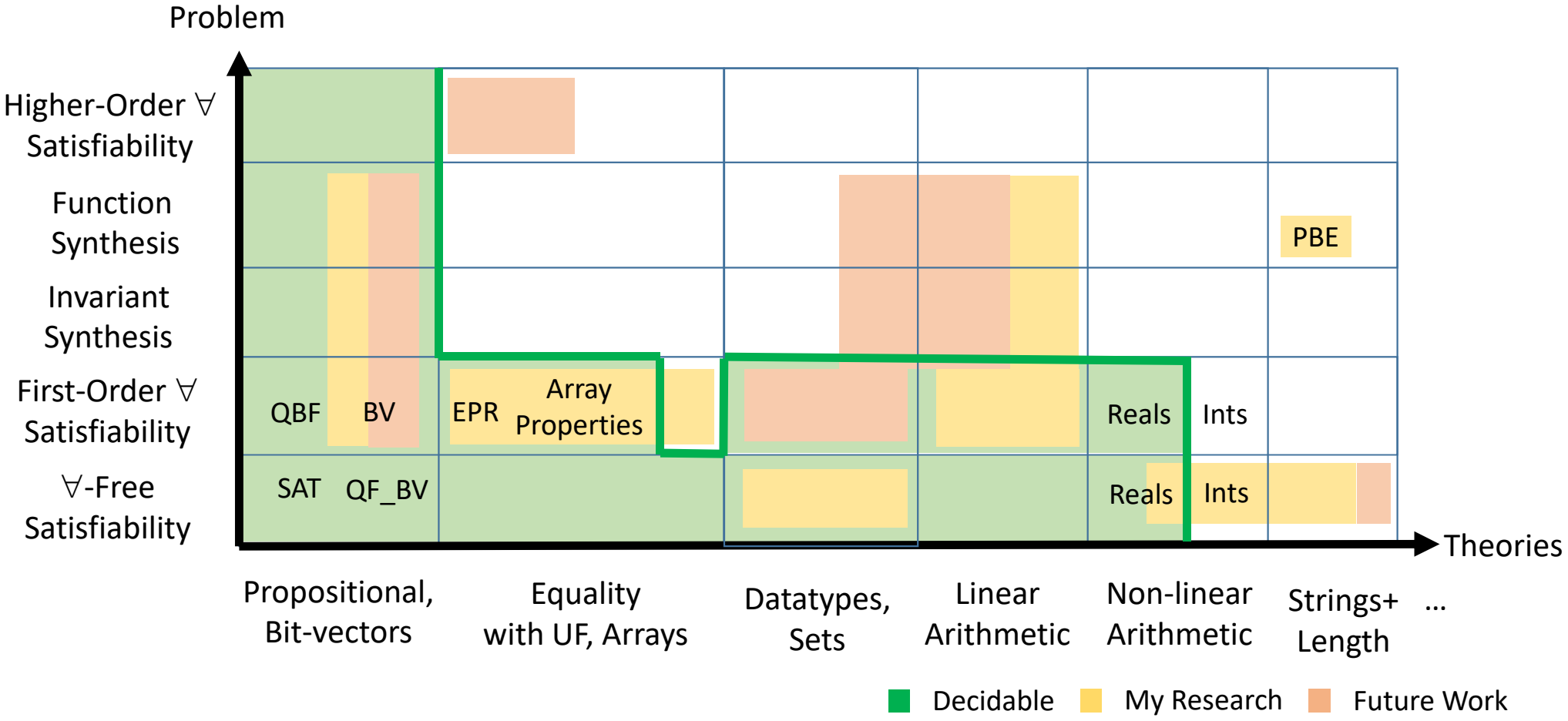
Fast and Slow Synthesis Procedures in SMT

Andrew Reynolds

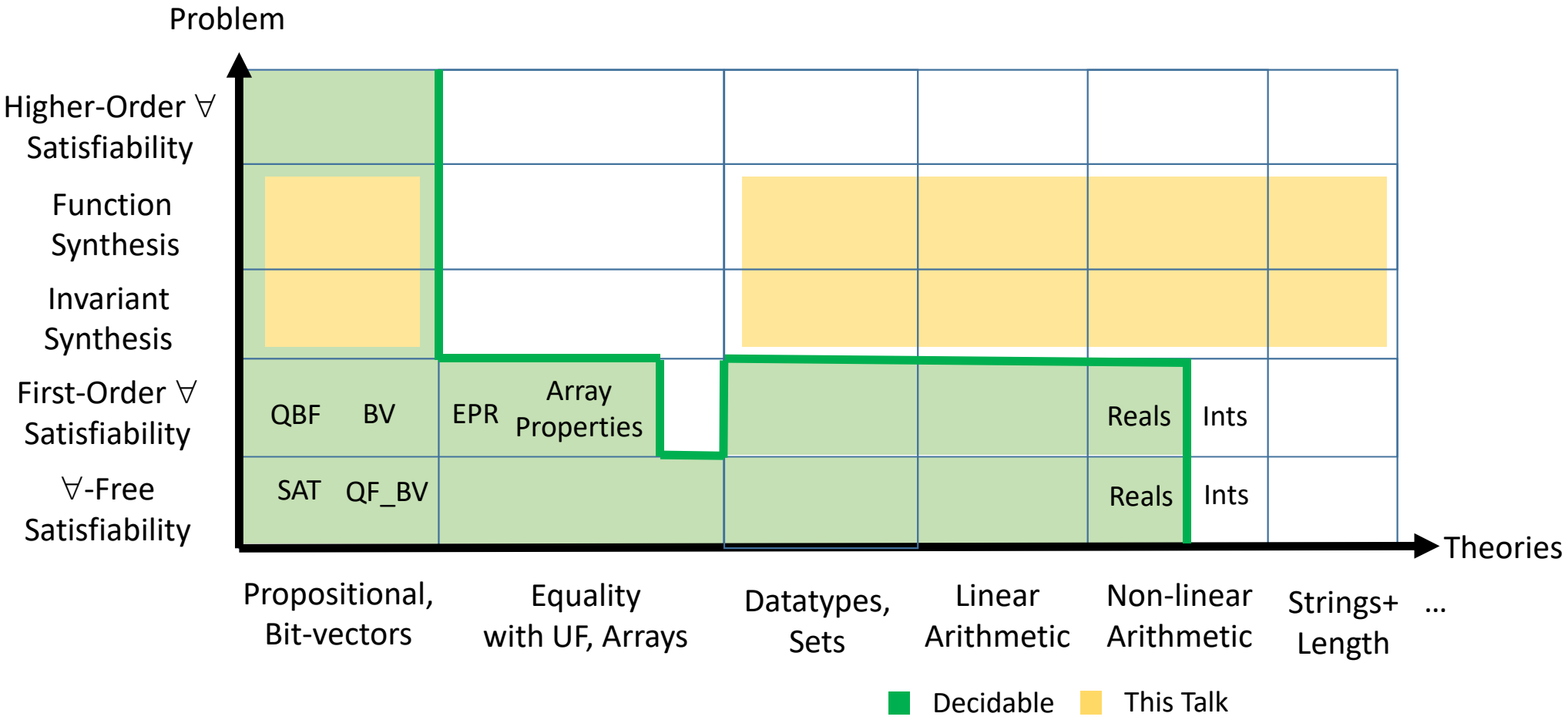
Dagstuhl Seminar: Deduction Beyond First-Order Logic

September 14, 2017

My Research in SMT



This Talk: Synthesis for (Interpreted) Theories



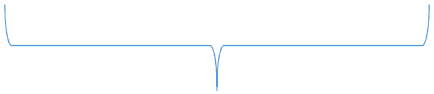
Synthesis

- 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

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



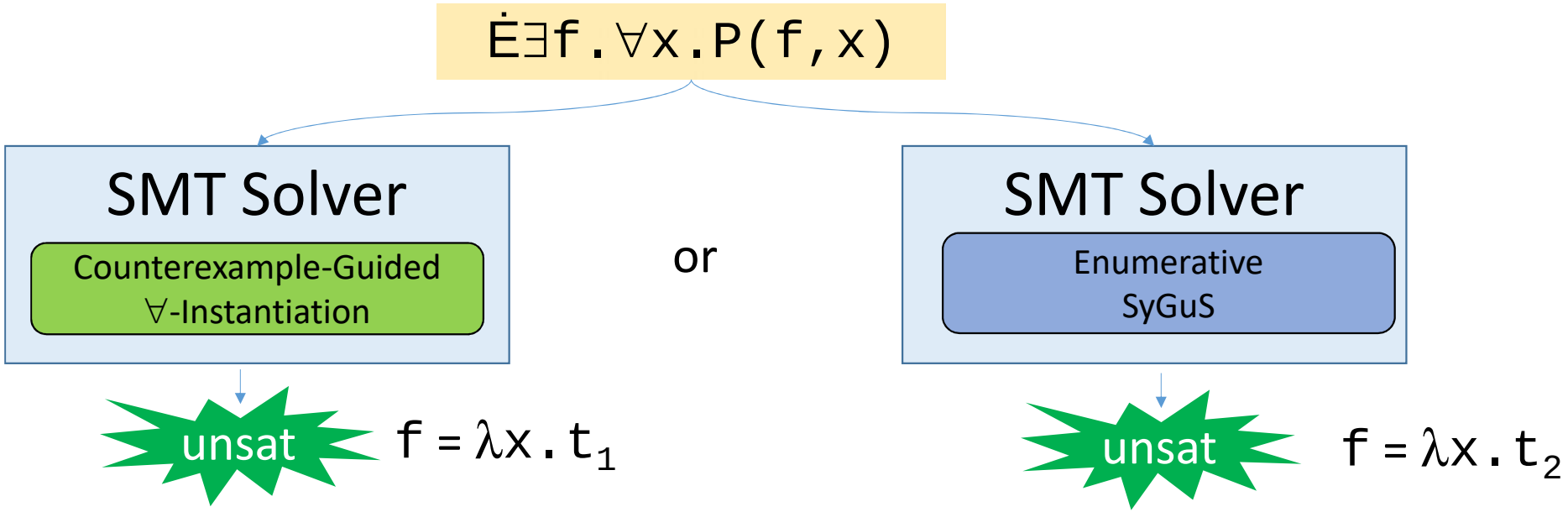
There exists a function f for which property P holds for all x

Refutation-Based Synthesis in SMT

$$\dot{E}\exists f.\forall x.P(f,x)$$

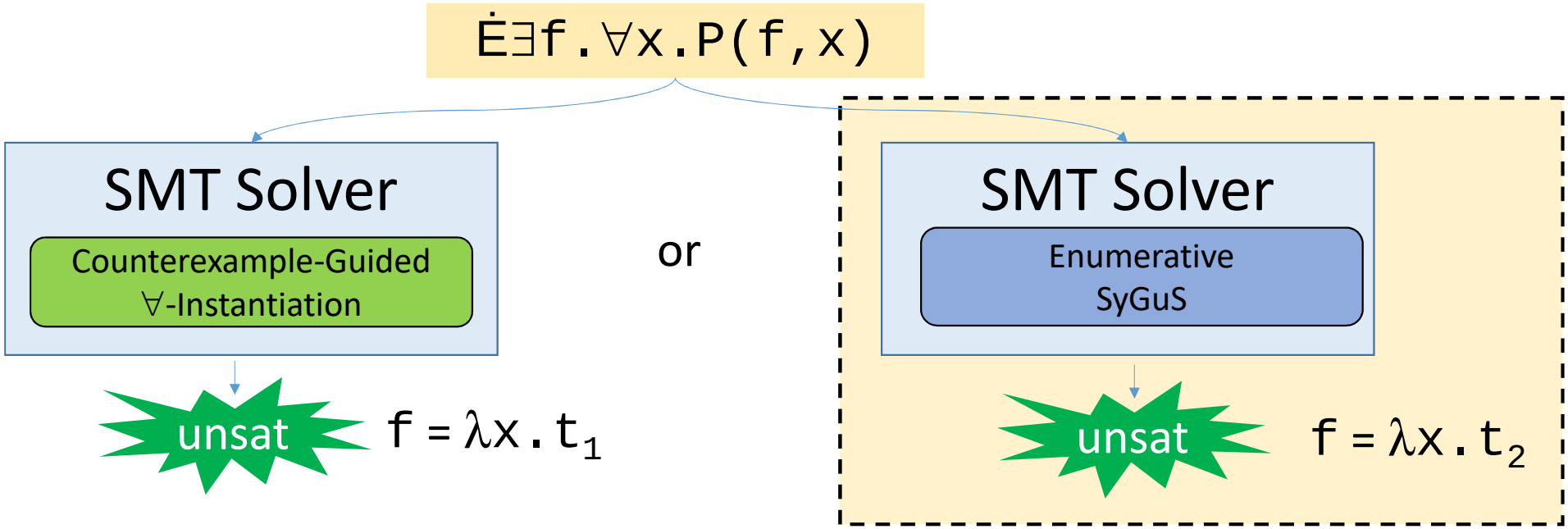
(**negated** synthesis conjecture)

Refutation-Based Synthesis in SMT



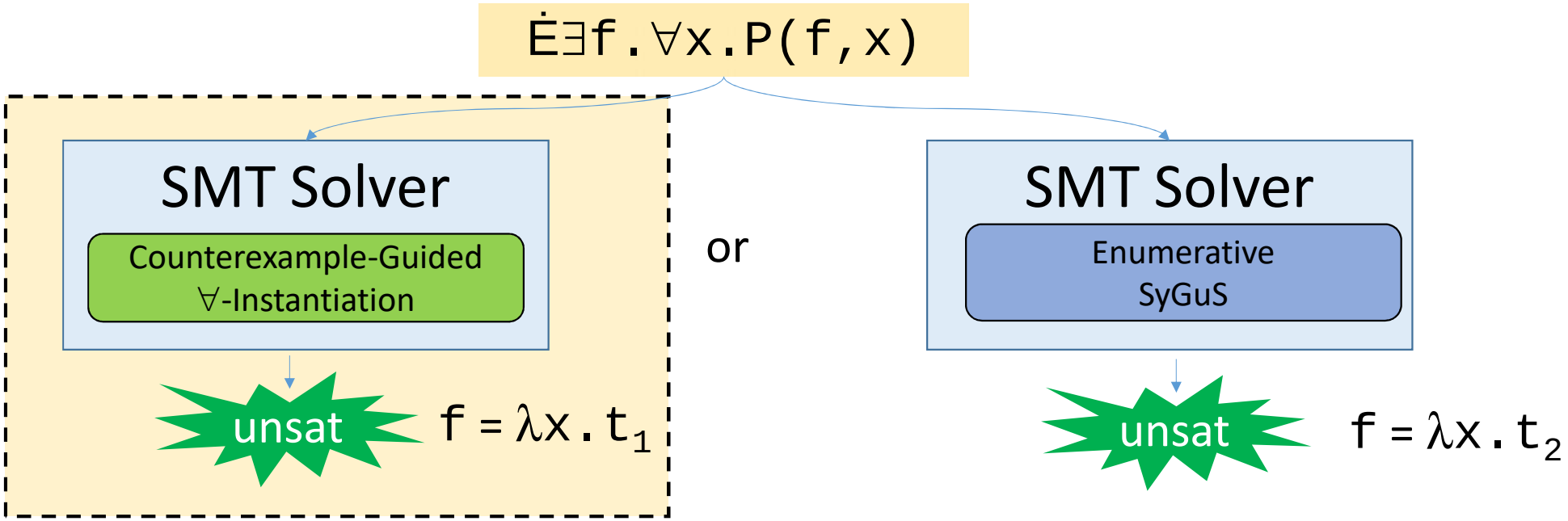
- Two approaches for refutation-based synthesis in SMT solvers **[Reynolds et al CAV2015]**

Refutation-Based Synthesis in SMT



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

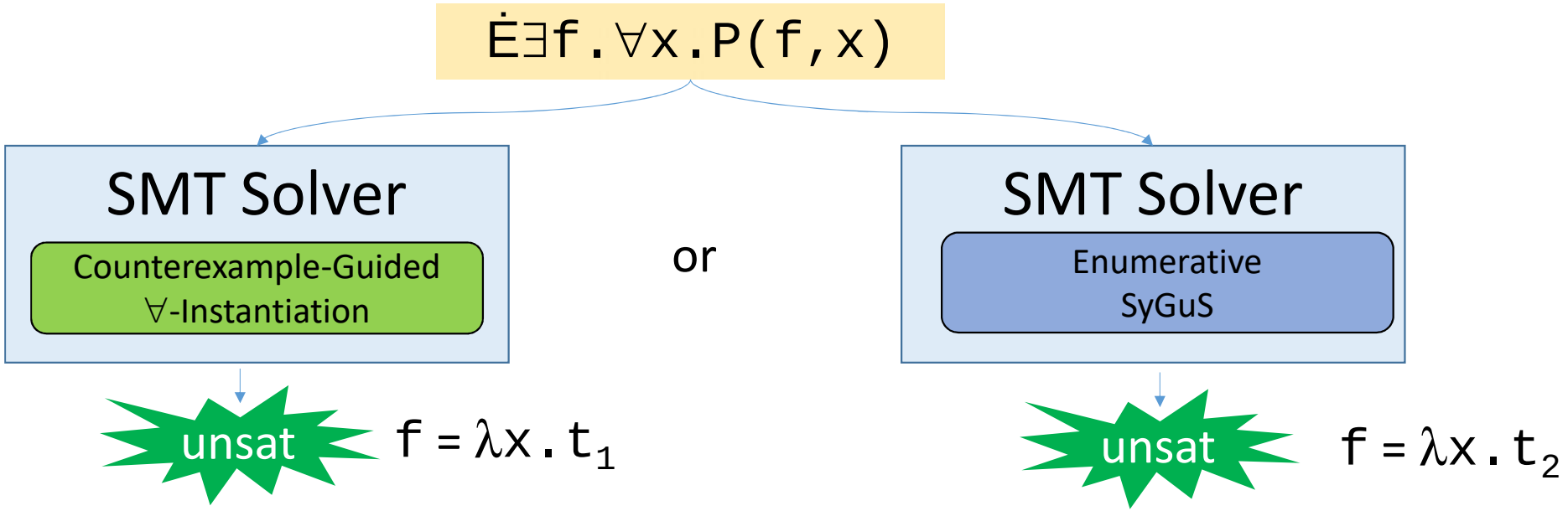
Refutation-Based Synthesis in SMT



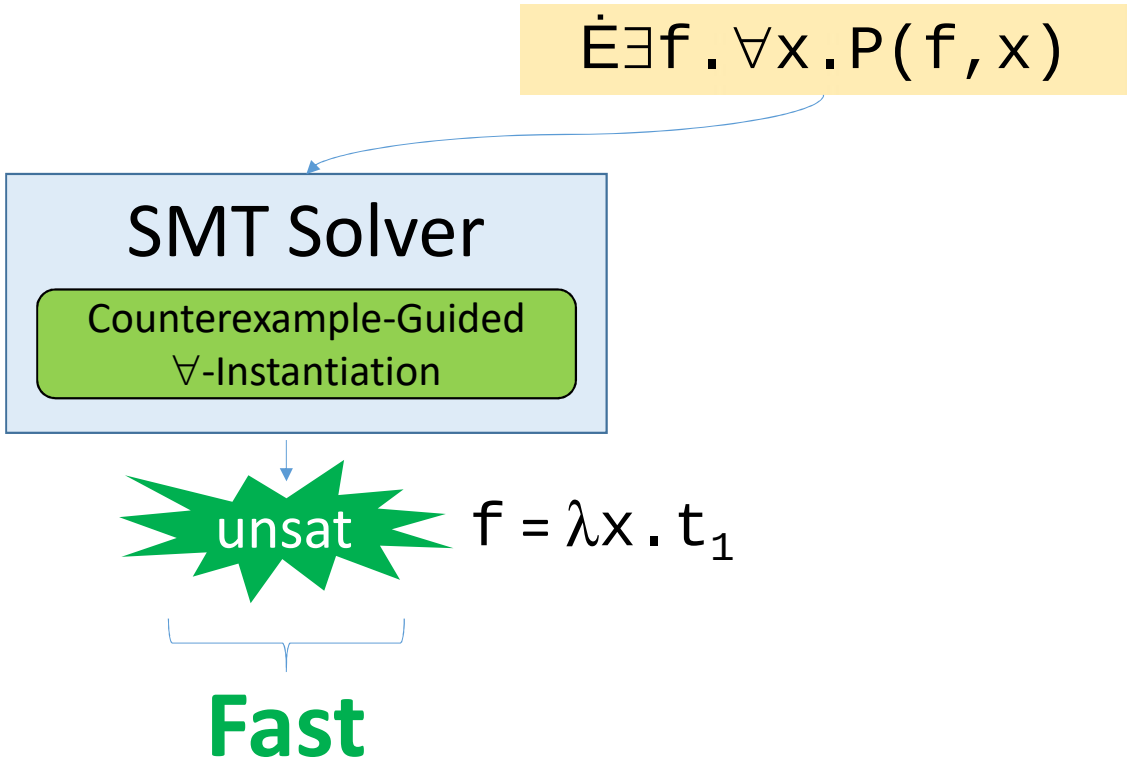
⇒ Based on first-order quantifier instantiation in SMT

[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*:

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

"**f(x,y)** is the maximum of x and y"

Fast Synthesis in SMT Solvers

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

$\mathtt{Int} \times \mathtt{Int} \rightarrow \mathtt{Int}$

All occurrence of f are in terms of the form $\mathbf{f(x,y)}$

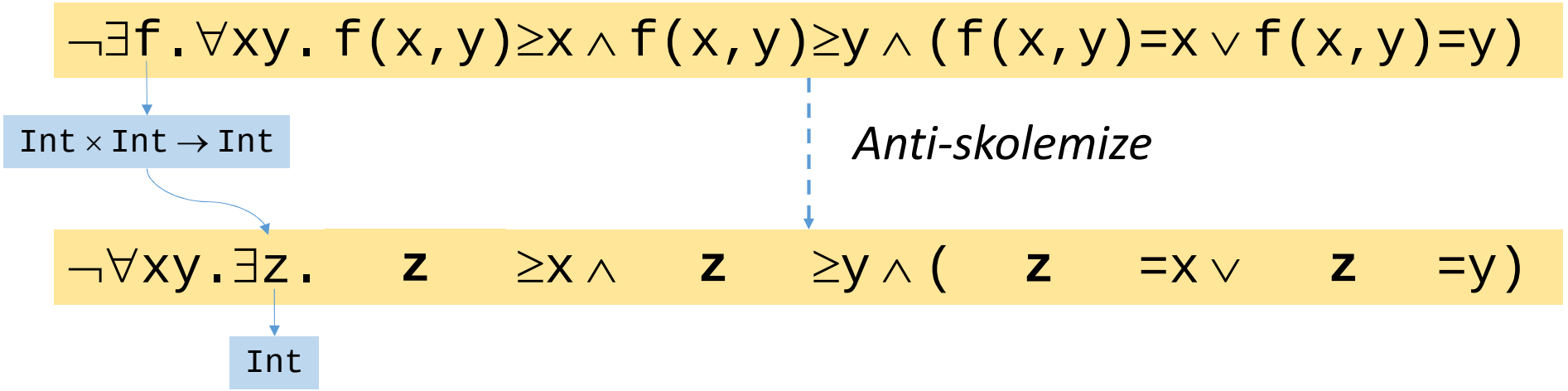
Ĝ “single invocation” synthesis conjectures

Fast Synthesis in SMT Solvers

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

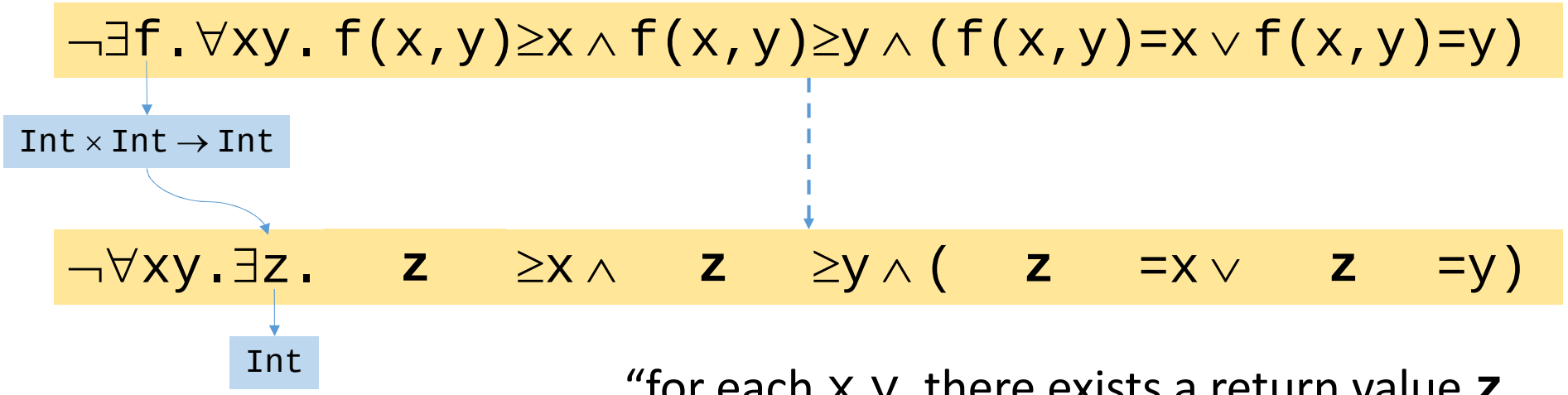
$\texttt{Int} \times \texttt{Int} \rightarrow \texttt{Int}$

Fast Synthesis in SMT Solvers



[Reynolds et al CAV2015]

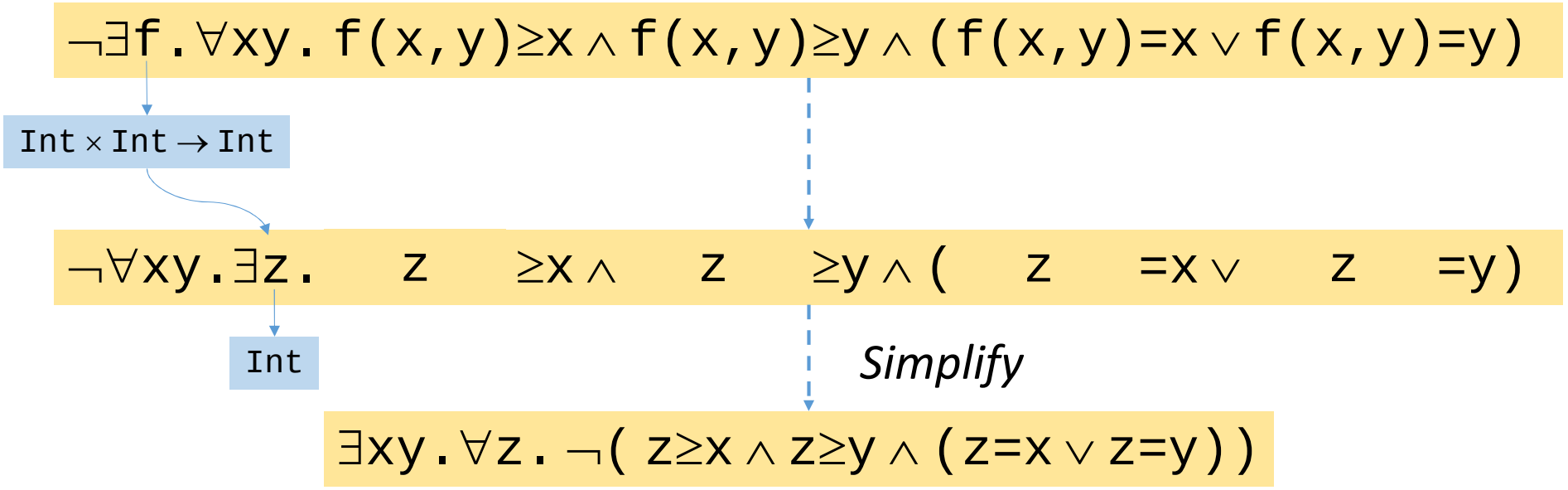
Fast Synthesis in SMT Solvers



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

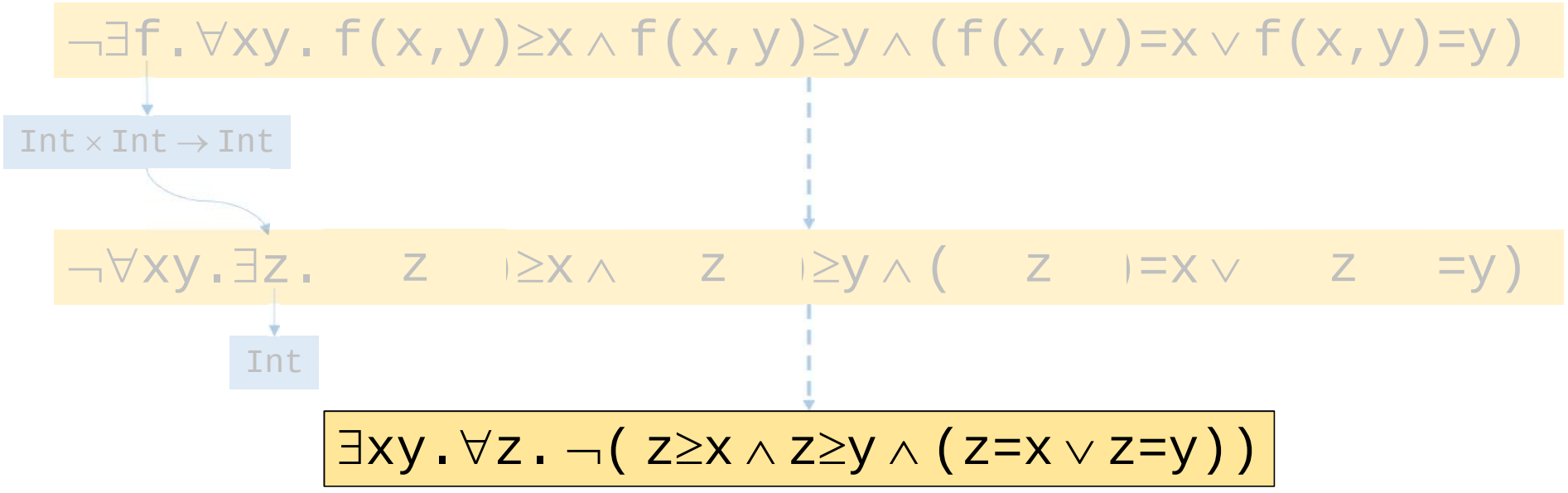
[Reynolds et al CAV2015]

Fast Synthesis in SMT Solvers



[Reynolds et al CAV2015]

Fast Synthesis in SMT Solvers



First-order linear arithmetic Ĝ Solvable by first-order ?-instantiation

[Reynolds et al CAV2015]

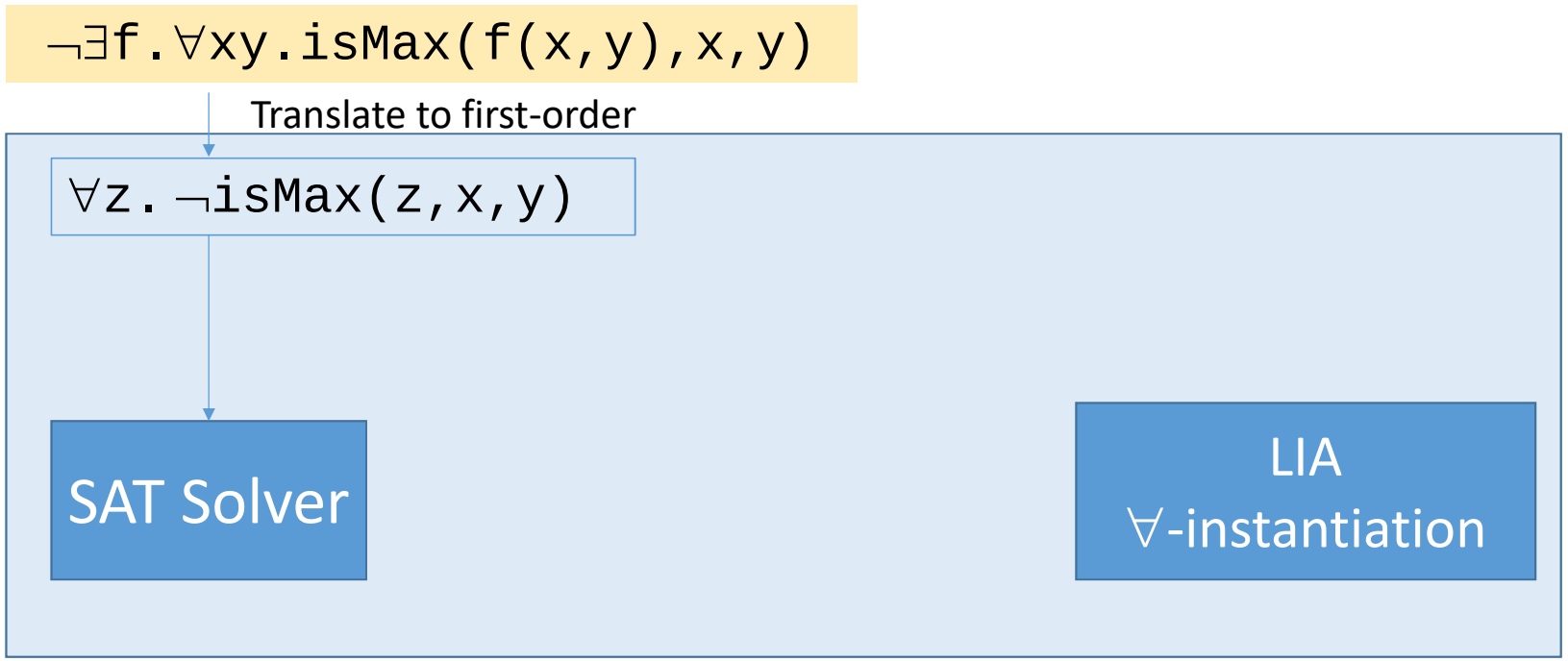
Fast Synthesis in SMT Solvers

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

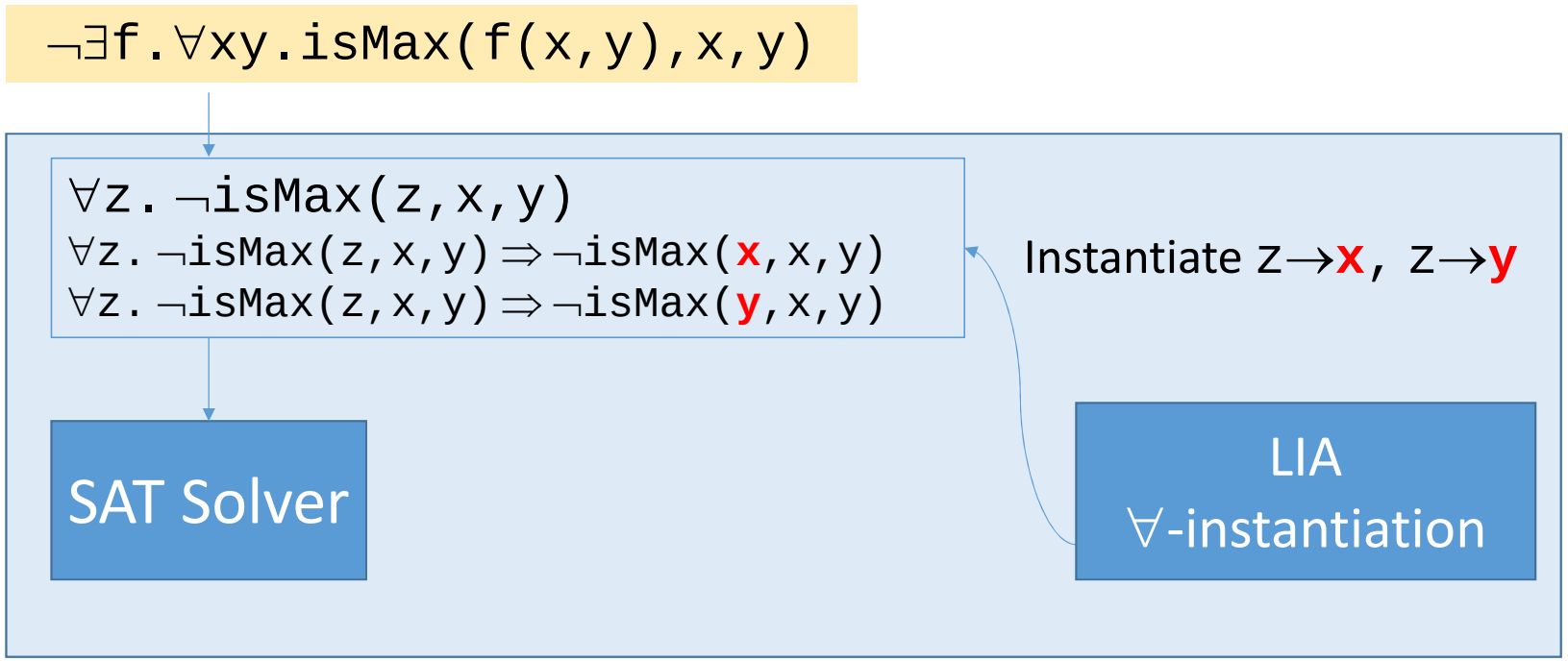
Fast Synthesis in SMT Solvers

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

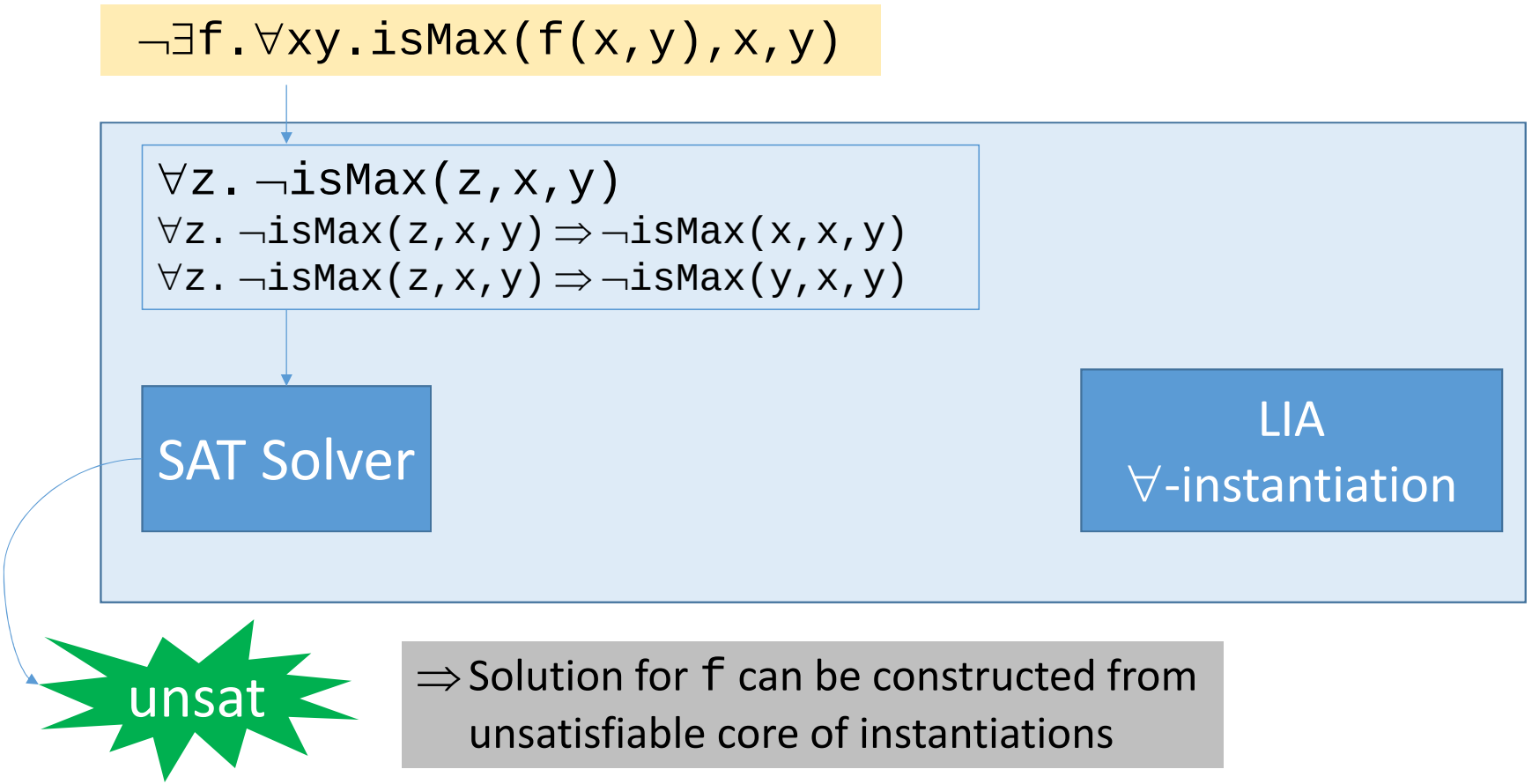
Fast Synthesis in SMT Solvers



Fast Synthesis in SMT Solvers

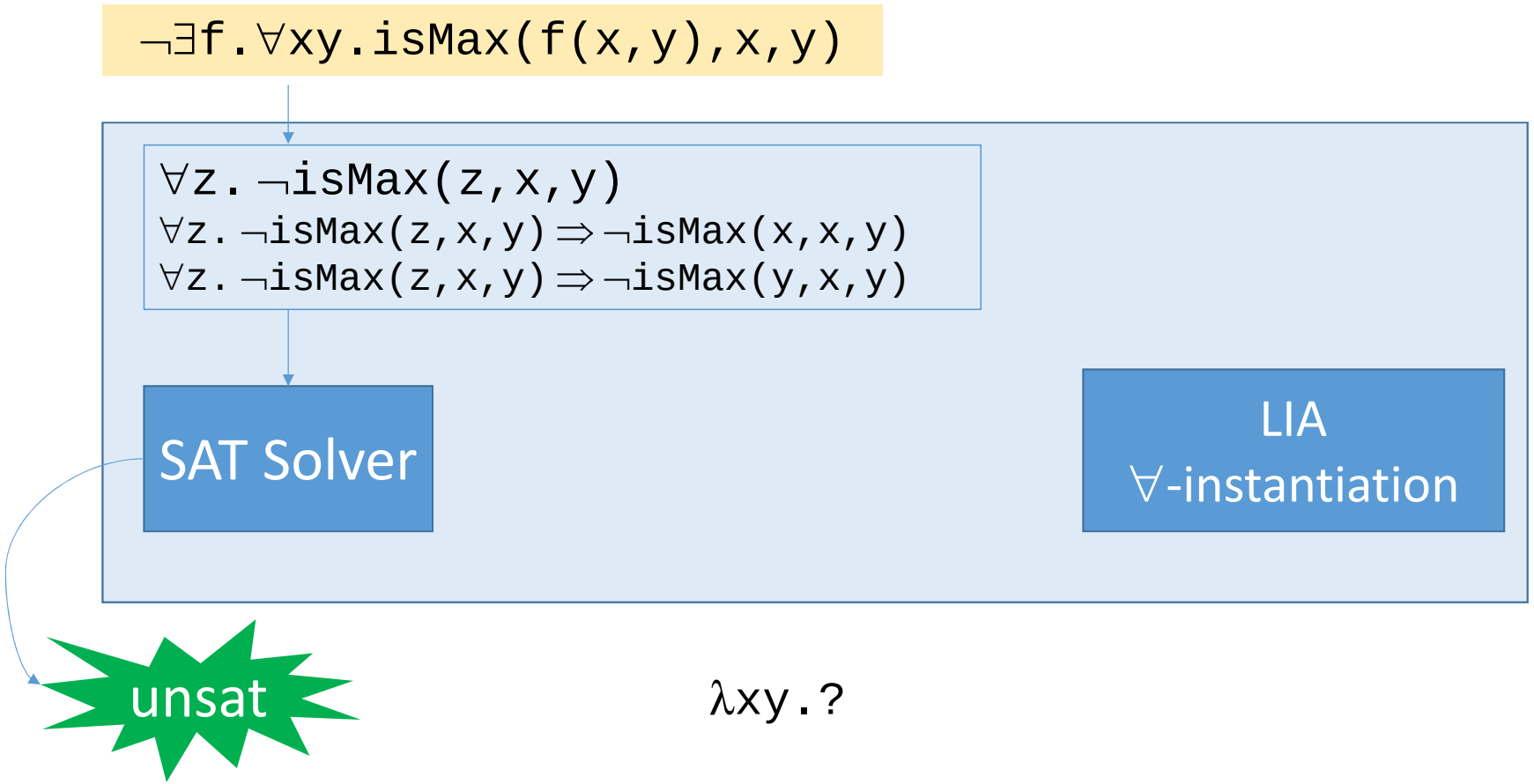


Fast Synthesis in SMT Solvers



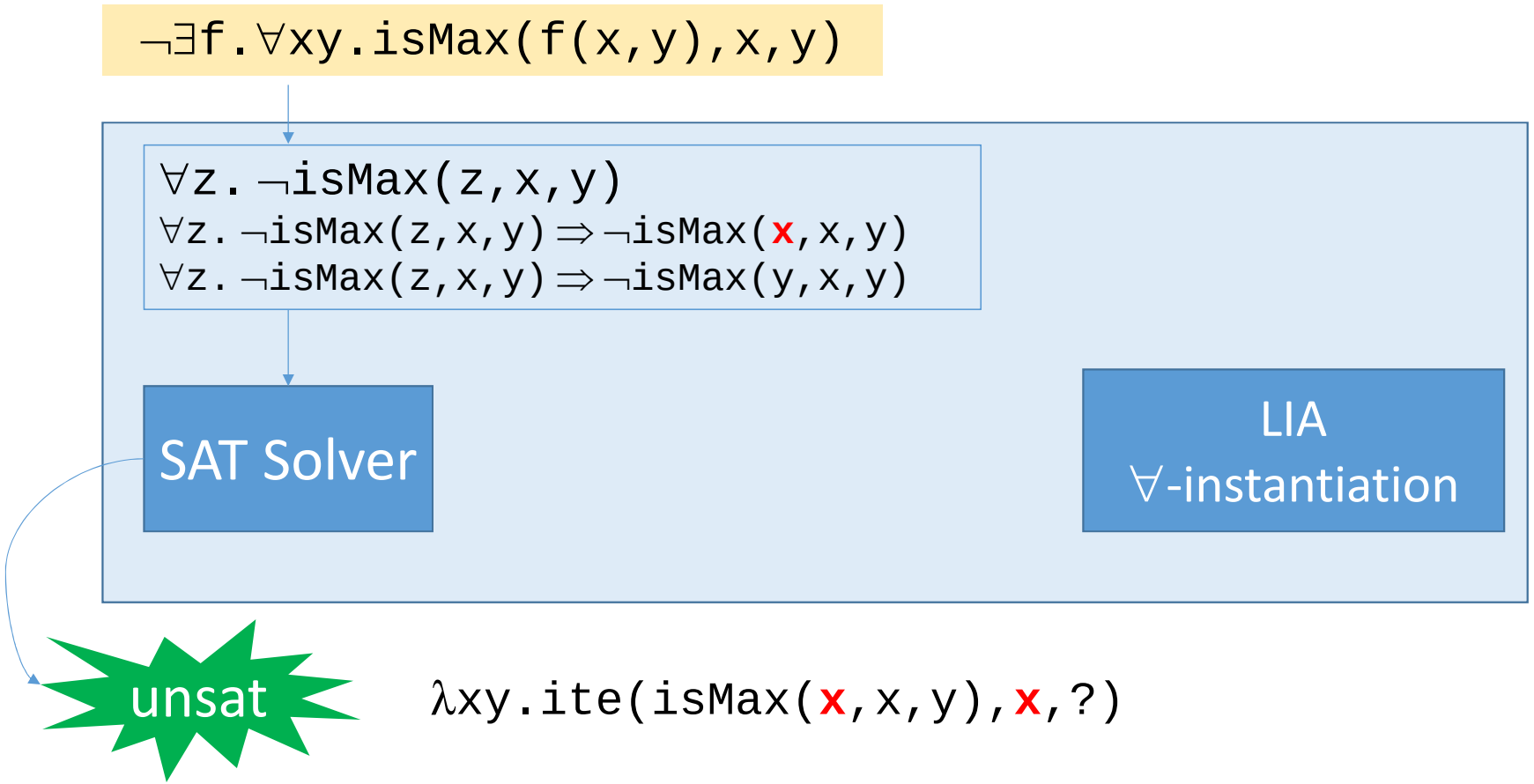
⇒ Solution for f can be constructed from unsatisfiable core of instantiations

Fast Synthesis in SMT Solvers

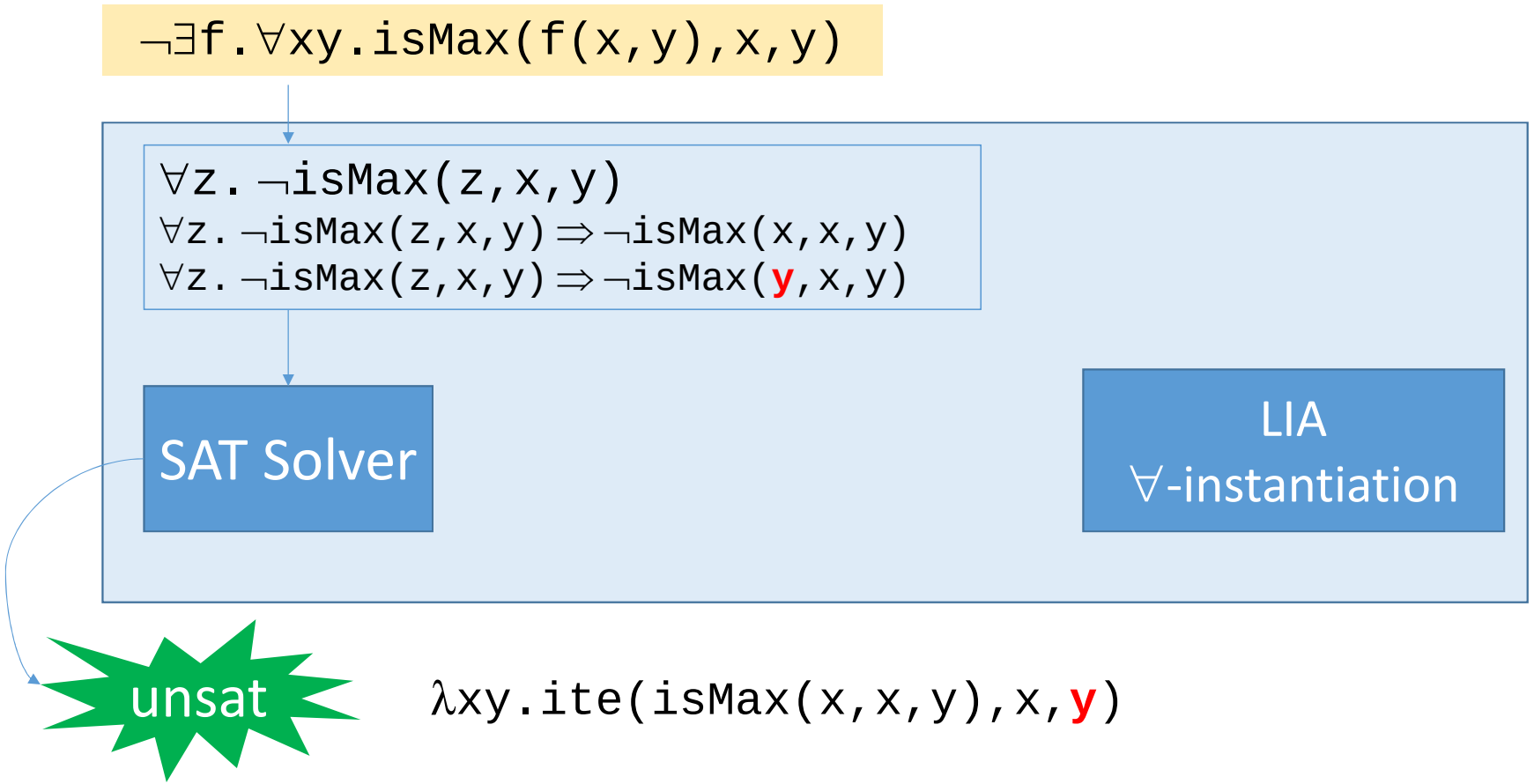


λxy.?

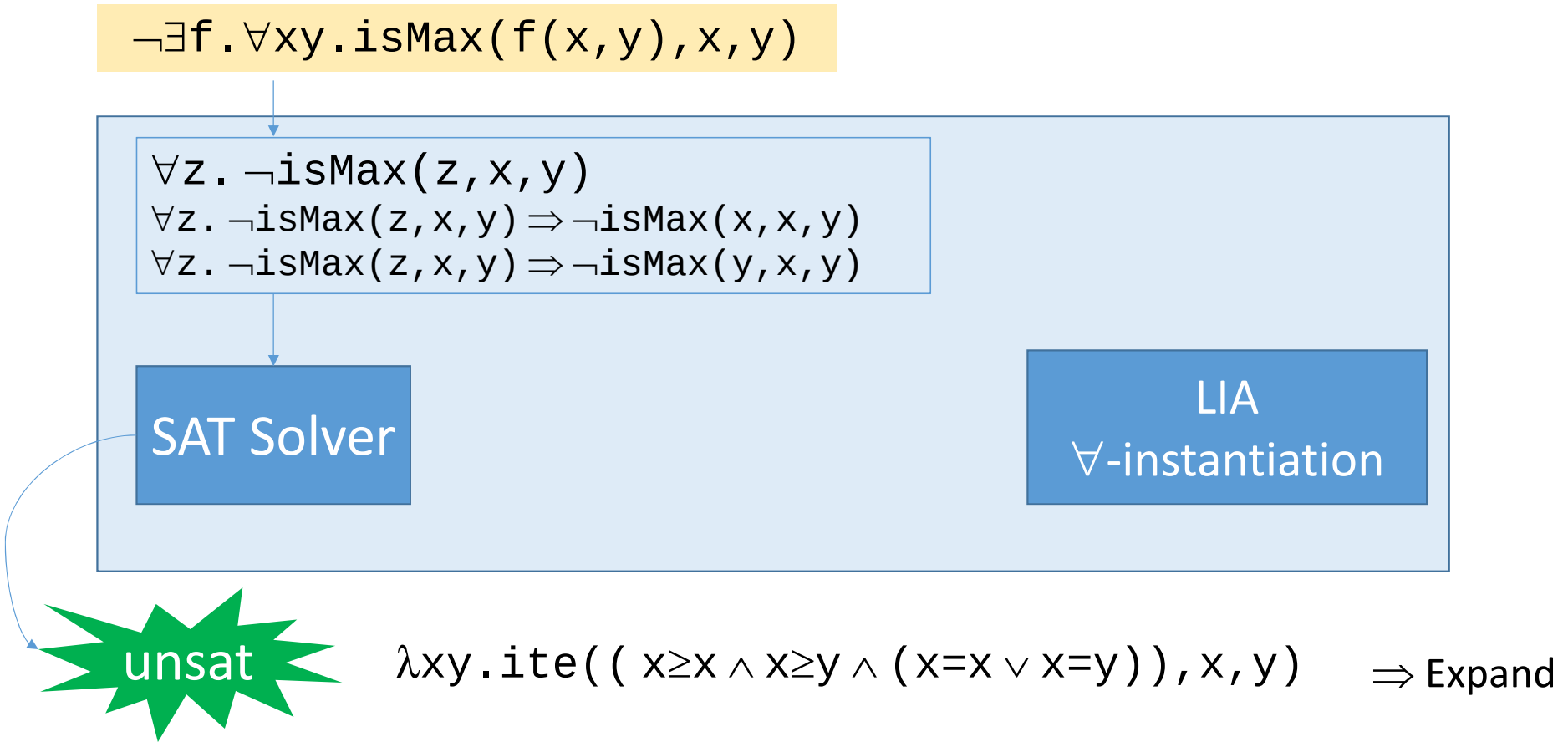
Fast Synthesis in SMT Solvers



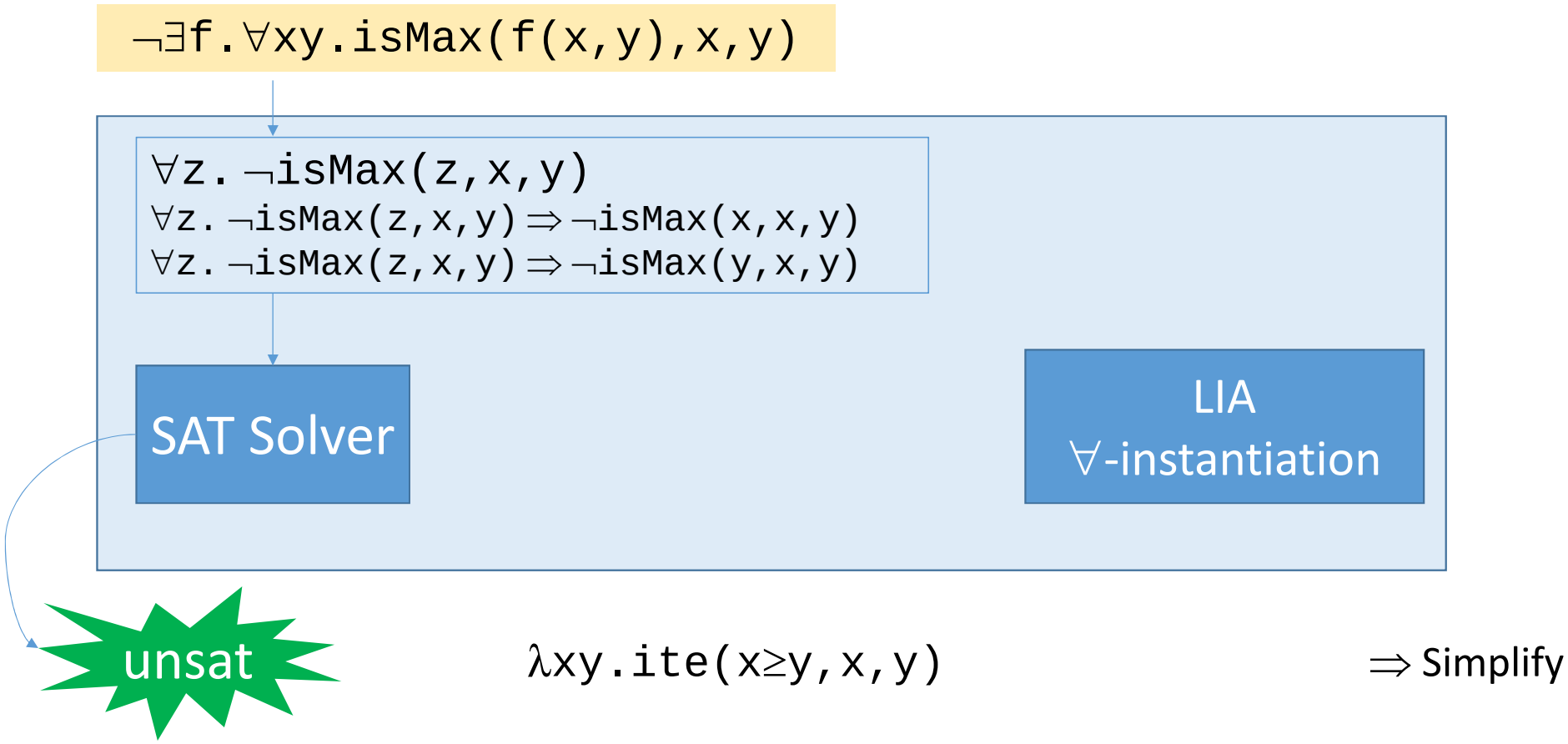
Fast Synthesis in SMT Solvers



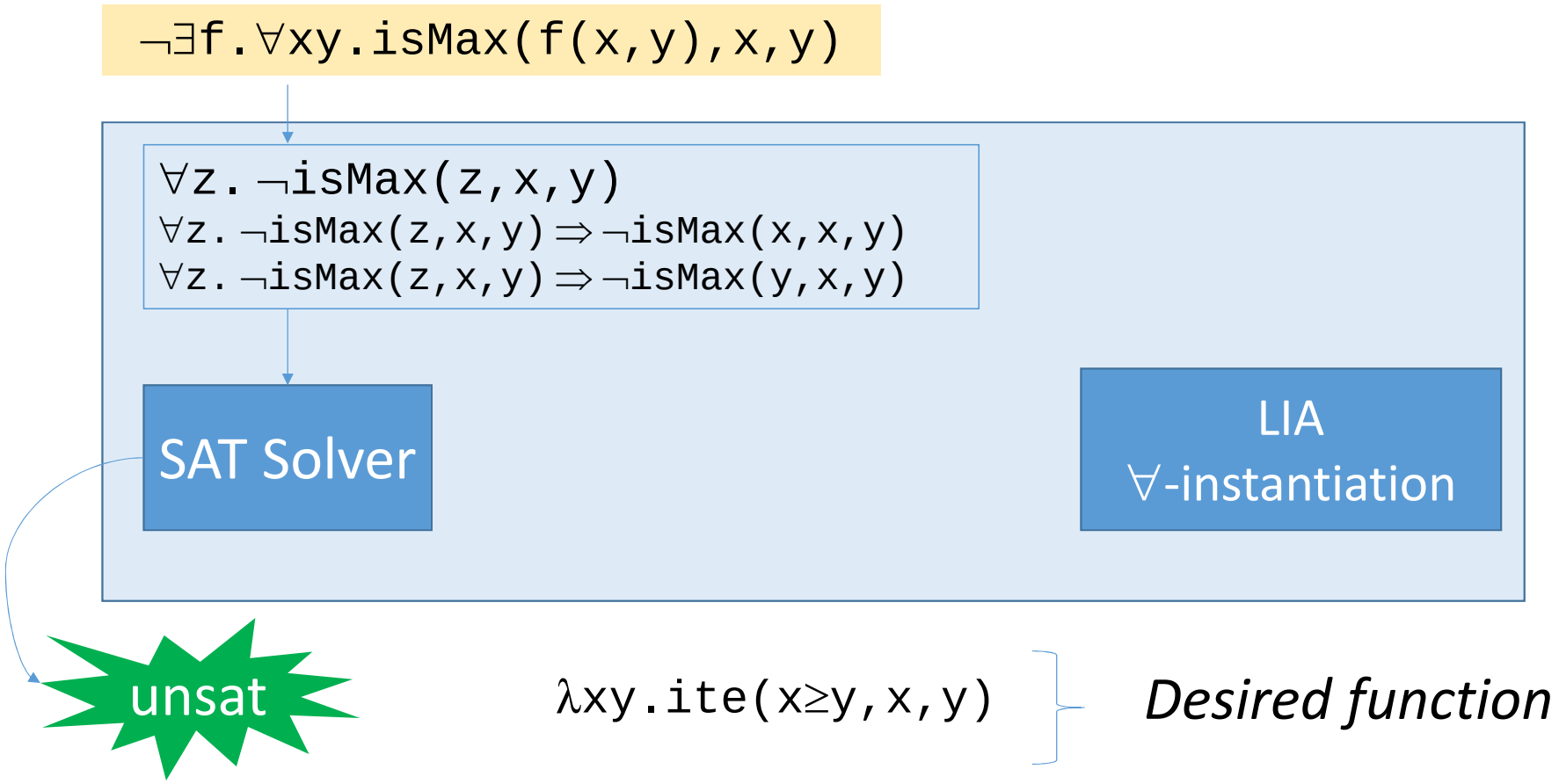
Fast Synthesis in SMT Solvers



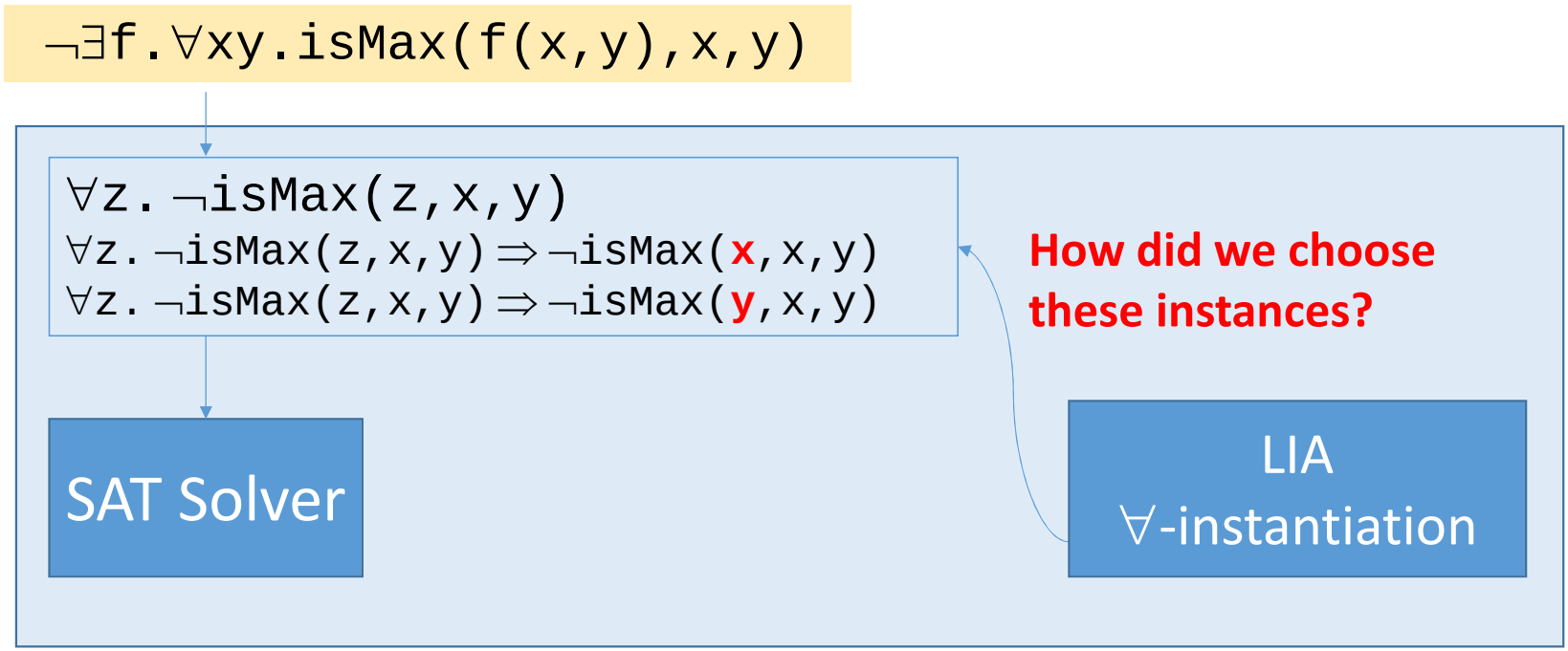
Fast Synthesis in SMT Solvers



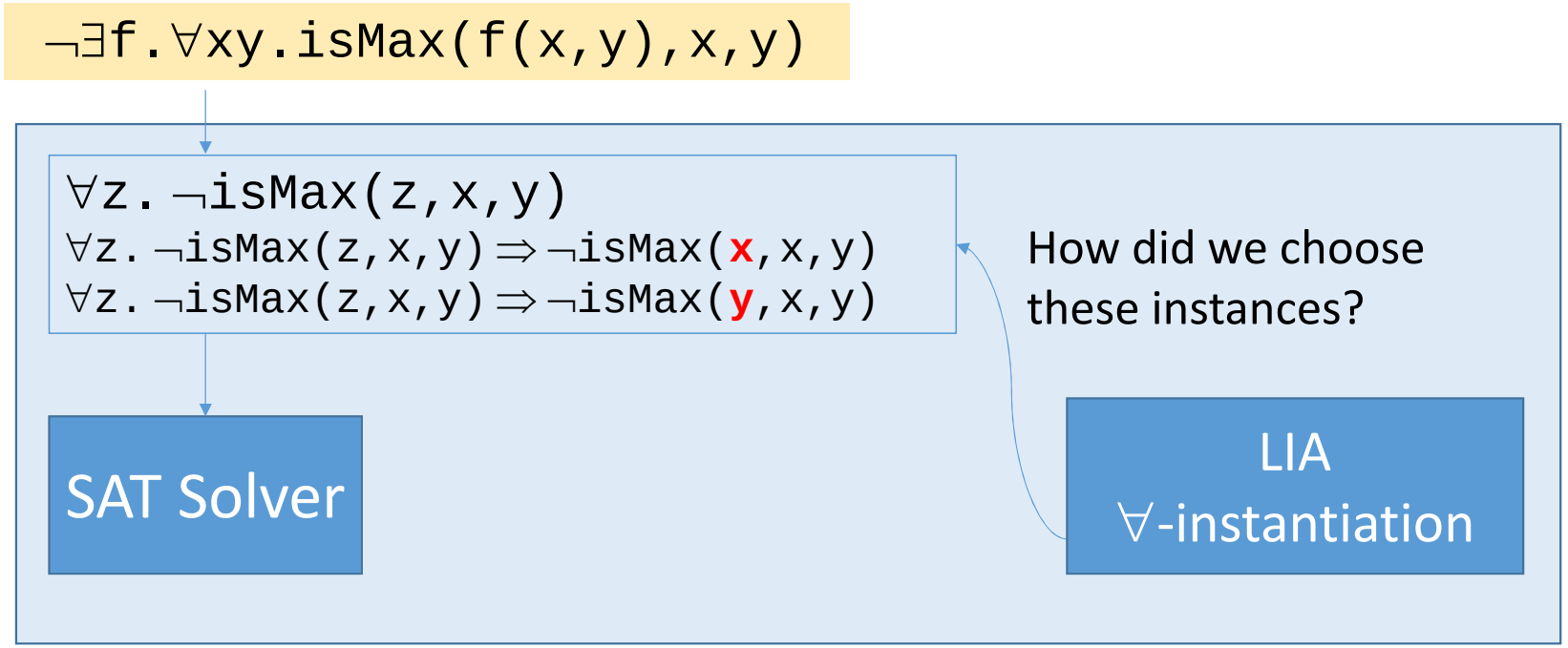
Fast Synthesis in SMT Solvers



Fast Synthesis in SMT Solvers



Fast Synthesis in SMT Solvers



⇒ 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 ∀-Instantiation

Quantifier Elimination Procedures

$\Leftarrow(\Rightarrow)^{?}$

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

$\Leftarrow\Rightarrow$

Synthesis procedures for single-invocation properties

Counterexample-Guided ∀-Instantiation

- SMT+∀ 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 + ∀ 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
  - to be determined?

***Finite** instantiation strategy ę **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:

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

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

CEGQI : Quality of Solutions

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

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

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

  - …but will not produce the intended solution for:

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

$$\lambda x.x-1$$

CEGQI : Quality of Solutions

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

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

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

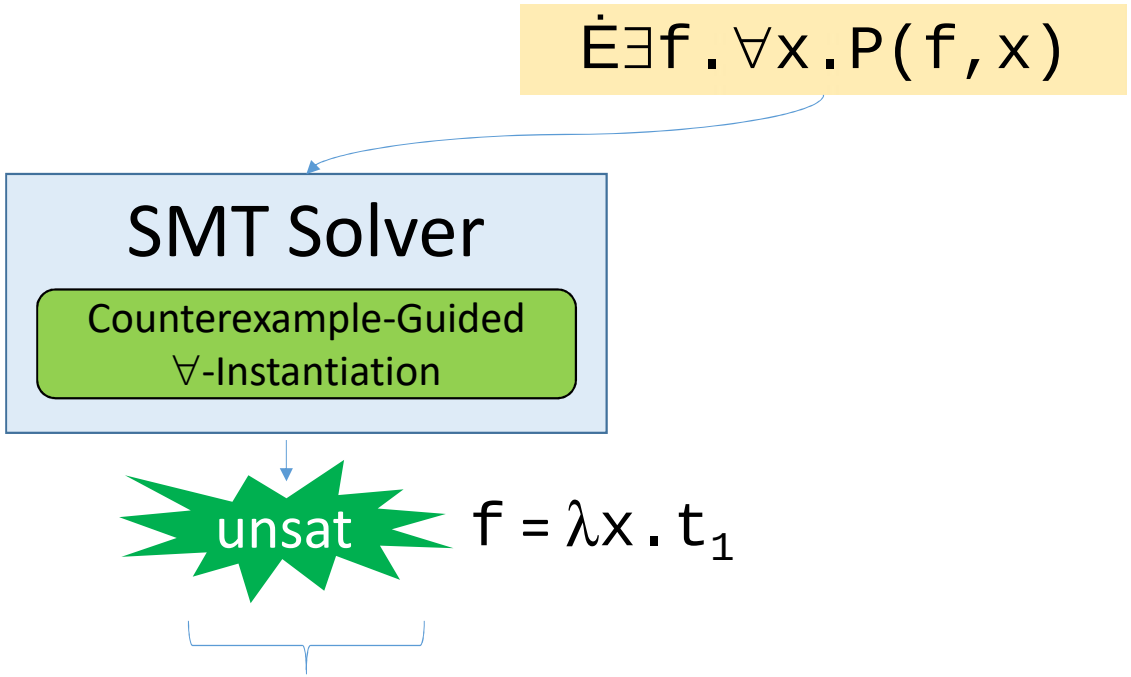
  - …but will not produce the intended solution for:

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

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

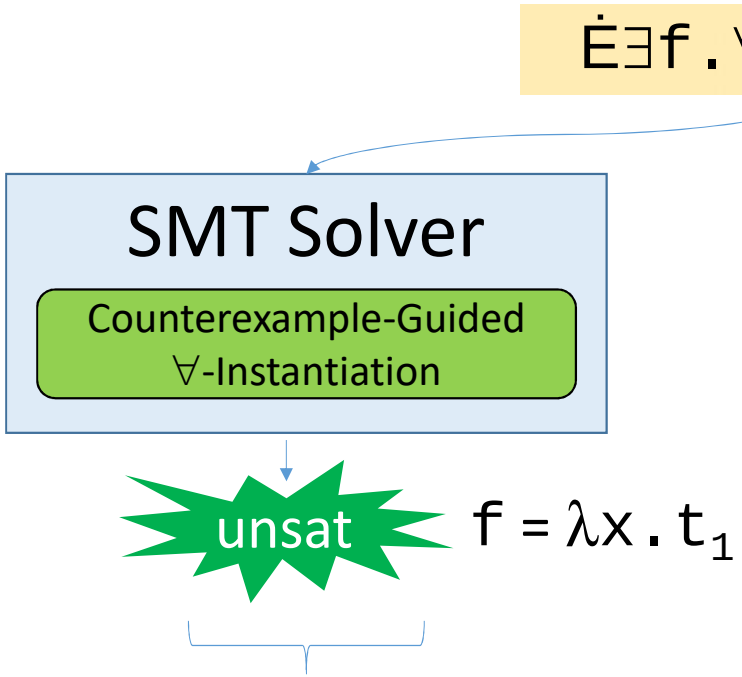
$\Rightarrow$This kind of solution is not useful for invariant synthesis and programming-by-examples

Refutation-Based Synthesis in SMT



Fast (and Complete)

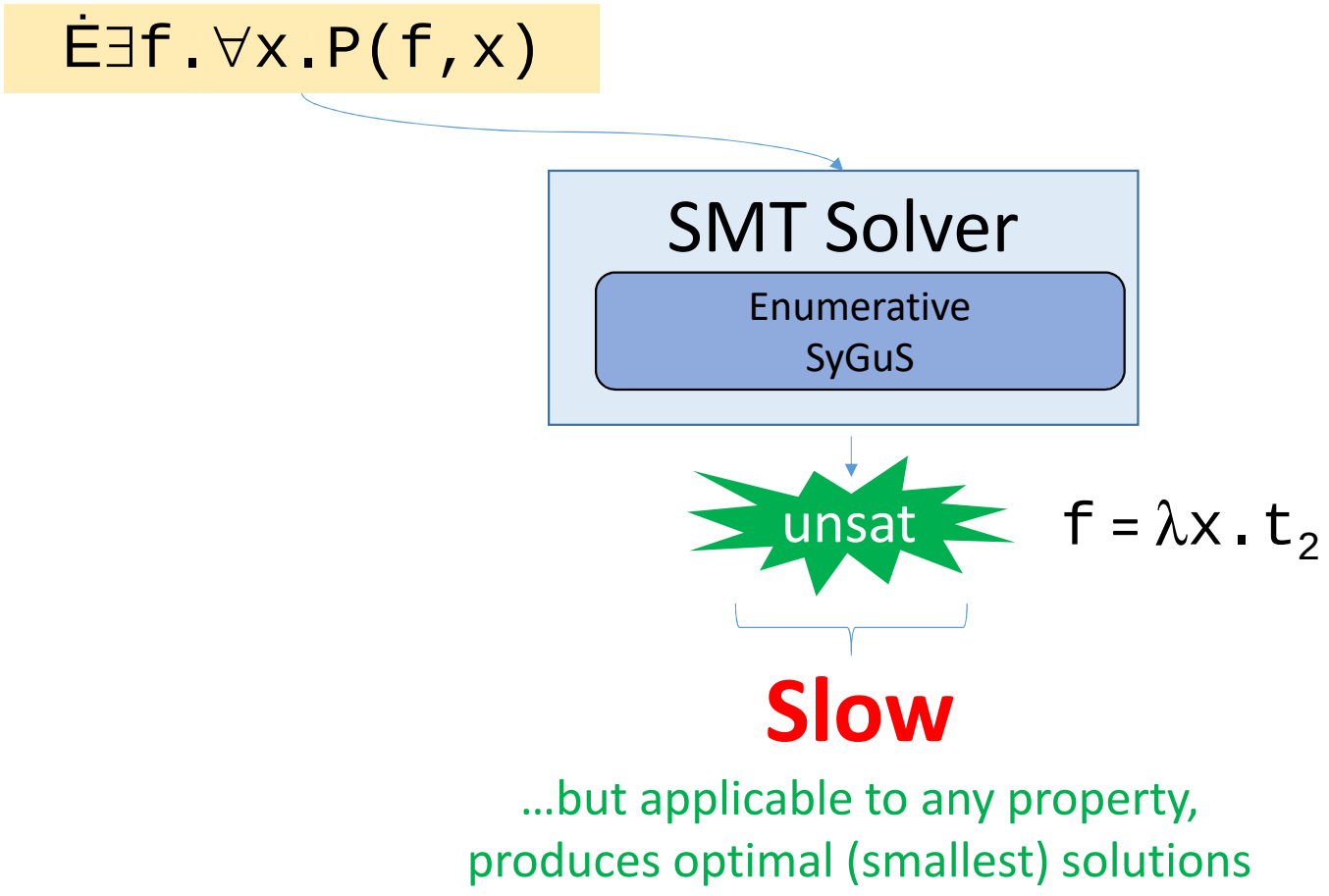
Refutation-Based Synthesis in SMT



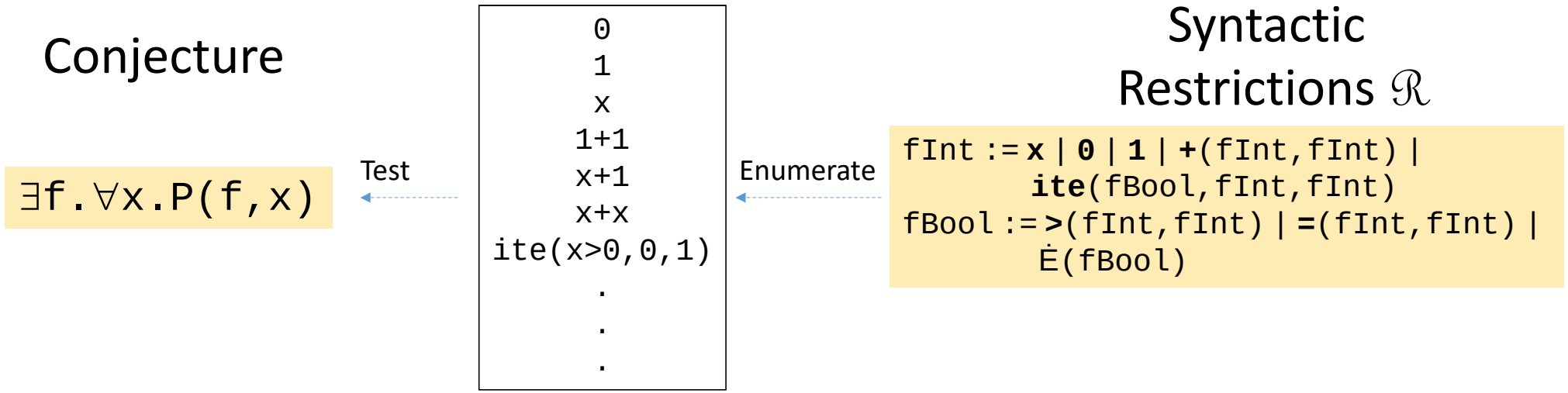
Fast (and Complete)

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

Refutation-Based Synthesis in SMT



Slow Synthesis in SMT Solvers



- 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

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

Syntactic Restrictions

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

View syntactic restrictions as an *inductive datatypes*

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 := ñ(fInt,fInt) | =(fInt,fInt)
```

Slow Synthesis in SMT Solvers

Conjecture

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

$$\mathtt{Int}\times\mathtt{Int}\rightarrow\mathtt{Int}$$

Encode using *deep embedding* involving **fInt**

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

fInt

Inductive Datatype

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

Slow Synthesis in SMT Solvers

Conjecture

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

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

$$E:\ \mathtt{I}\times\mathtt{Int}\times\mathtt{Int}\rightarrow\mathtt{Int}$$

Inductive Datatype

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

- **E**(d,x,y) *evaluates* d for arguments x,y, e.g. E(+(**x**,**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 := ñ(fInt,fInt) | =(fInt,fInt)
```

Ĝ *Solvable by combination of datatypes, LIA, UF, and ∃-instantiation*

Enumerative Syntax-Guided Synthesis

Conjecture

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

Inductive Datatype

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

Enumerative Syntax-Guided Synthesis

Conjecture

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

Inductive Datatype

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

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

Inductive Datatype

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

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

Inductive Datatype

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

- Solver generates a stream of candidate models, normalizes values $\downarrow$:
  - $d^{\mathcal{M}}$ = **x** … x $=\downarrow$ x
  - $d^{\mathcal{M}}$ = **y** … y $=\downarrow$ y
  - $d^{\mathcal{M}}$ = **+(1,y)** … 1+y $=\downarrow$ y+1
  - $d^{\mathcal{M}}$ = **+(0,x)** … 0+x $=\downarrow$ x
  - $d^{\mathcal{M}}$ = **+(y,1)** … y+1 $=\downarrow$ y+1

Enumerative Syntax-Guided Synthesis

Conjecture

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

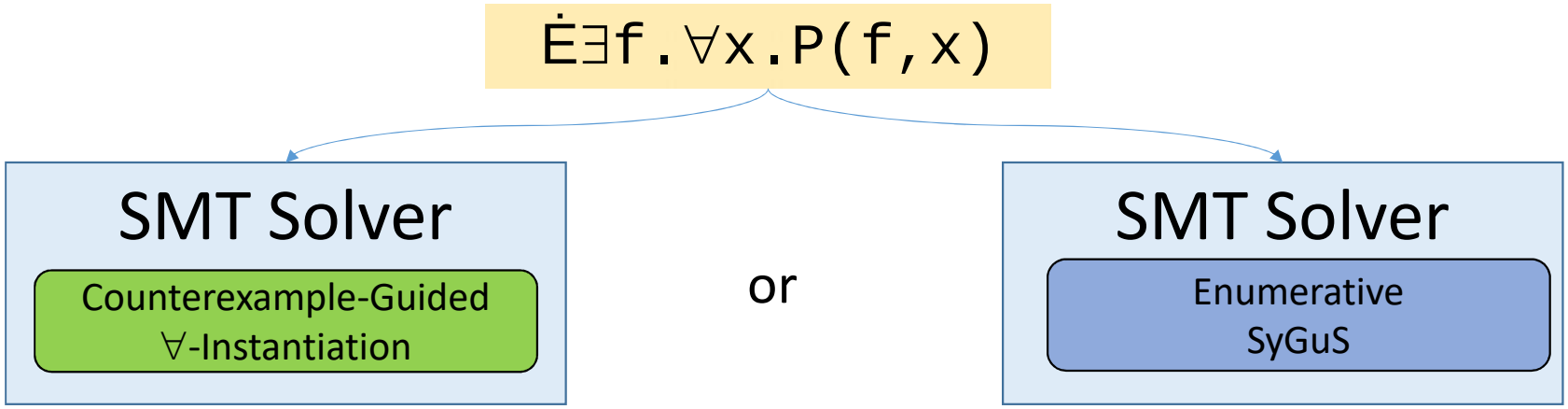
Inductive Datatype

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

- Solver generates a stream of candidate models, normalizes values $\downarrow$:
  - $d^{\mathcal{M}}$ = **x** … x $=\downarrow$ x
  - $d^{\mathcal{M}}$ = **y** … y $=\downarrow$ y
  - $d^{\mathcal{M}}$ = **+(1,y)** … 1+y $=\downarrow$ y+1
  - $d^{\mathcal{M}}$ = ~~**+(0,x)**~~ … 0+x $=\downarrow$ x
  - $d^{\mathcal{M}}$ = ~~**+(y,1)**~~ … y+1 $=\downarrow$ y+1

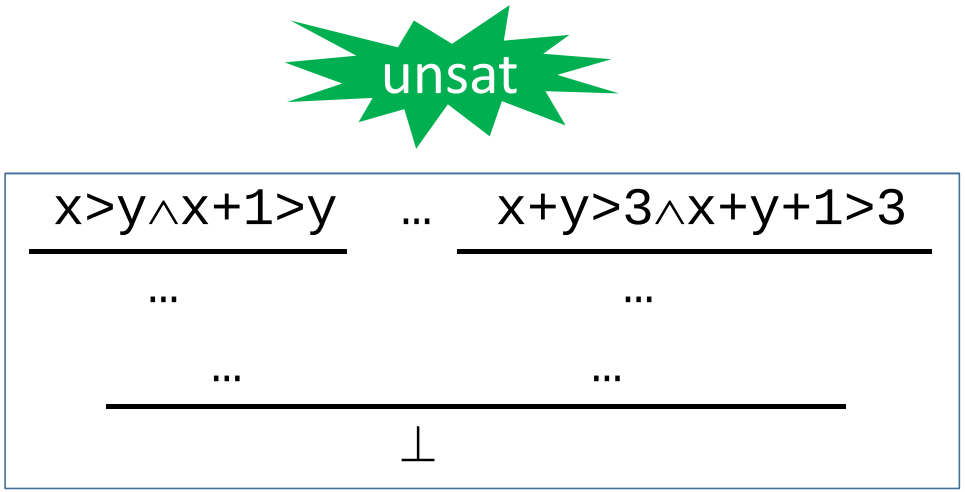
Avoid candidate solutions not unique up to theory normalization

Fast and Slow Synthesis Procedures in SMT



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

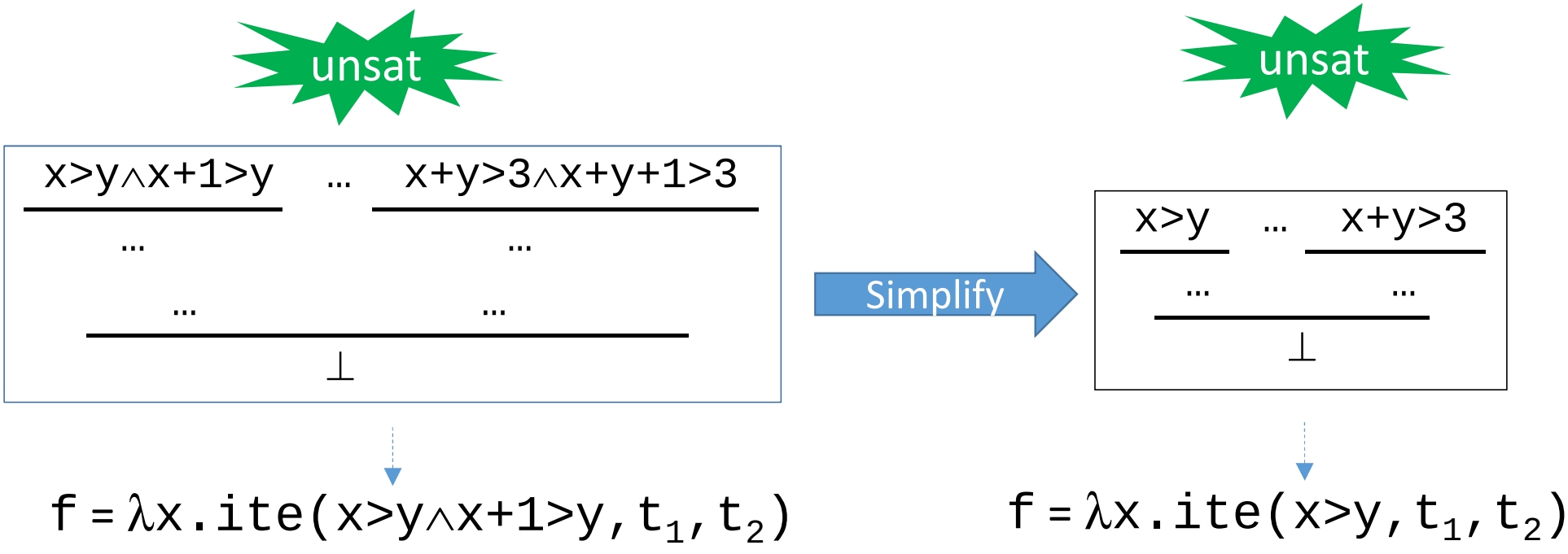
Shorter Solutions via Proof Compression



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

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?*

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

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

What if conjecture is *Partially Single Invocation?*

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

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…

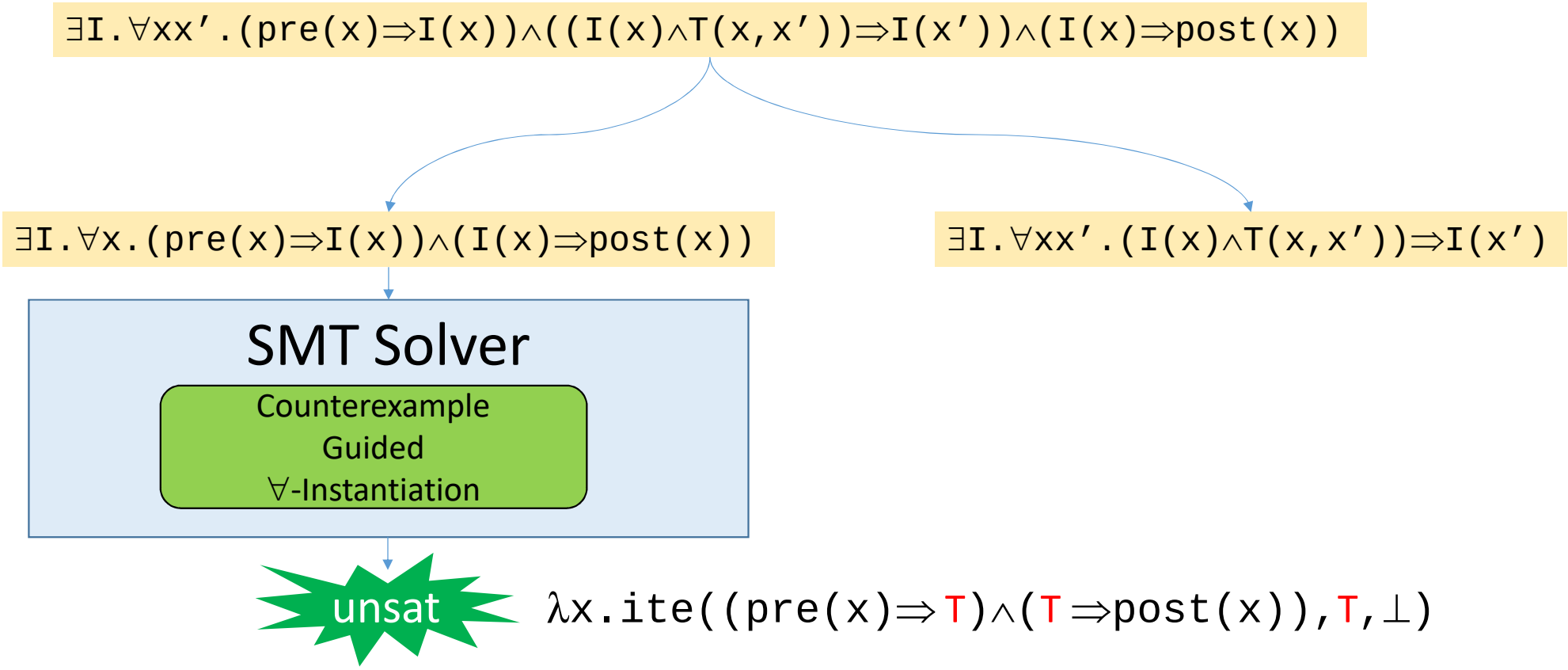
$\exists I.\forall x.(pre(x)\Rightarrow \mathbf{I(x)})\wedge(\mathbf{I(x)}\Rightarrow post(x))$

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

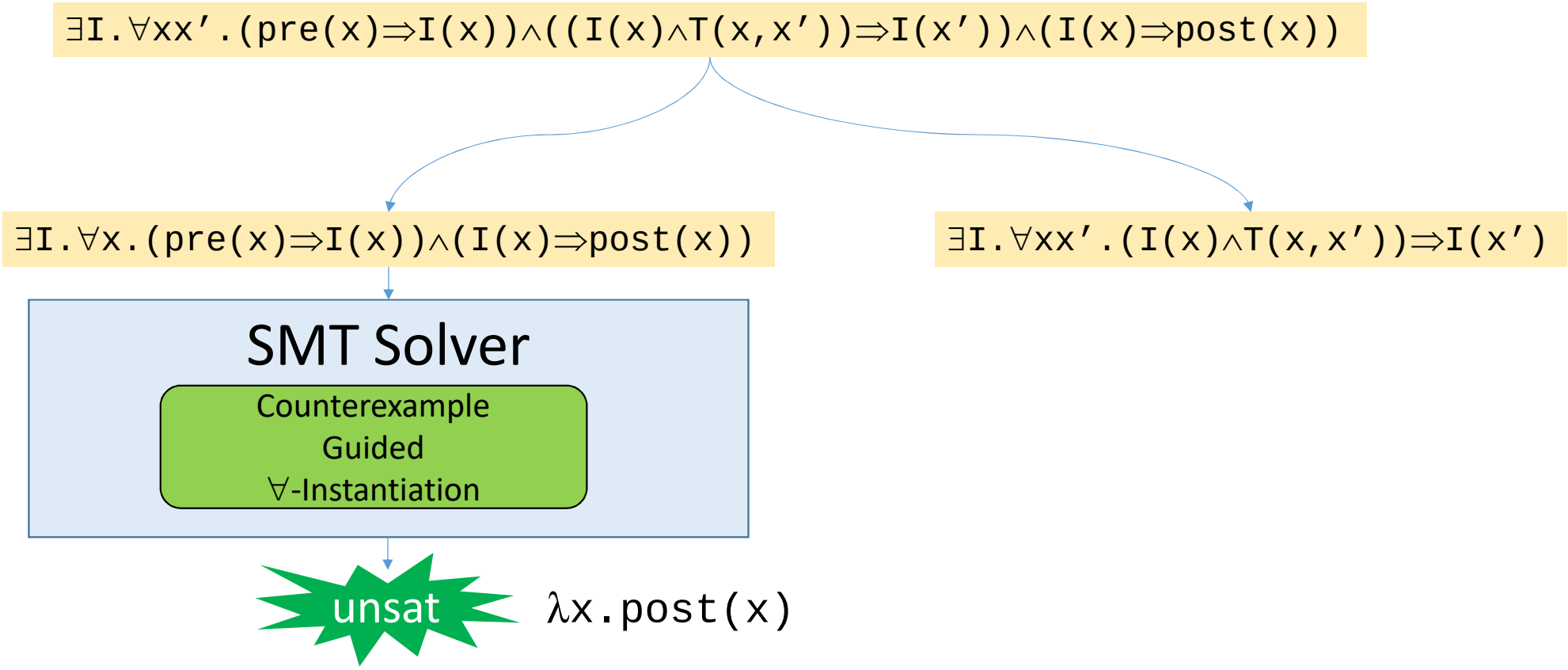
Single-invocation portion

Non-single-invocation portion

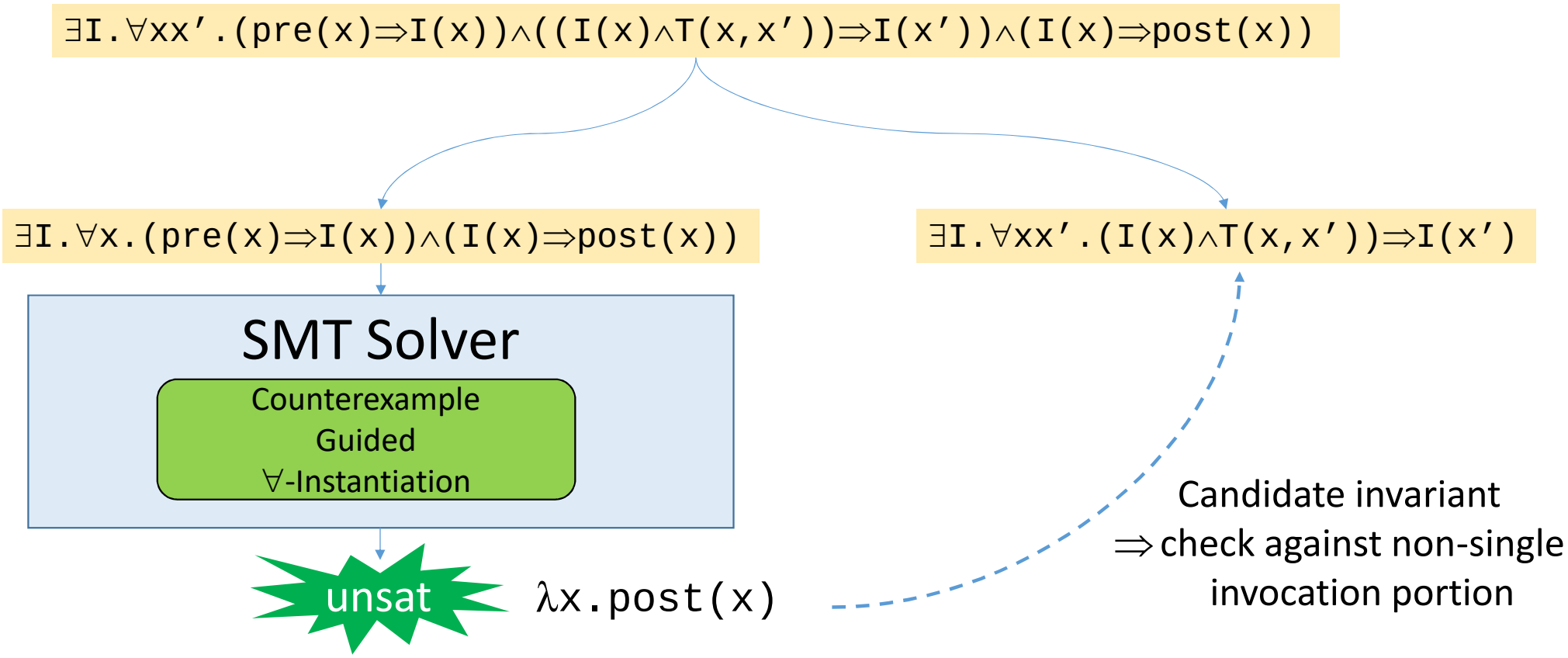
What if conjecture is *Partially Single Invocation?*



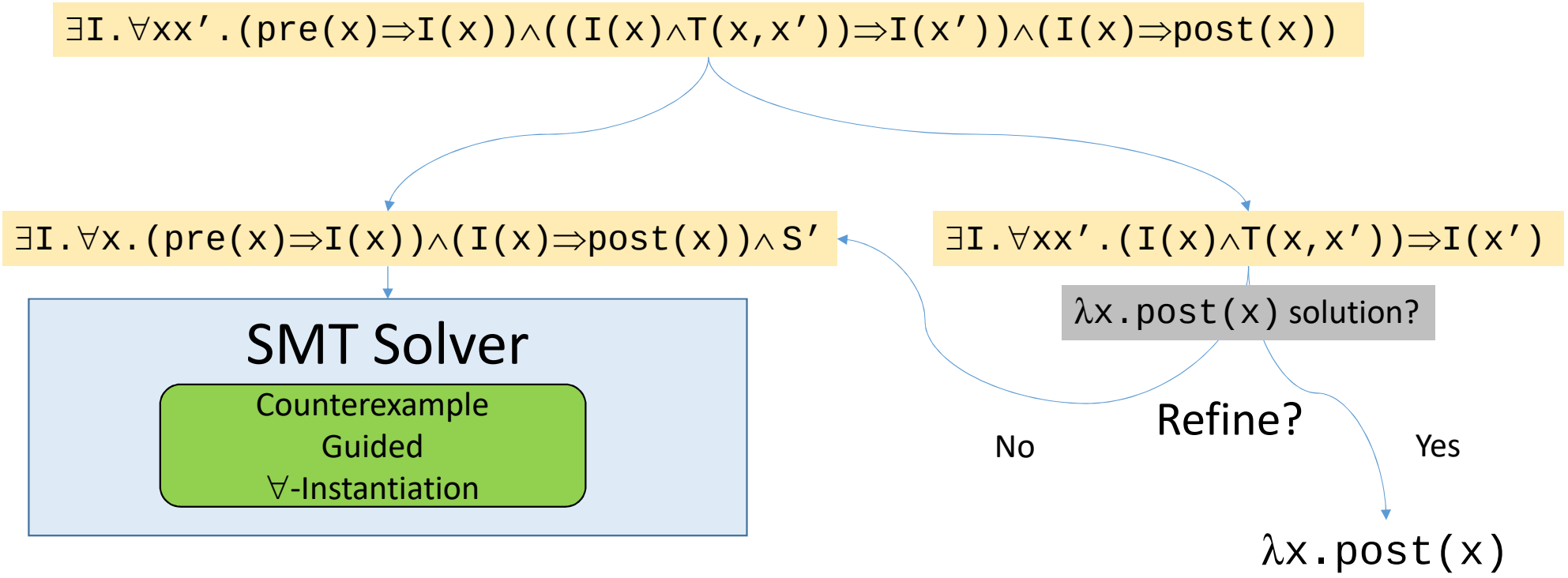
What if conjecture is *Partially Single Invocation?*



What if conjecture is *Partially Single Invocation?*



What if conjecture is *Partially Single Invocation?*



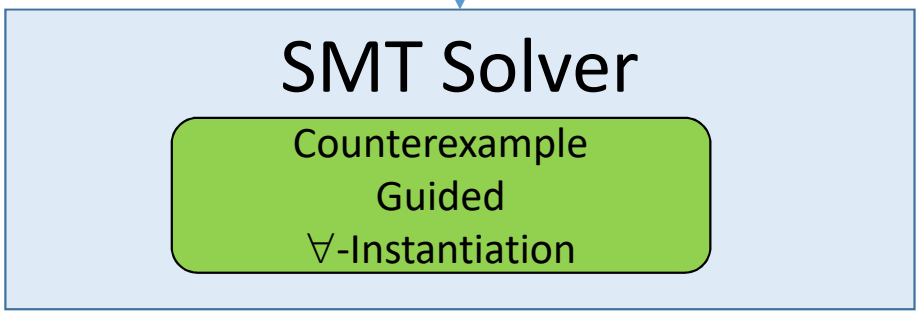
⇒Related to property-directed reachability (PDR) **[Bradley 2011]**

What if conjecture is *Partially Single Invocation?*

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

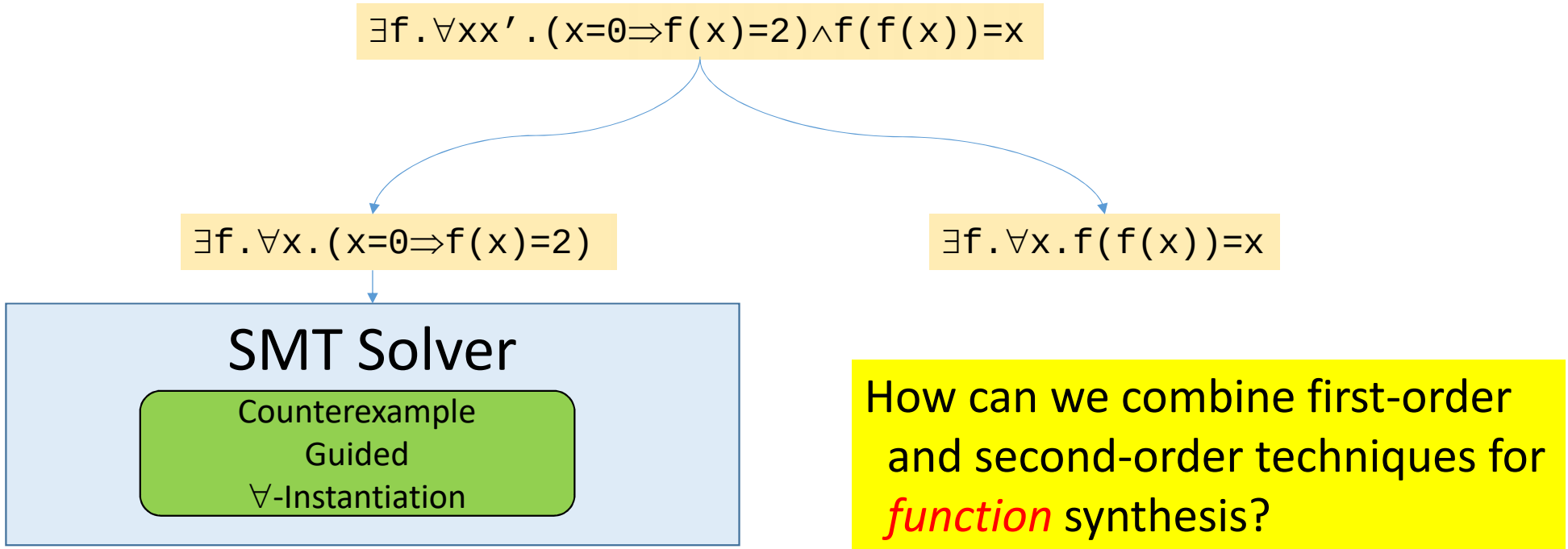
$\exists I.\forall x.(pre(x)\Rightarrow I(x))\wedge(I(x)\Rightarrow post(x))\wedge S'$

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



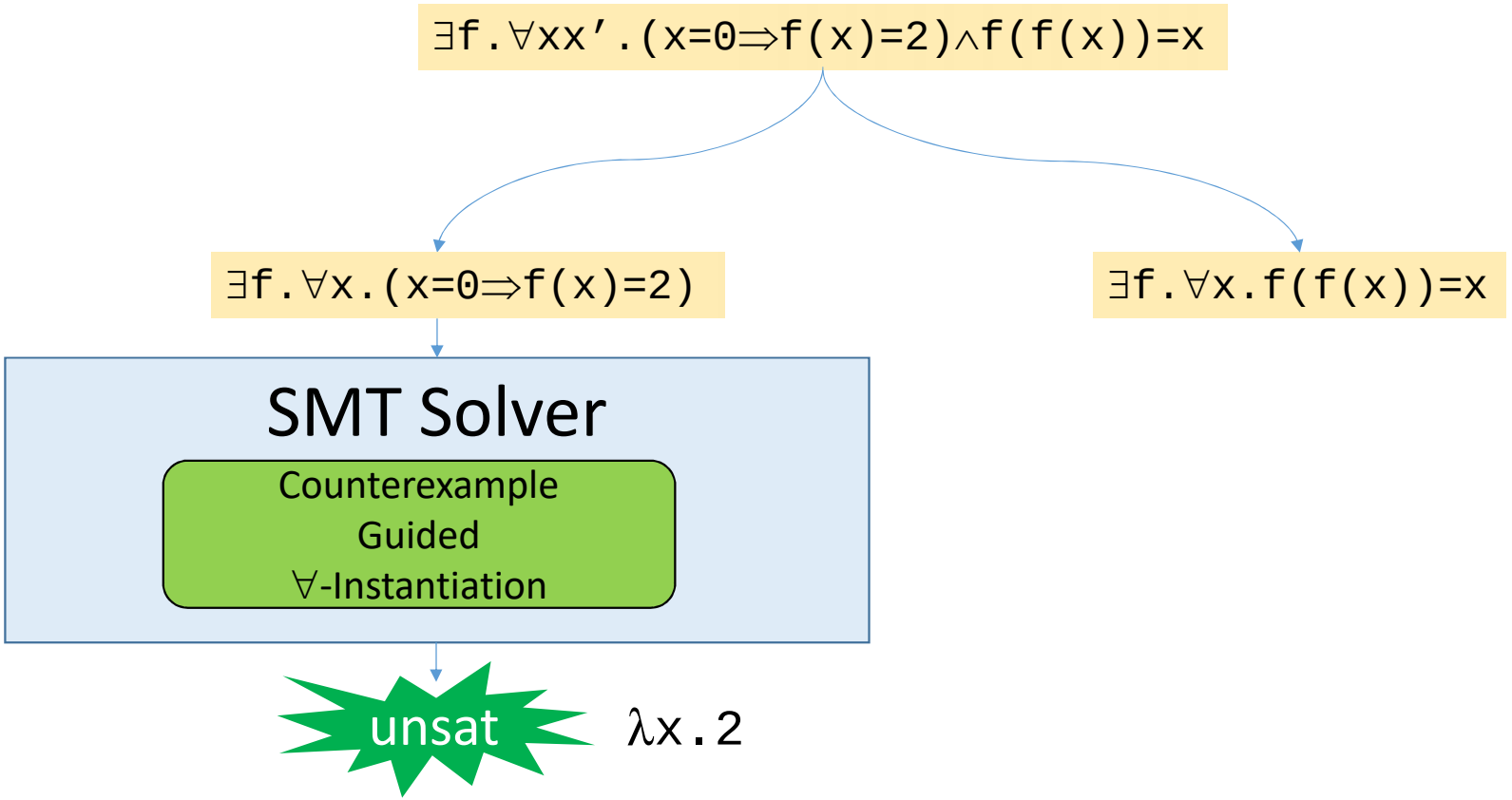
How can we combine first-order and second-order techniques for *function* synthesis?

What if conjecture is *Partially Single Invocation?*

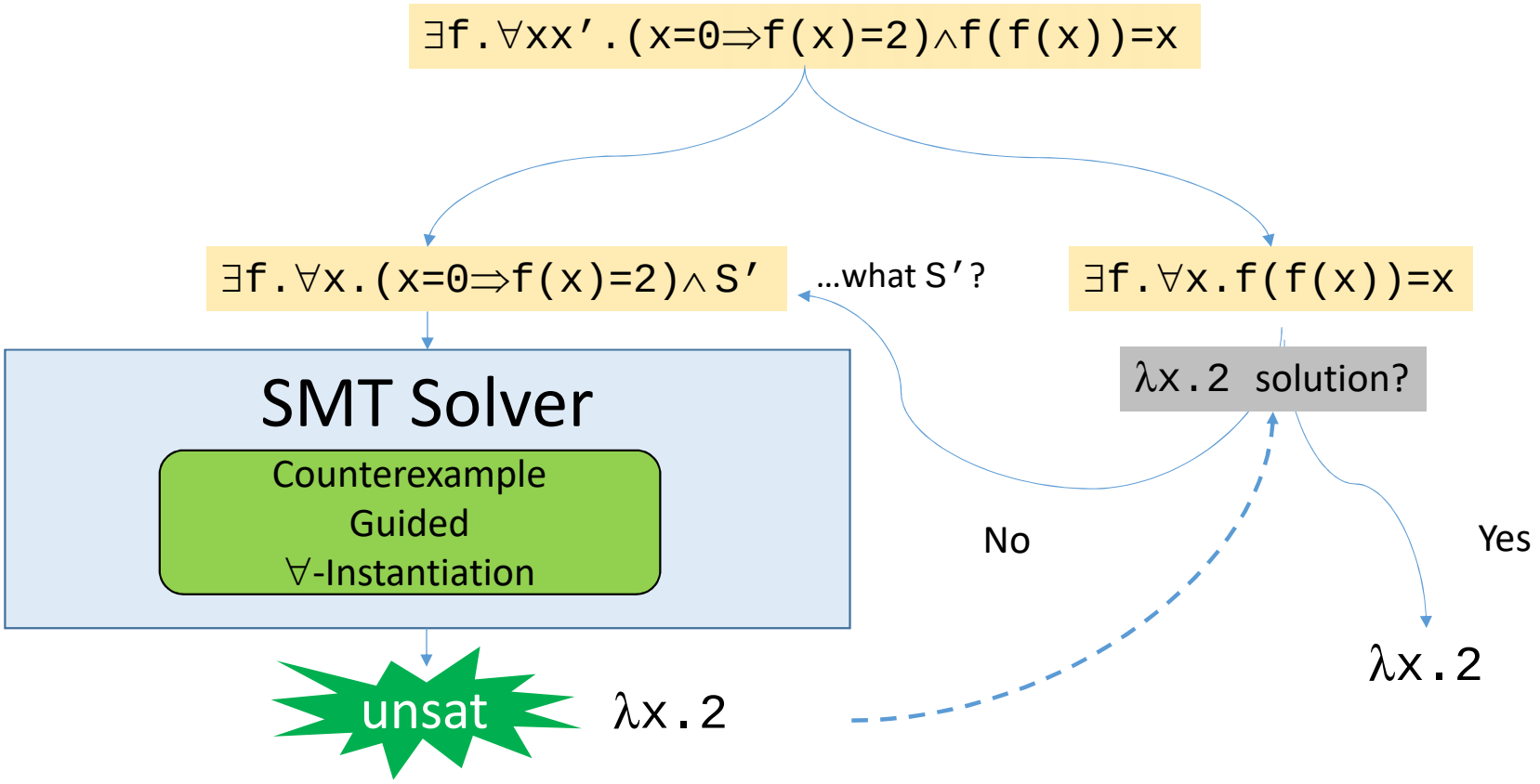


How can we combine first-order and second-order techniques for *function* synthesis?

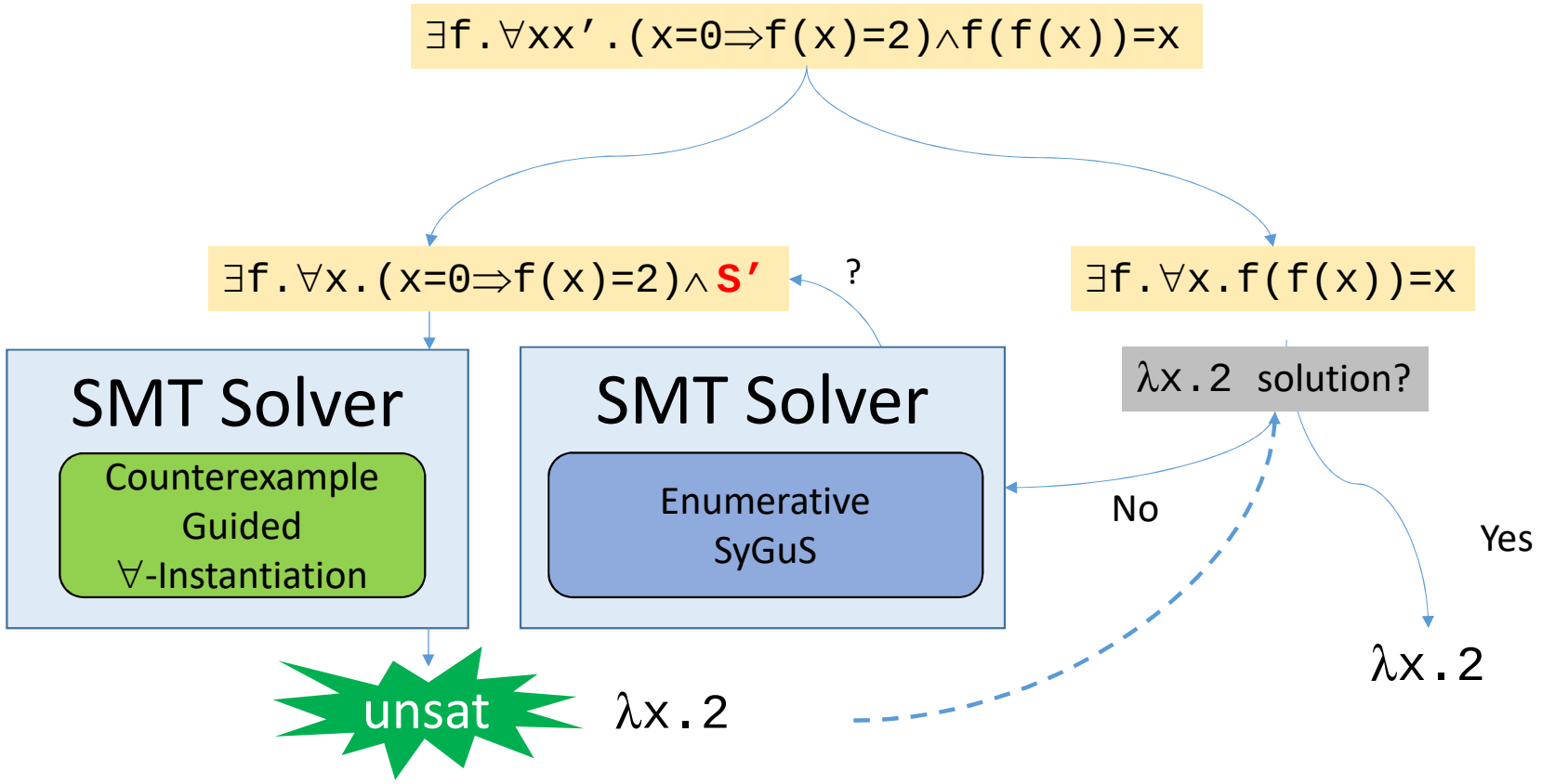
What if conjecture is *Partially Single Invocation?*



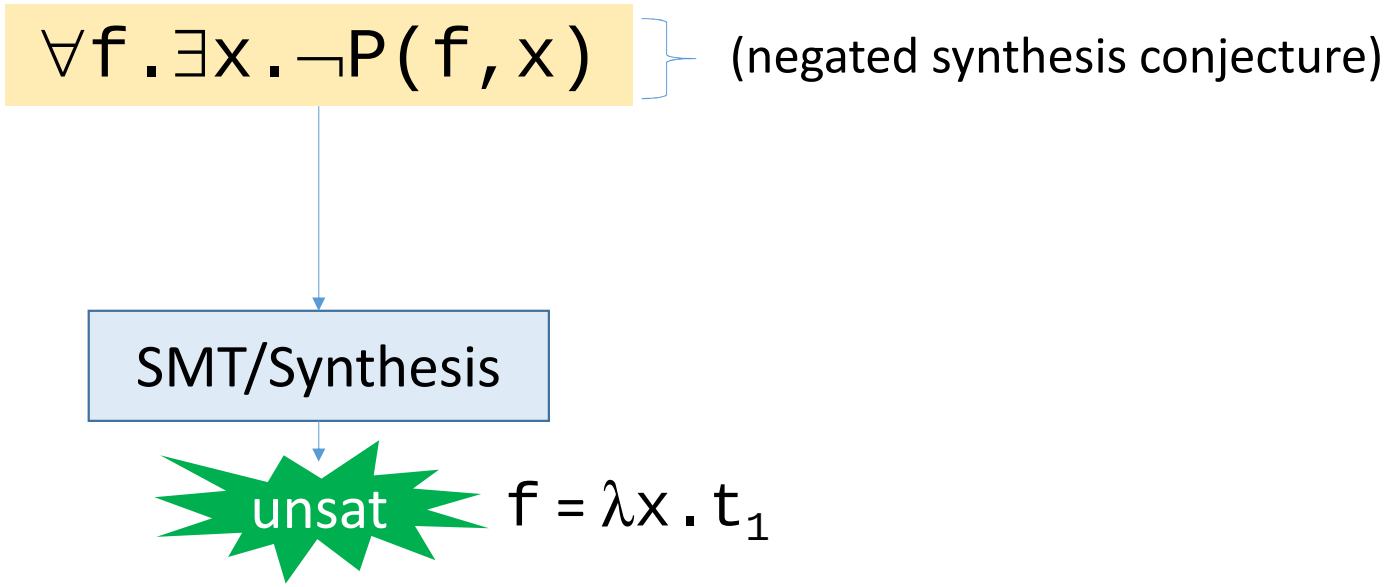
What if conjecture is *Partially Single Invocation?*



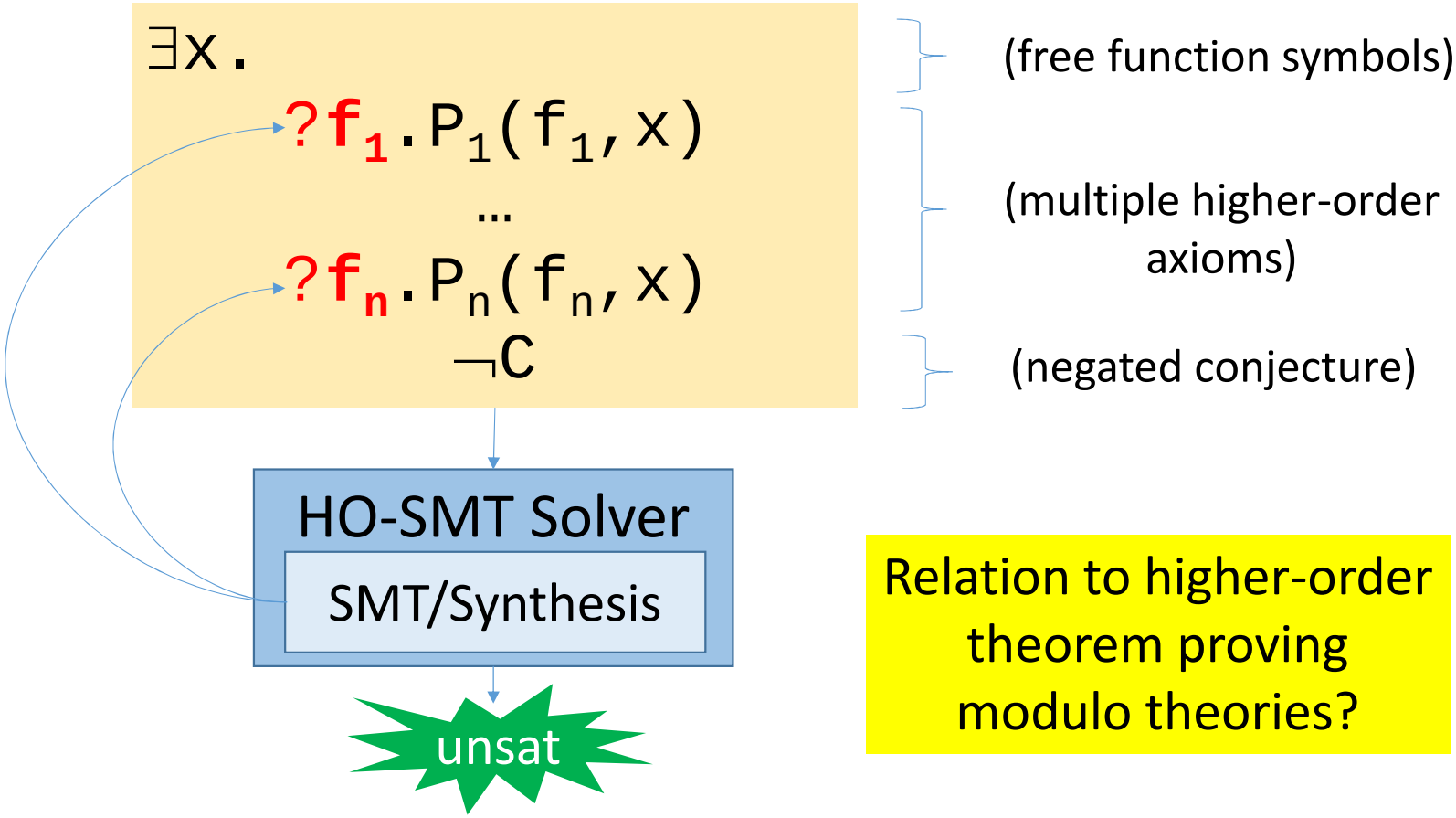
What if conjecture is *Partially Single Invocation?*



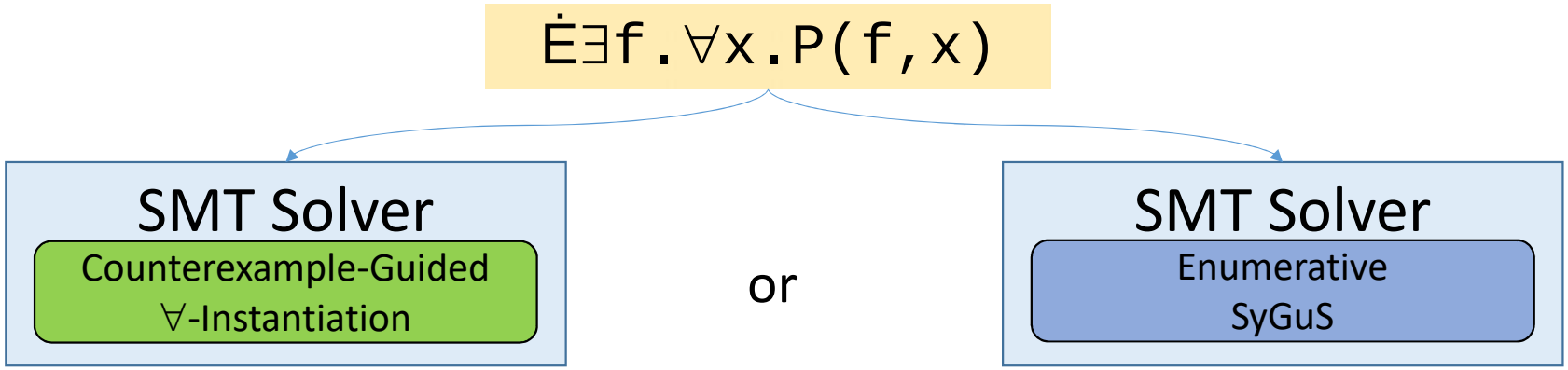
Synthesis for Higher-Order Theorem Proving?



Synthesis for Higher-Order Theorem Proving?



Fast and Slow Synthesis Procedures in SMT



Fast,

Complete for (in)Feasibility

Non-optimal Solutions

- Proof compression techniques?

Restricted to Single-Inv Properties

- Useful for partially single-inv properties?

Slow

- Better navigation of the enumerative space?

Cannot show infeasibility

- Generalize pruning of enumerative space?

Optimal (Shortest) Solutions,

Applies to all Second-Order Conjectures