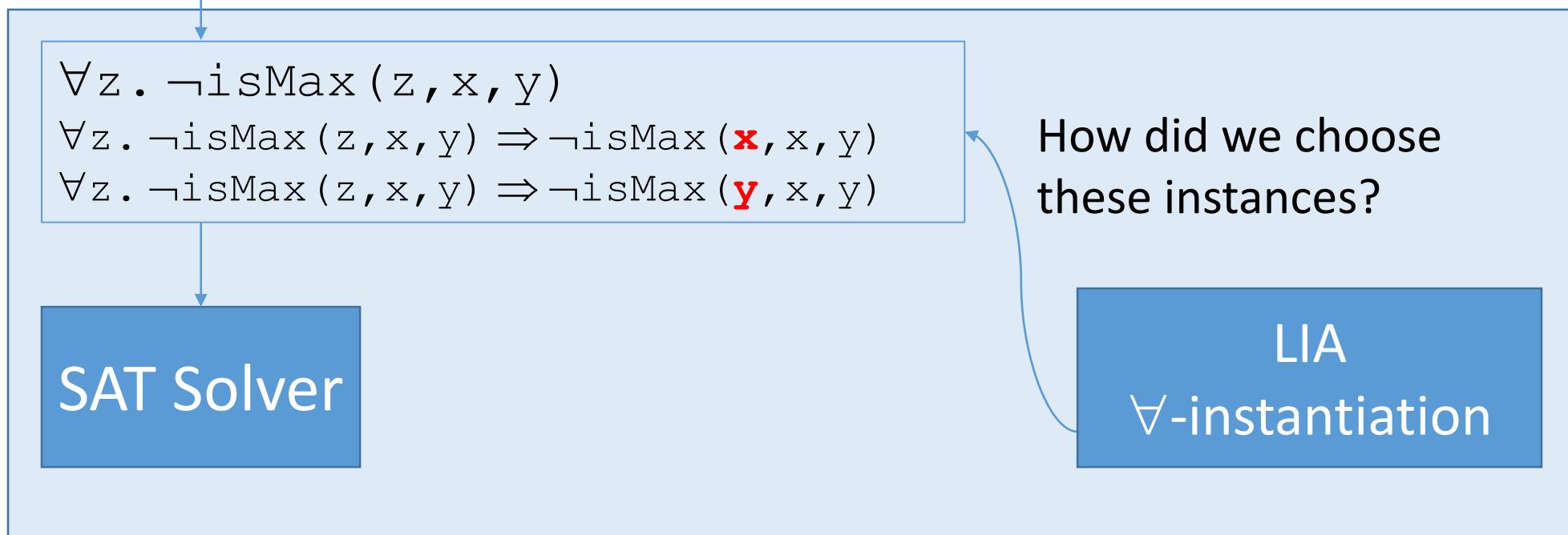
SAT Solver

How did we choose
these instances?

LIA
 \forall -instantiation

Fast Synthesis in SMT Solvers

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



⇒ Use *counterexample-guided quantifier instantiation* (CEGQI)

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

Counterexample-Guided \forall -Instantiation

Quantifier Elimination Procedures

$\Leftarrow(\Rightarrow)?$

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

\iff

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, \forall finite domains [Wintersteiger et al 2013, Rabe et al 2016, Preiner et al 2017]
 - Based on model value, SyGuS, others?
- SMT + \forall Strings, \forall sets, \forall floating points, \forall datatypes
 - to be determined?

Finite instantiation strategy \Leftrightarrow sound and complete synthesis procedure for s.i.

CEGQI : Quality of Solutions

- Users of synthesis tools also care about **quality of solutions**
 - CEGQI produced the intended solution for:

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



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

CEGQI : Quality of Solutions

- Users of synthesis tools also care about **quality of solutions**
 - CEGQI produced the intended solution for:

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



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

- ...but will not produce the intended solution for:

$$\exists f. \forall x. (x=1 \Rightarrow f(x)=0) \wedge (x=2 \Rightarrow f(x)=1) \wedge (x=3 \Rightarrow f(x)=2)$$



$$\lambda x. x - 1$$

CEGQI : Quality of Solutions

- Users of synthesis tools also care about **quality of solutions**
 - CEGQI produced the intended solution for:

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



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

- ...but will not produce the intended solution for:

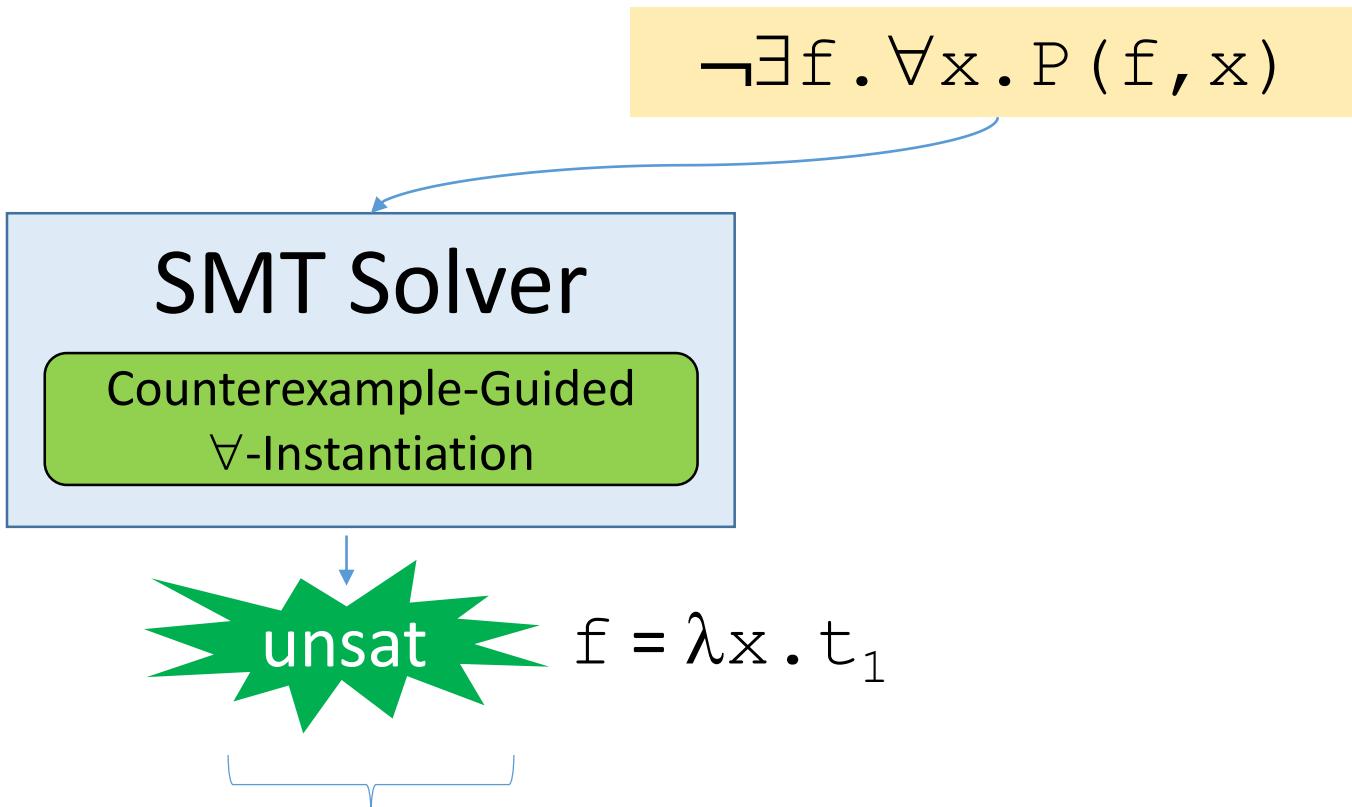
$$\exists f. \forall x. (x=1 \Rightarrow f(x)=0) \wedge (x=2 \Rightarrow f(x)=1) \wedge (x=3 \Rightarrow f(x)=2)$$



$$\lambda x. \text{ite}(x=1, 0, x=2, 1, 2)$$

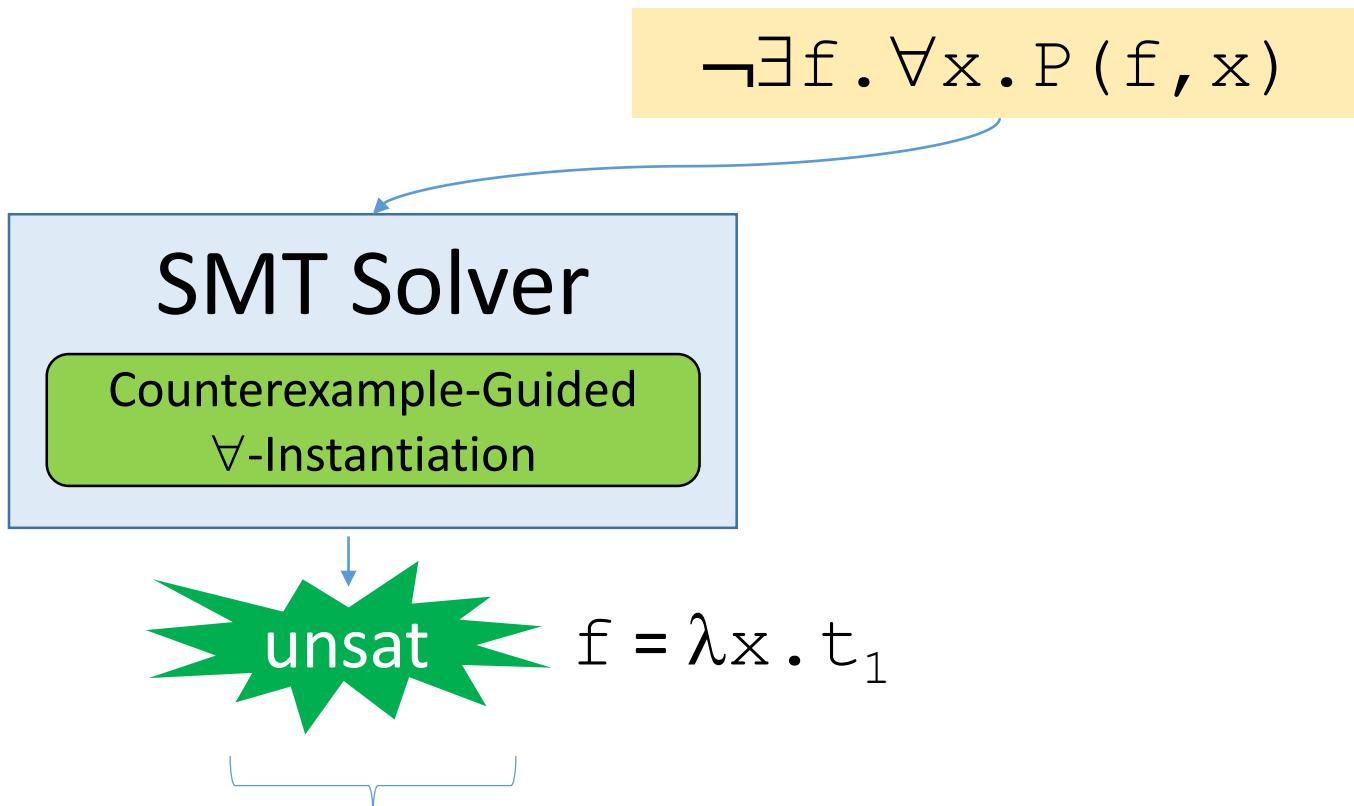
⇒ This kind of solution is not useful for invariant synthesis and programming-by-examples

Refutation-Based Synthesis in SMT



Fast (and Complete)

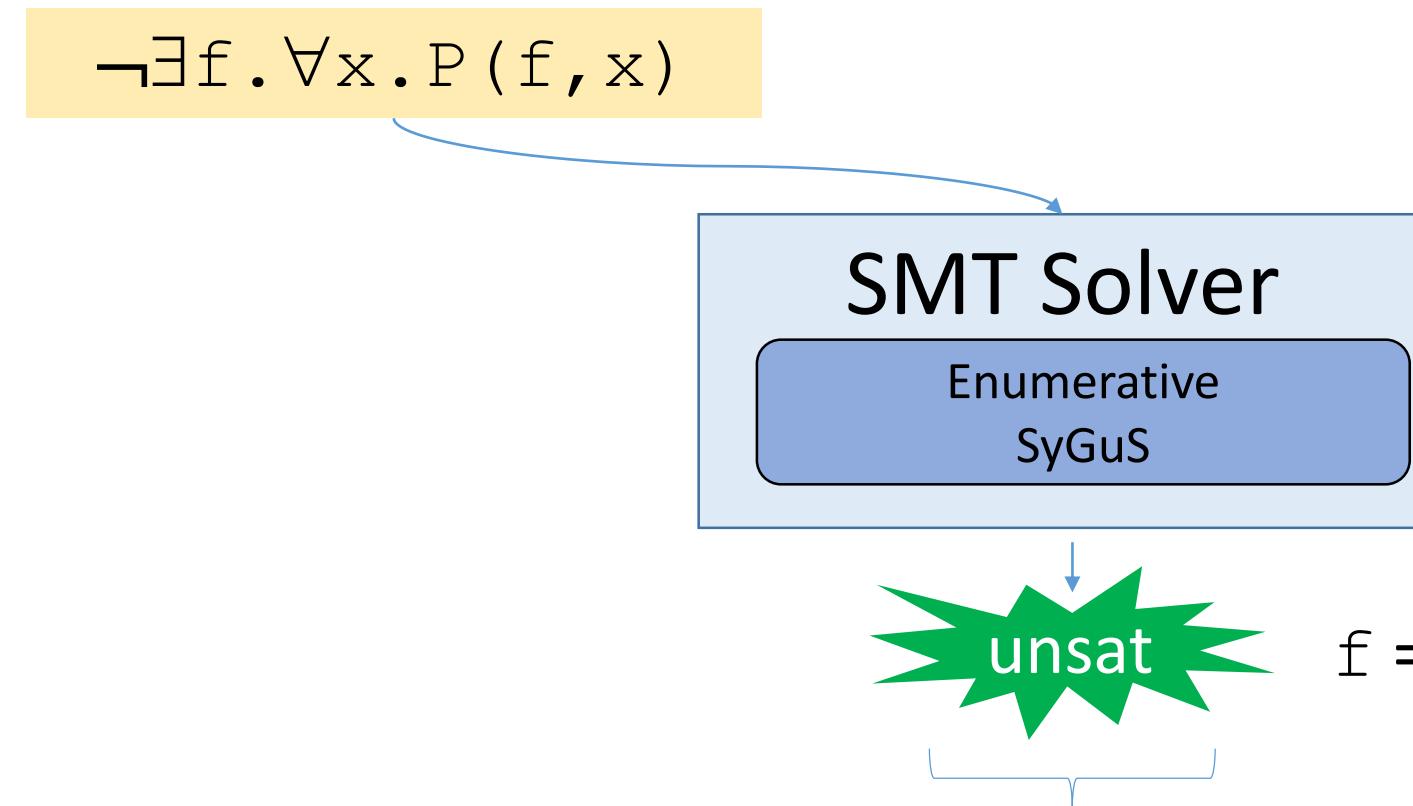
Refutation-Based Synthesis in SMT



Fast (and Complete)

...but restricted to single invocation properties,
and produces non-optimal solutions

Refutation-Based Synthesis in SMT



Slow

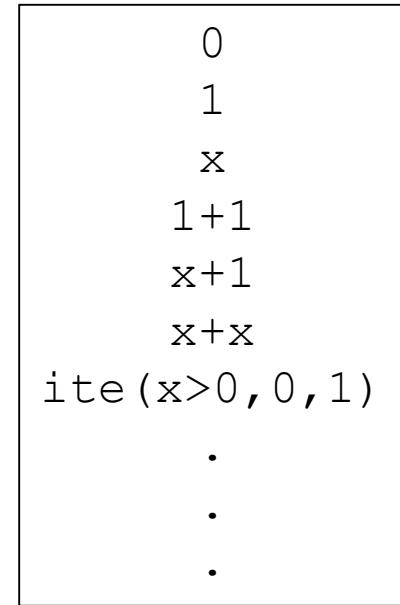
...but applicable to any property,
produces optimal (smallest) solutions

Slow Synthesis in SMT Solvers

Conjecture

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

Test



Syntactic
Restrictions \mathcal{R}

- Idea: enumerate terms generated by the grammar
- Approach used by number of synthesis solvers [Solar-Lezama 2013, Udupa et al 2013]

Slow Synthesis in SMT Solvers

Conjecture

$$\exists f. \forall xy. f(x, y) \geq x \wedge \textcolor{red}{f(x, y)} = \textcolor{blue}{f(y, x)}$$

Syntactic Restrictions

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |  
       ite(fBool, fInt, fInt)  
fBool := >(fInt, fInt) | =(fInt, fInt)
```

Slow Synthesis in SMT Solvers

Conjecture

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

Inductive Datatype

~~Syntactic Restrictions~~

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |  
       ite(fBool, fInt, fInt)  
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

View syntactic restrictions as an *inductive datatype*

Slow Synthesis in SMT Solvers

Conjecture

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

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |  
       ite(fBool, fInt, fInt)  
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

Slow Synthesis in SMT Solvers

Conjecture

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

Int × Int → Int

$$\exists \mathbf{d}. \forall xy. \mathbf{E}(\mathbf{d}, x, y) \geq x \wedge \mathbf{E}(\mathbf{d}, x, y) = \mathbf{E}(\mathbf{d}, y, x)$$

fInt

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |  
        ite(fBool, fInt, fInt)  
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

Encode using *deep embedding* involving fInt

Slow Synthesis in SMT Solvers

Conjecture

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

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |  
       ite(fBool, fInt, fInt)  
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

$$\exists d. \forall xy. E(d, x, y) \geq x \wedge E(d, x, y) = E(d, y, x)$$

$E: \mathbb{I} \times \text{Int} \times \text{Int} \rightarrow \text{Int}$

- $E(d, x, y)$ evaluates d for arguments x, y , e.g. $E(+(\mathbf{x}, \mathbf{y}), 2, 3) = 5$

Slow Synthesis in SMT Solvers

Conjecture

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



$$\exists d. \forall xy. E(d, x, y) \geq x \wedge E(d, x, y) = E(d, y, x)$$

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |
        ite(fBool, fInt, fInt)
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

\Rightarrow Solvable by combination of datatypes, LIA, UF, and \forall -instantiation

Enumerative Syntax-Guided Synthesis

Conjecture

$$\exists d. \forall x y. E(d, x, y) \geq x \wedge E(d, x, y) = E(d, y, x)$$

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |
         ite(fBool, fInt, fInt)
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

Enumerative Syntax-Guided Synthesis

Conjecture

$$\exists d. \forall x y. E(d, x, y) \geq x \wedge E(d, x, y) = E(d, y, x)$$

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |
          ite(fBool, fInt, fInt)
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

- Solver generates a stream of candidate models:

- $d^{\mathcal{M}} = x$
- $d^{\mathcal{M}} = y$
- $d^{\mathcal{M}} = +(1, y)$
- $d^{\mathcal{M}} = +(0, x)$
- $d^{\mathcal{M}} = +(y, 1)$
- ...

Enumerative Syntax-Guided Synthesis

Conjecture

$$\exists d. \forall x y. E(d, x, y) \geq x \wedge E(d, x, y) = E(d, y, x)$$

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |
         ite(fBool, fInt, fInt)
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

- Solver generates a stream of candidate models:

- $d^{\mathcal{M}} = x$
- $d^{\mathcal{M}} = y$
- $d^{\mathcal{M}} = + (1, y)$
- $d^{\mathcal{M}} = + (0, x)$
- $d^{\mathcal{M}} = + (y, 1)$

Optimization: Only consider terms $d^{\mathcal{M}}$ whose *analog* is unique up to theory-specific simplification ↓

Enumerative Syntax-Guided Synthesis

Conjecture

$$\exists d. \forall x y. E(d, x, y) \geq x \wedge E(d, x, y) = E(d, y, x)$$

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |
          ite(fBool, fInt, fInt)
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

- Solver generates a stream of candidate models, normalizes values ↓:

• $d^{\mathcal{M}} = x$...	x	=↓	x
• $d^{\mathcal{M}} = y$...	y	=↓	y
• $d^{\mathcal{M}} = +(1, y)$...	1+y	=↓	y+1
• $d^{\mathcal{M}} = +(0, x)$...	0+x	=↓	x
• $d^{\mathcal{M}} = +(y, 1)$...	y+1	=↓	y+1

Enumerative Syntax-Guided Synthesis

Conjecture

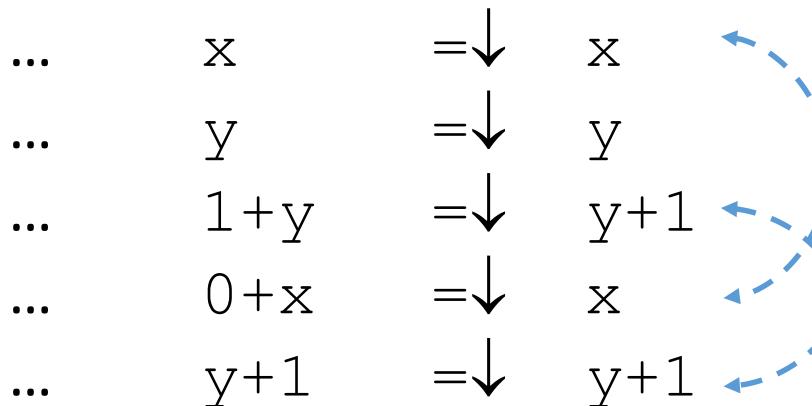
$$\exists d. \forall x y. E(d, x, y) \geq x \wedge E(d, x, y) = E(d, y, x)$$

Inductive Datatype

```
fInt := x | y | 0 | 1 | +(fInt, fInt) |
         ite(fBool, fInt, fInt)
fBool := ≥(fInt, fInt) | = (fInt, fInt)
```

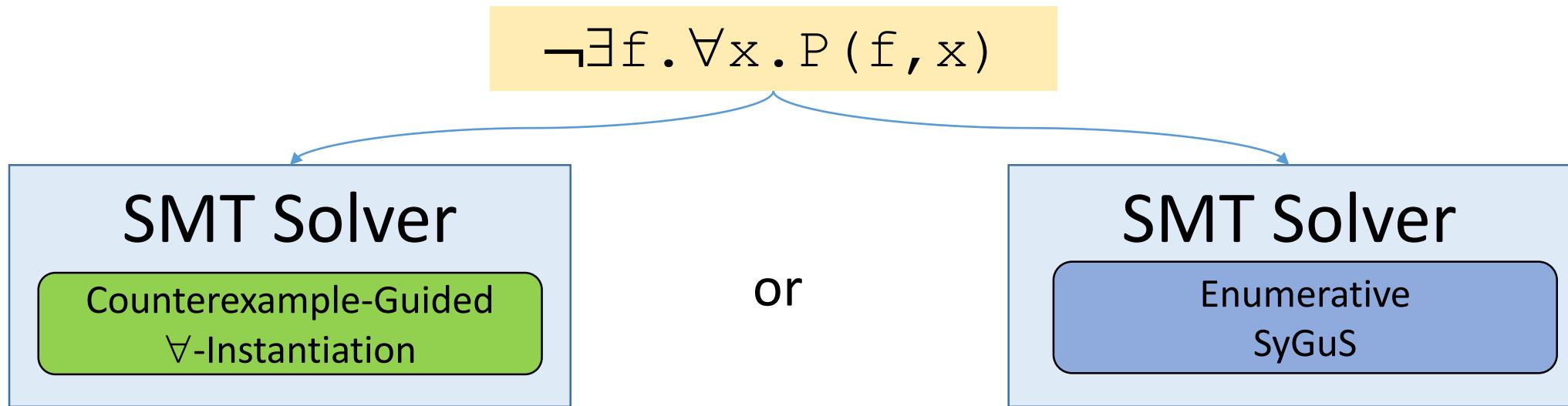
- Solver generates a stream of candidate models, normalizes values ↓:

- $d^{\mathcal{M}} = x$...
- $d^{\mathcal{M}} = y$...
- $d^{\mathcal{M}} = +(1, y)$...
- $d^{\mathcal{M}} = +(0, x)$...
- $d^{\mathcal{M}} = +(y, 1)$...



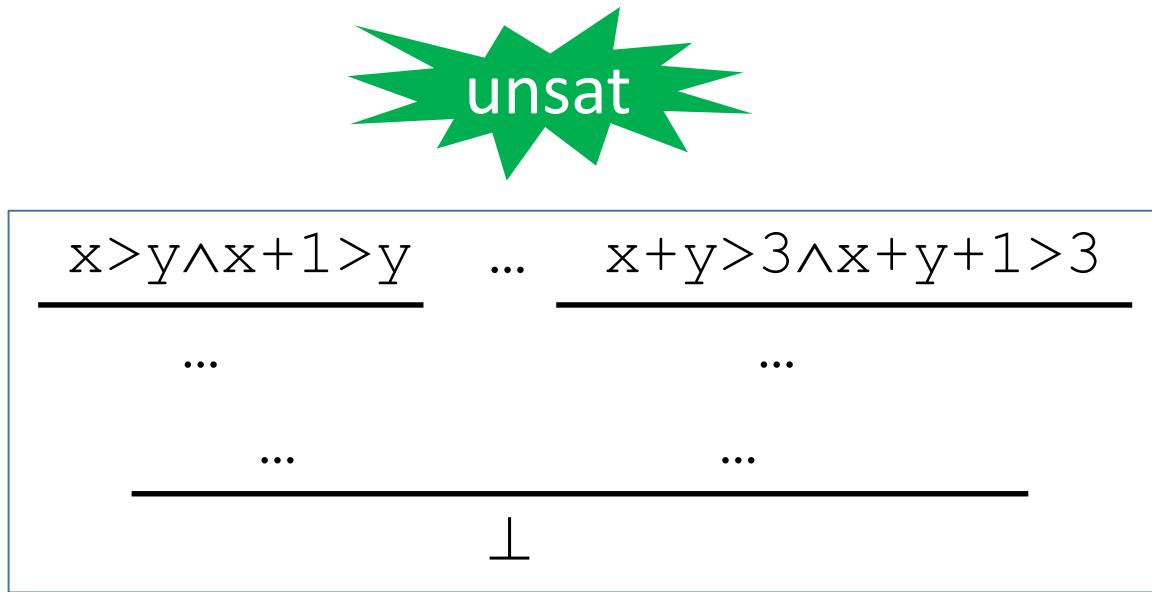
Avoid candidate solutions not unique up to theory normalization

Fast and Slow Synthesis Procedures in SMT



<p>Fast, Complete for (in)Feasibility Restricted to Single-Inv Properties, Non-optimal Solutions</p>	<p>Slow, Cannot show infeasibility Applies to all Second-Order Conjectures, Optimal (Shortest) Solutions</p>
----------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------

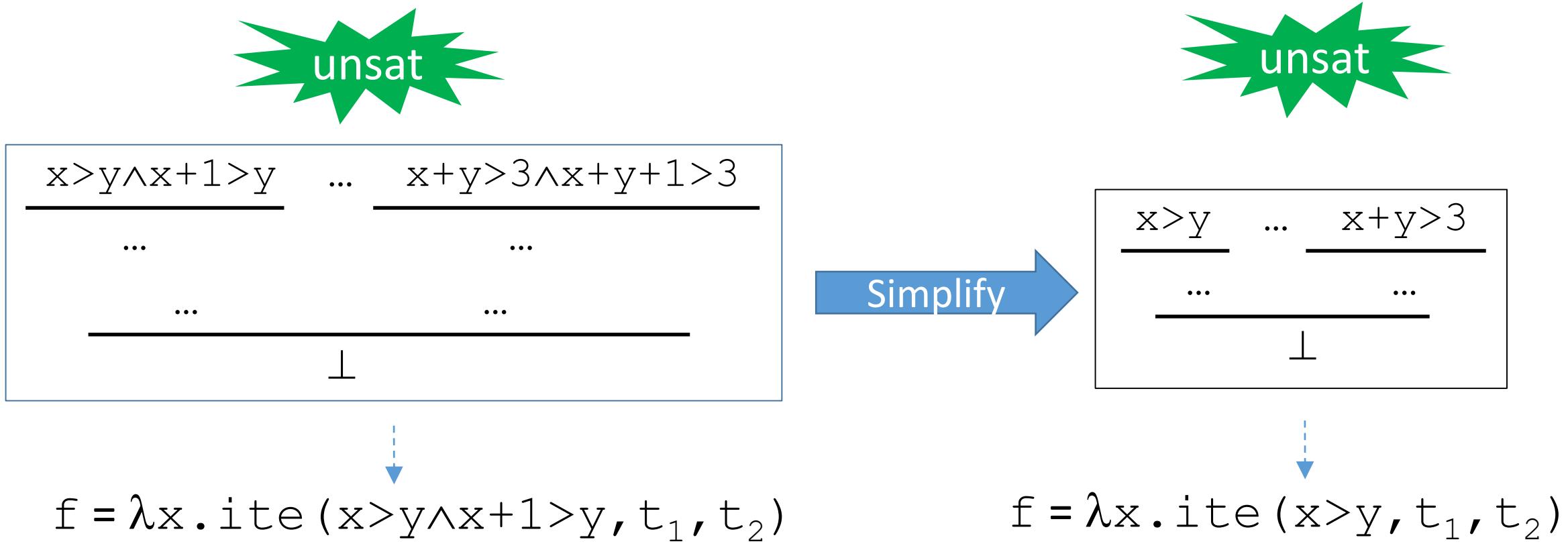
Shorter Solutions via Proof Compression



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

When using counterexample-guided instantiation

Shorter Solutions via Proof Compression



What *proof* techniques are relevant for solution minimization?

What if conjecture is *Partially Single Invocation*?

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

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

What if conjecture is *Partially Single Invocation*?

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

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

What if conjecture is *Partially Single Invocation*?

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

Partition into...

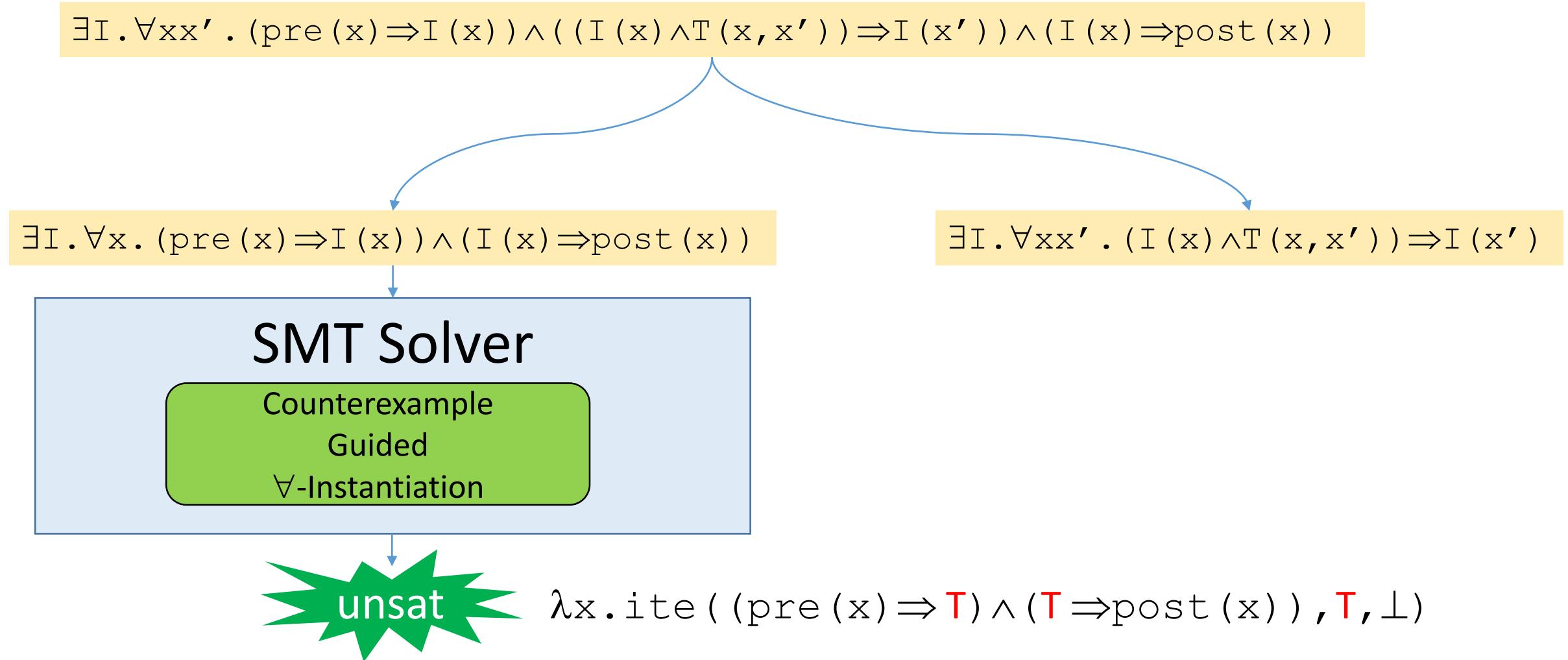
$$\exists I. \forall x. (pre(x) \Rightarrow I(x)) \wedge (I(x) \Rightarrow post(x))$$


Single-invocation portion

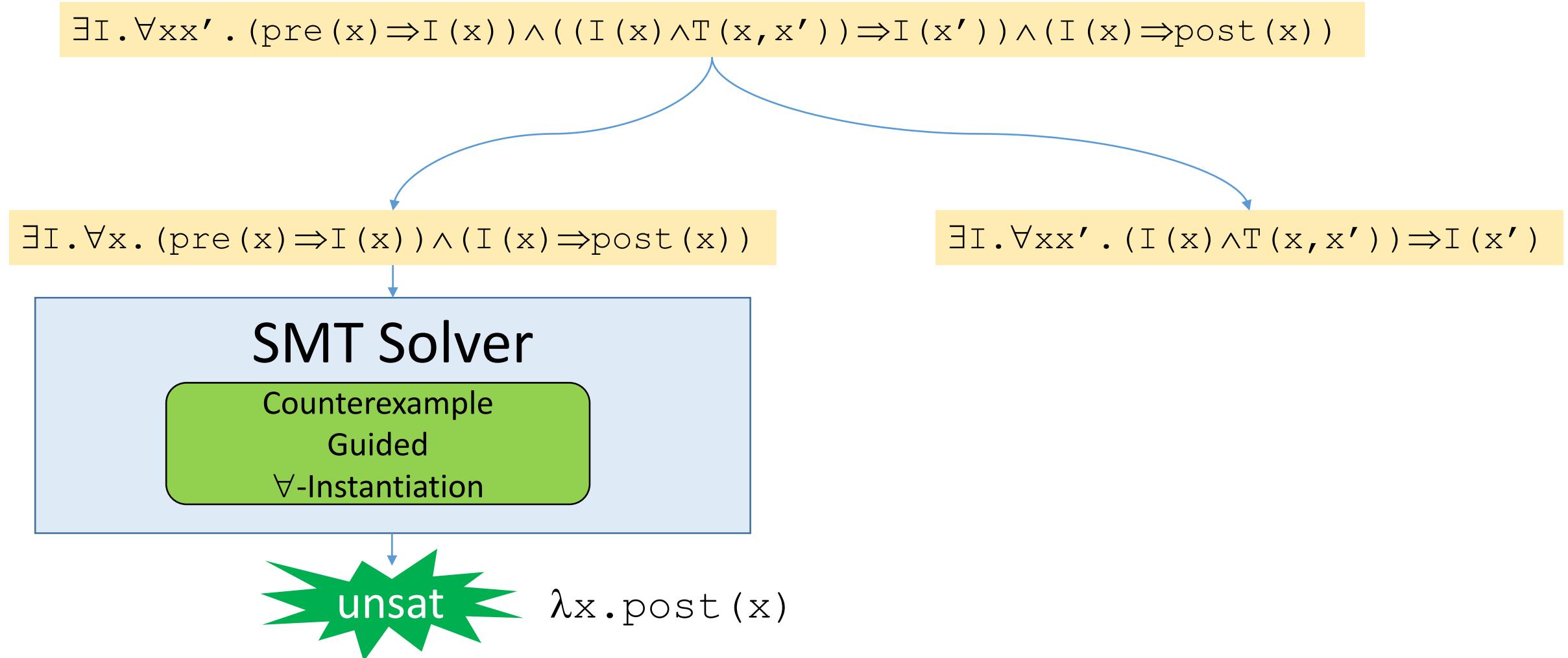
$$\exists I. \forall xx'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$$


Non-single-invocation portion

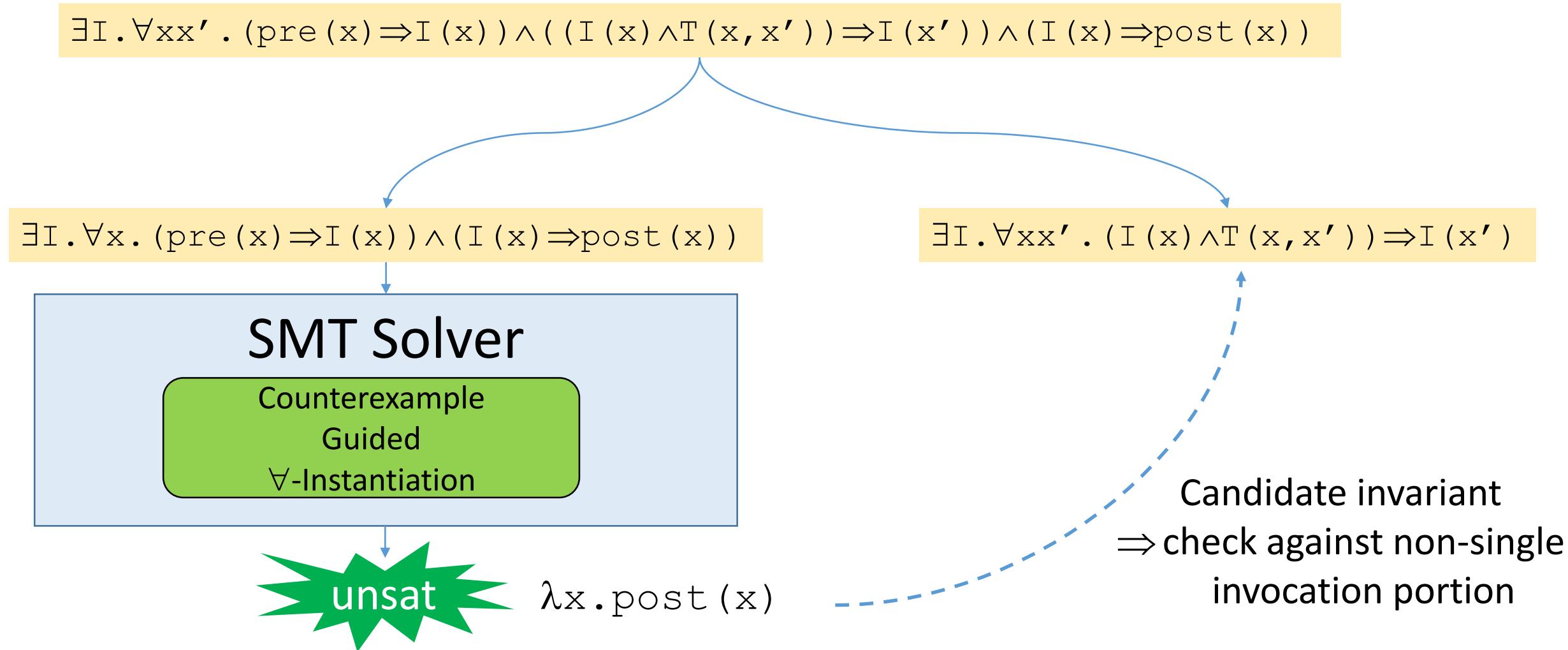
What if conjecture is *Partially Single Invocation*?



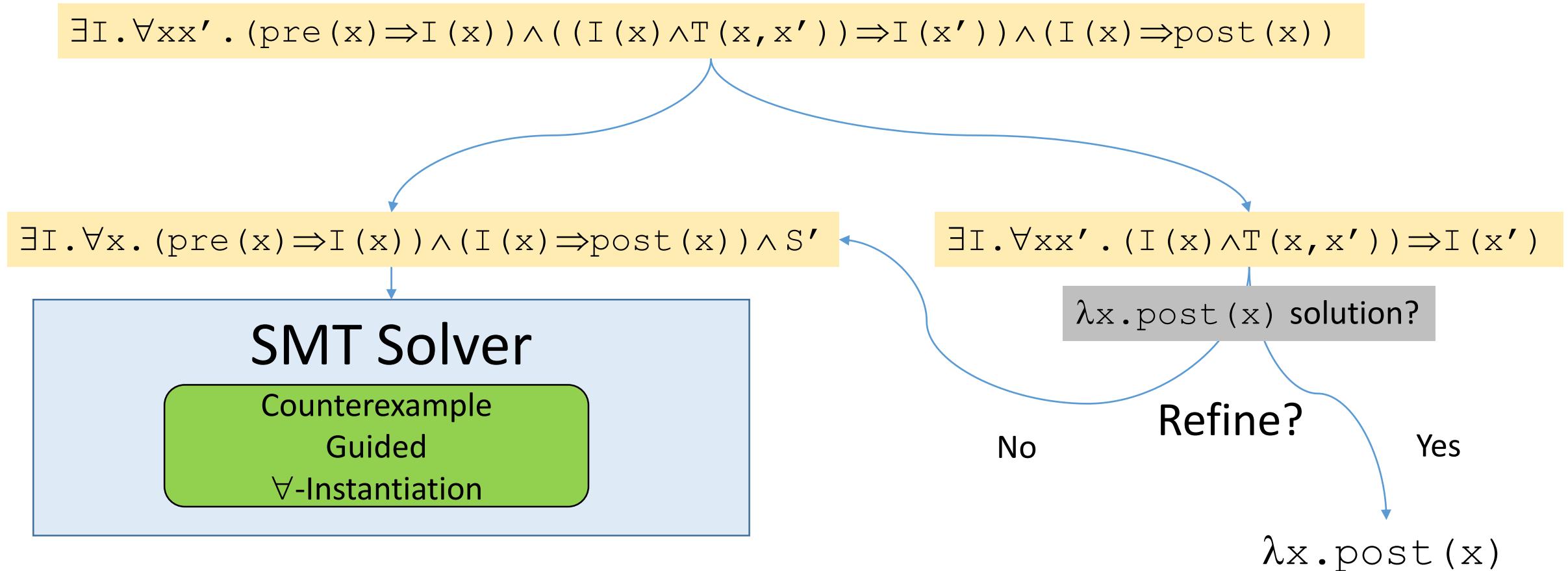
What if conjecture is *Partially Single Invocation*?



What if conjecture is *Partially Single Invocation*?



What if conjecture is *Partially Single Invocation*?



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

What if conjecture is *Partially Single Invocation*?

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

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

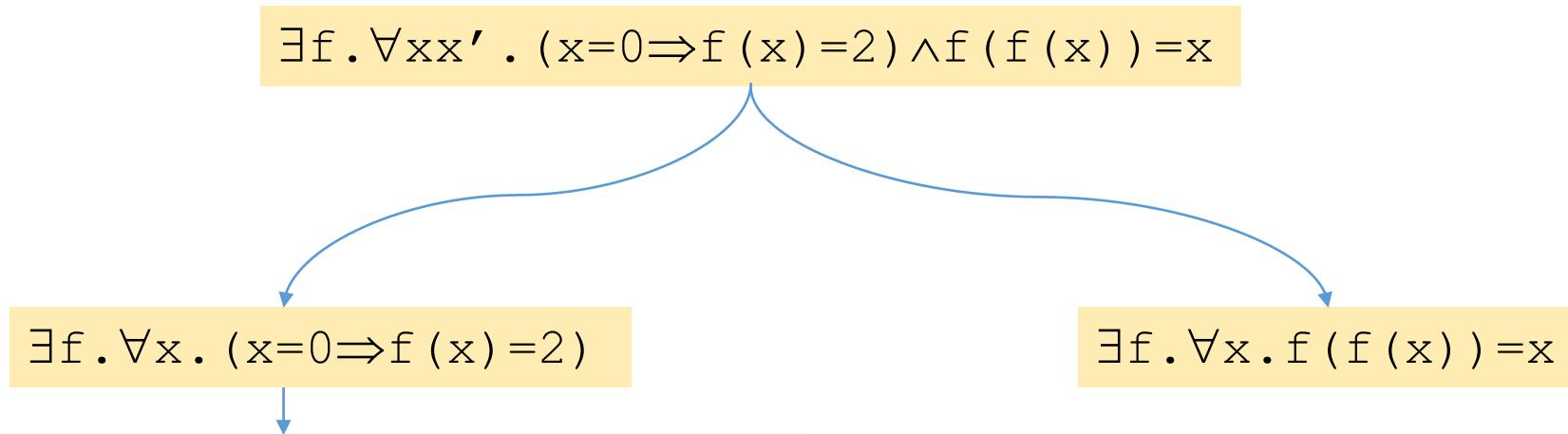
$$\exists I. \forall xx'. (I(x) \wedge T(x, x')) \Rightarrow I(x')$$

SMT Solver

Counterexample
Guided
 \forall -Instantiation

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

What if conjecture is *Partially Single Invocation*?

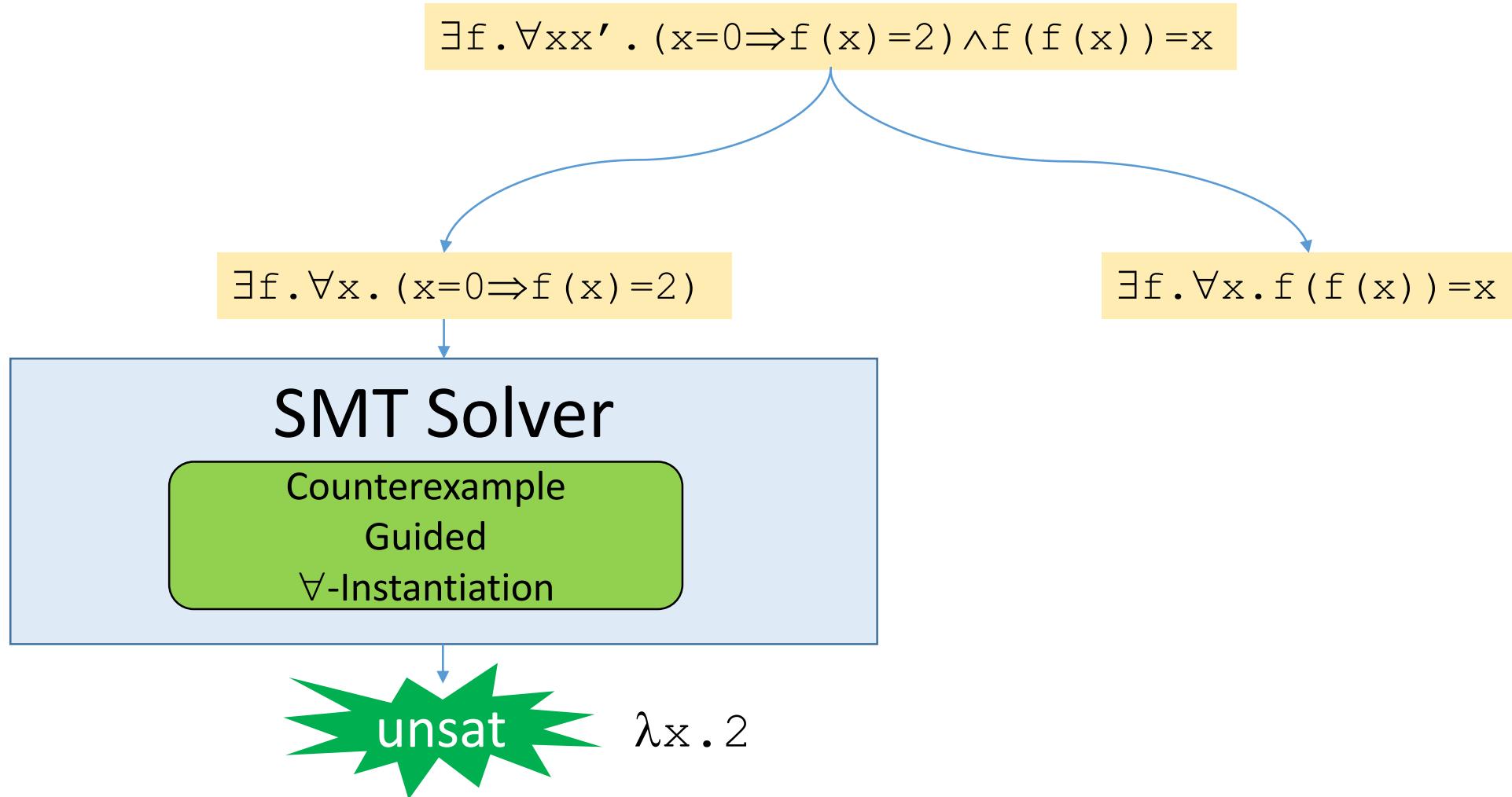


SMT Solver

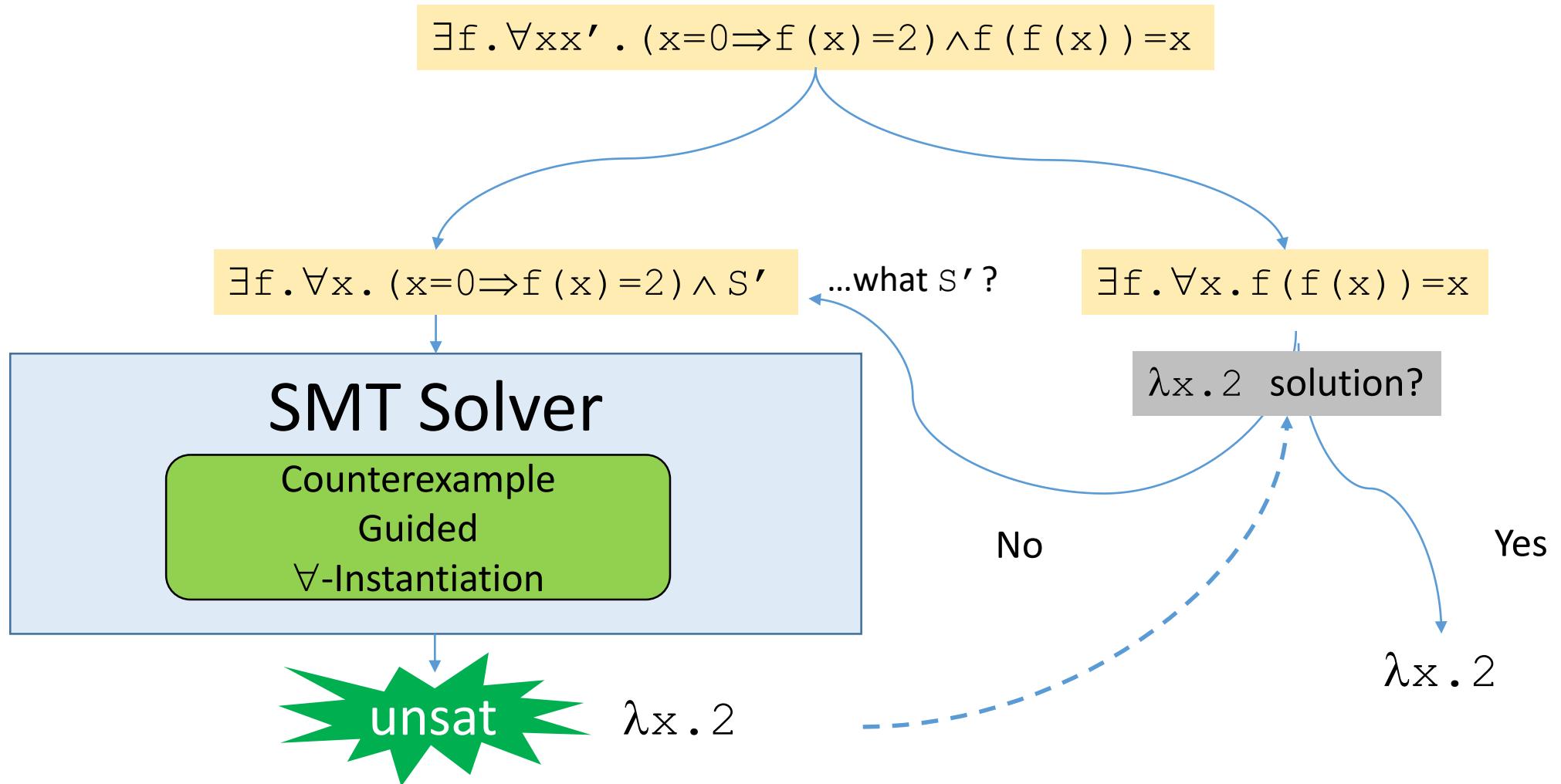
Counterexample
Guided
 \forall -Instantiation

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

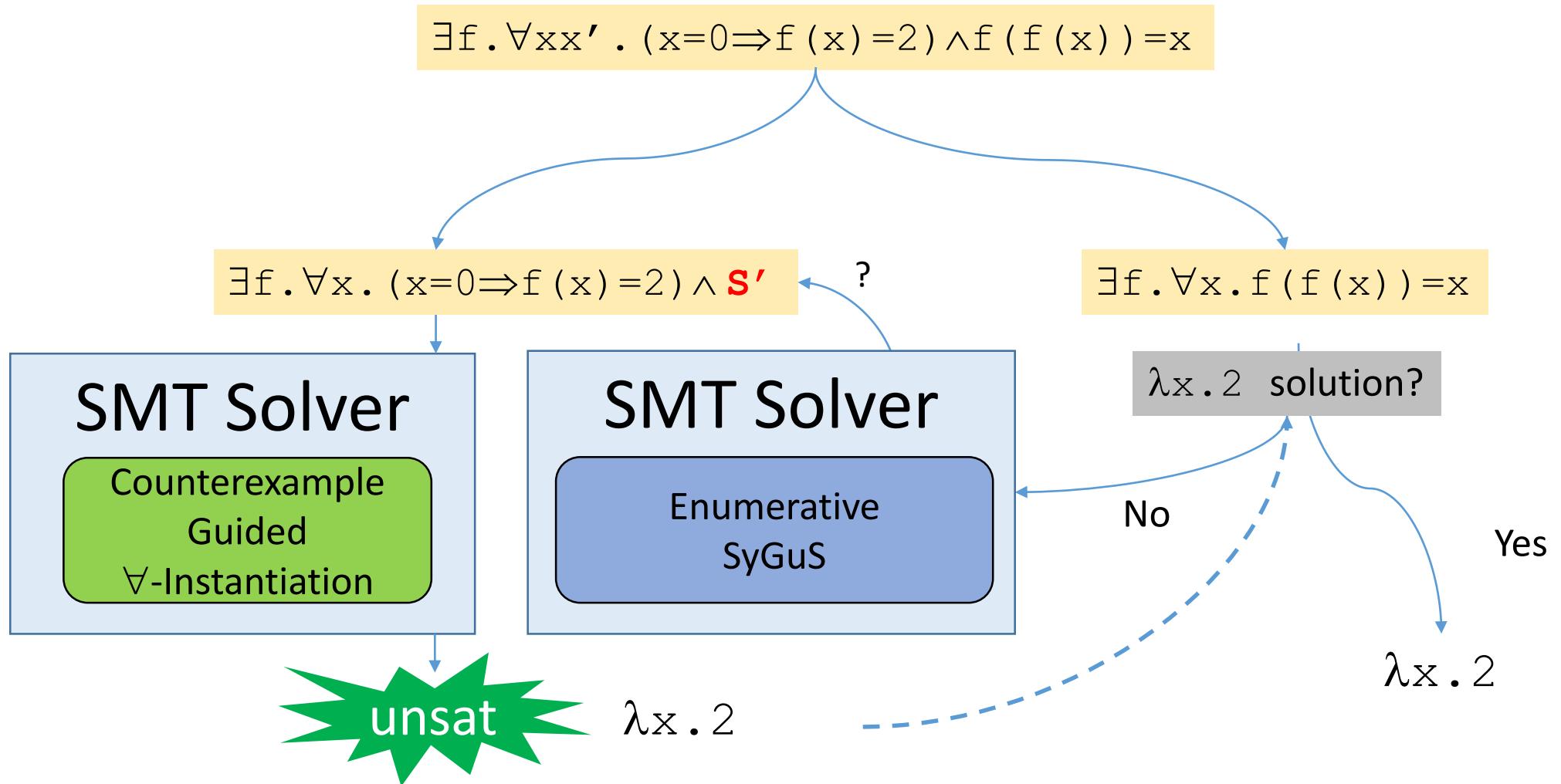
What if conjecture is *Partially Single Invocation*?



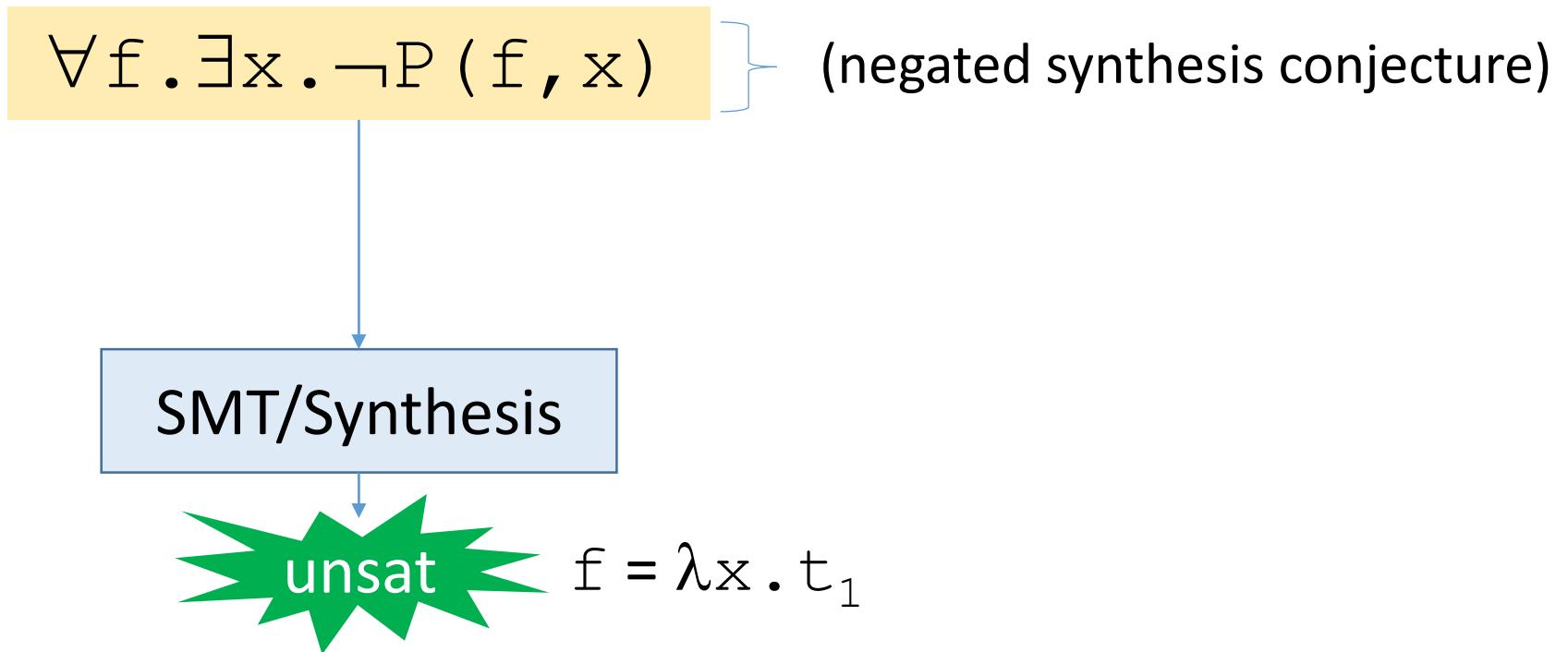
What if conjecture is *Partially Single Invocation*?



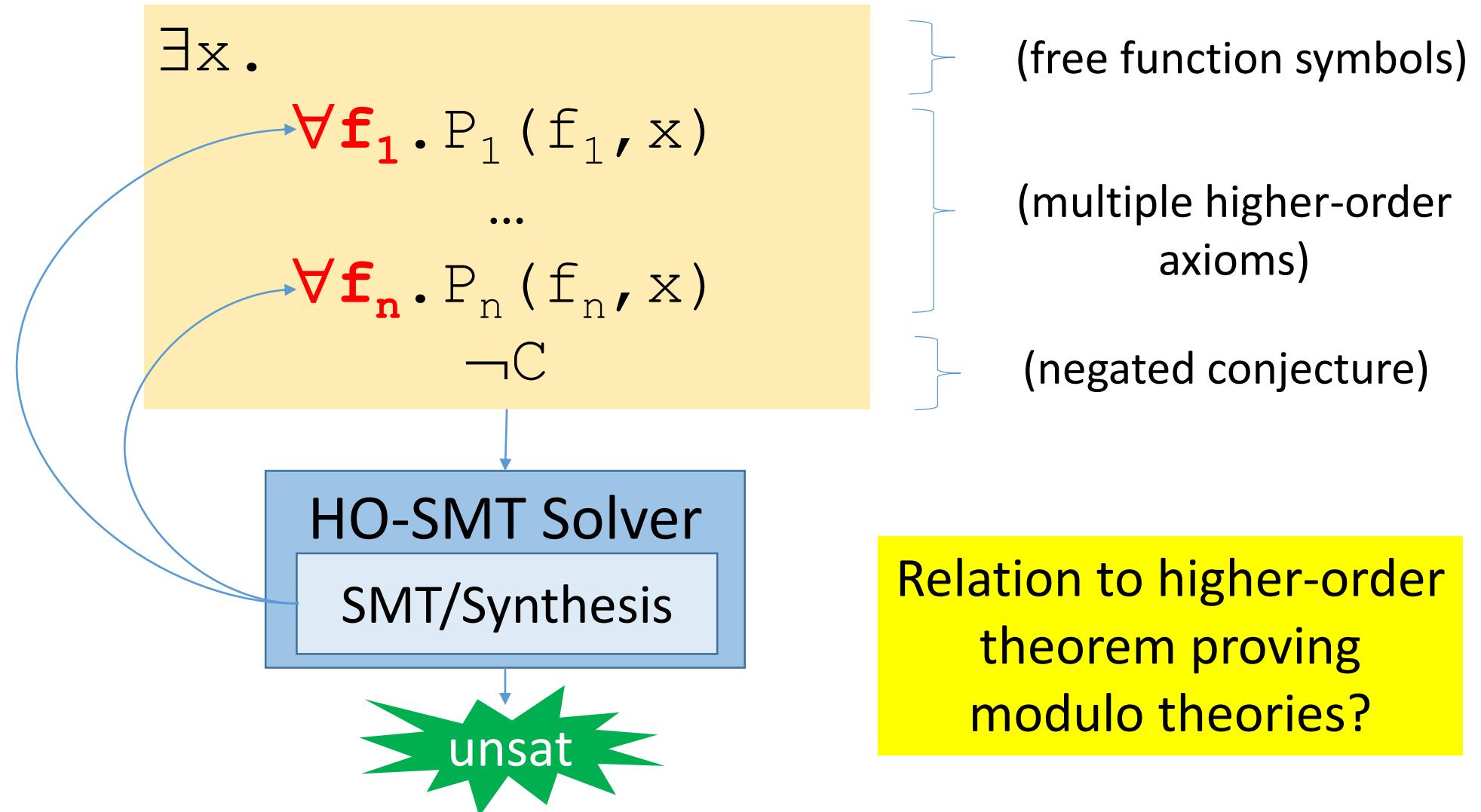
What if conjecture is *Partially Single Invocation*?



Synthesis for Higher-Order Theorem Proving?



Synthesis for Higher-Order Theorem Proving?



Fast and Slow Synthesis Procedures in SMT

