SyGuS Techniques in the Core of an SMT Solver

Andrew Reynolds

SYNT Workshop

July 22, 2017

THE UNIVERSITY OF IOWA

Acknowledgements

- Thanks to past and present collaborators:
  - EPFL : Viktor Kuncak
  - University of Iowa : Cesare Tinelli, Arjun Viswanathan
  - Stanford University : Clark Barrett
  - Google : Tim King
  - NYU : Morgan Deters

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

THE UNIVERSITY OF IOWA

Stanford University

Google

NYU

SMT Solvers for Synthesis

- Act as *subroutines* for automated synthesis tasks

- More recently, instrumented as *stand-alone tools* for synthesis
  - SMT solver CVC4 has entered SyGuS comp 2015, 2016, 2017

    [Reynolds et al CAV2015]

In This Talk

- Synthesis conjectures: $\exists f . \forall x . P(f,x)$

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

In This Talk

- Synthesis conjectures: $\exists f. \forall x. P(f,x)$

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

…(optionally) with syntactic restrictions:

$\mathcal{R}$:
fInt := **x** | **0** | **1** | **+**(fInt, fInt) | **ite**(fBool, fInt, fInt)
fBool := **>**(fInt, fInt) | **=**(fInt, fInt) | **¬**(fBool)

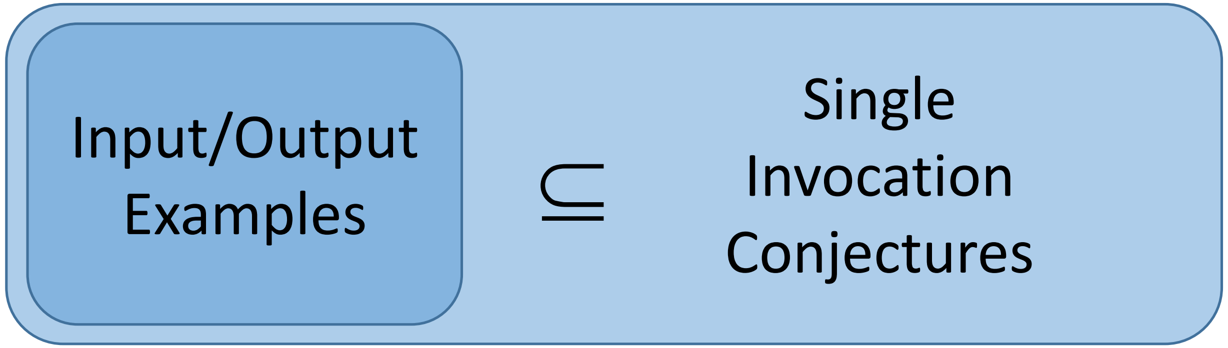
Find solutions $f = \lambda x . t$, where t is generated by grammar $\mathcal{R}$

Synthesis Conjectures : Overview

Input/Output Examples

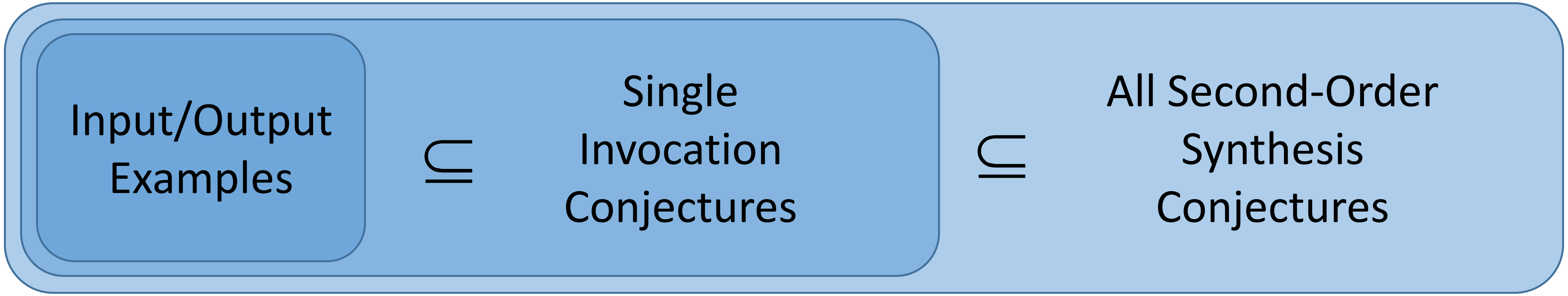
e.g. $\exists f. \forall x. (x=\mathbf{i}_1 \Rightarrow f(x)=\mathbf{o}_1) \wedge (x=\mathbf{i}_2 \Rightarrow f(x)=\mathbf{o}_2) \wedge (x=\mathbf{i}_3 \Rightarrow f(x)=\mathbf{o}_3)$

Synthesis Conjectures : Overview



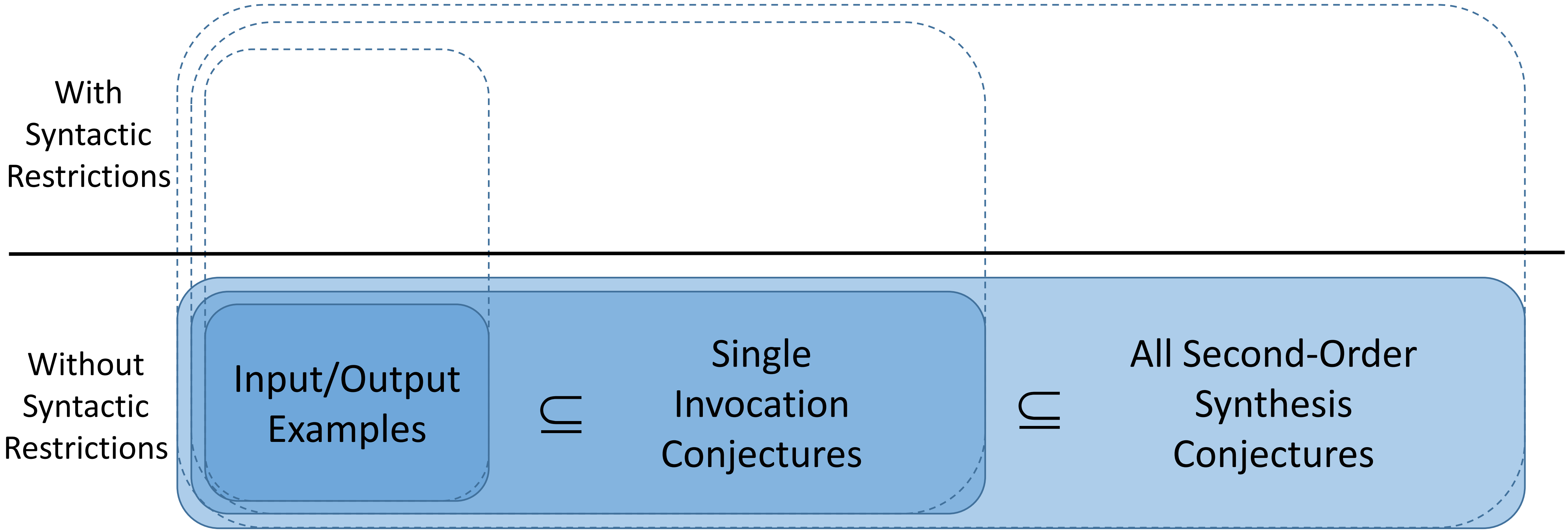
e.g. $\exists f.\forall xy.\ f(x,y) \geq x \wedge f(x,y) \geq y \wedge (f(x,y) = x \vee f(x,y) = y)$

Synthesis Conjectures : Overview



e.g. $\exists f. \forall x y.\ f(x,y) = f(y,x)$

Synthesis Conjectures : Overview



Synthesis Conjectures : Overview

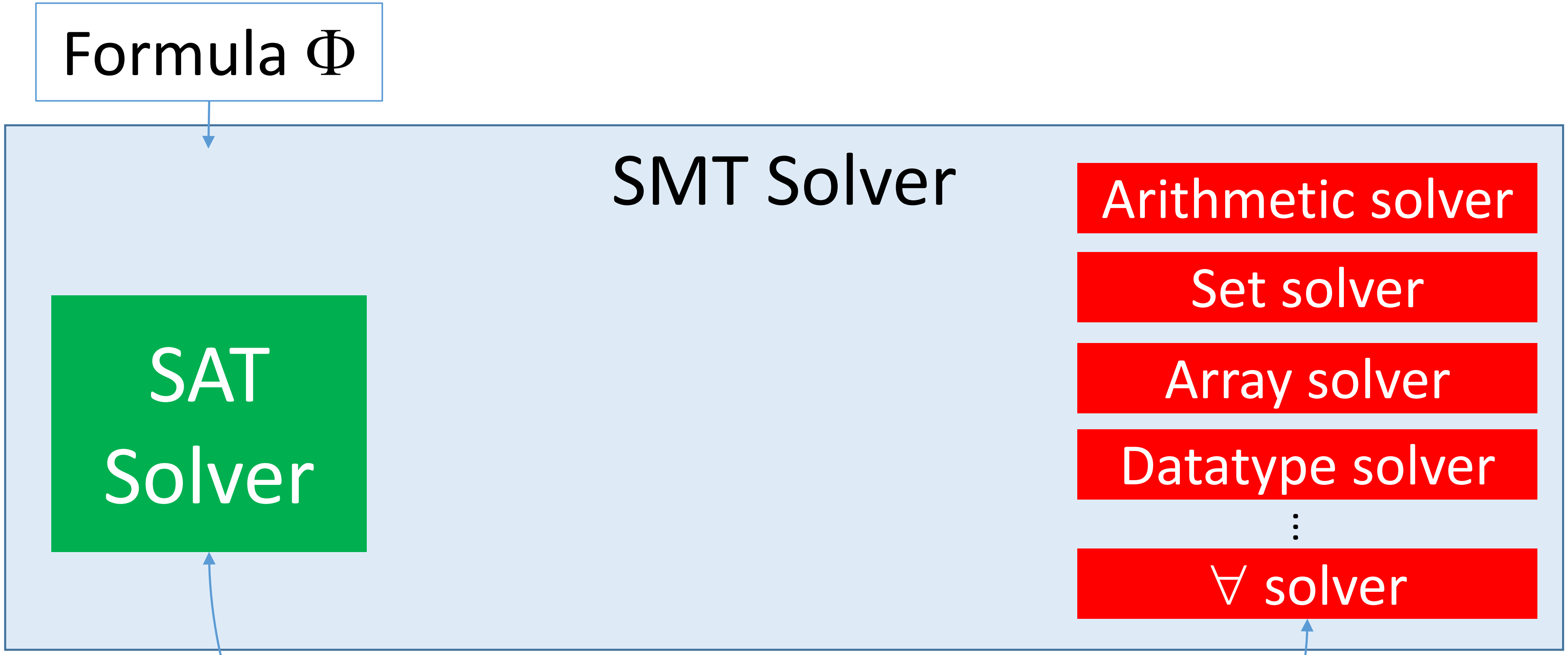
	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	?	?	?
Without Syntactic Restrictions	?	?	?

Synthesis Conjectures : Overview

	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	?	?	?
Without Syntactic Restrictions	?	?	?

Will show how **DPLL(T)-based SMT solvers** can handle each class of conjecture

DPLL(T)-based SMT Solvers



- Determines satisfiability of Φ via combination of: *SAT solver* and *dedicated theory solvers*

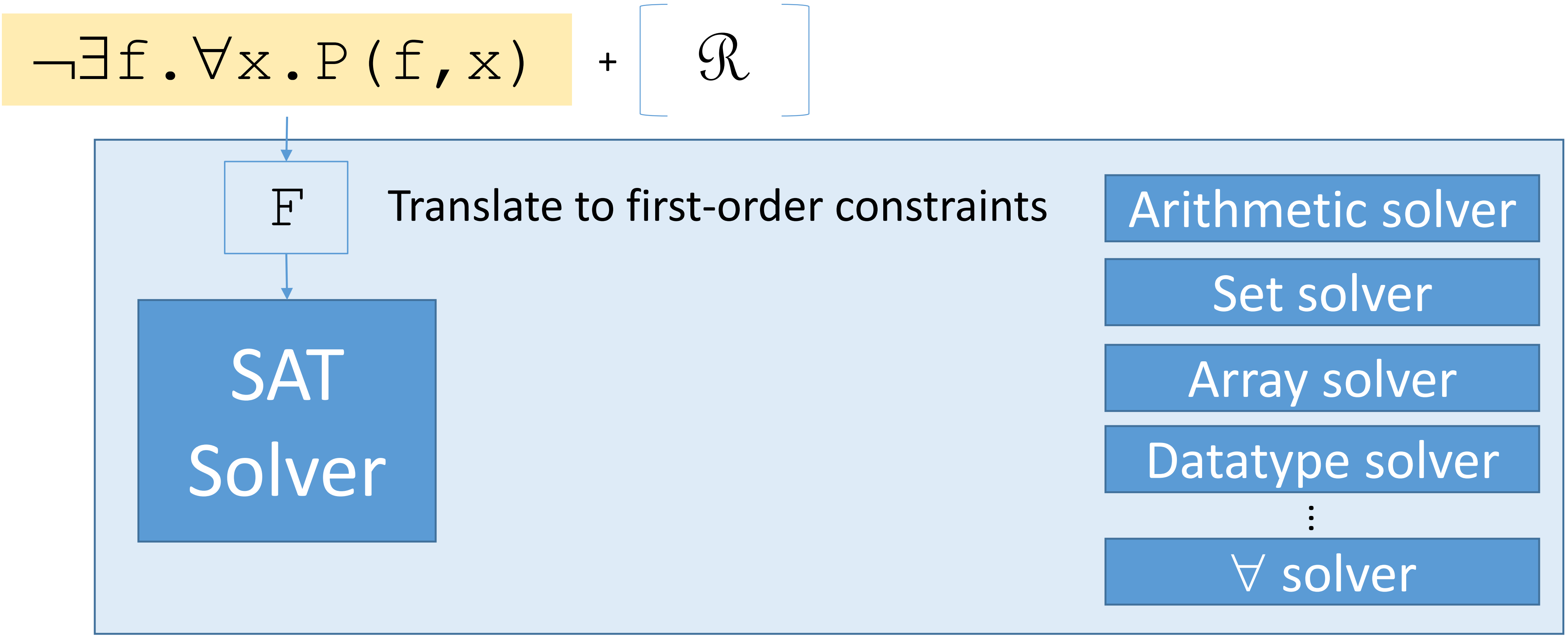
DPLL(T)-based SMT Solvers for Synthesis

DPLL(T)-based SMT Solvers for Synthesis

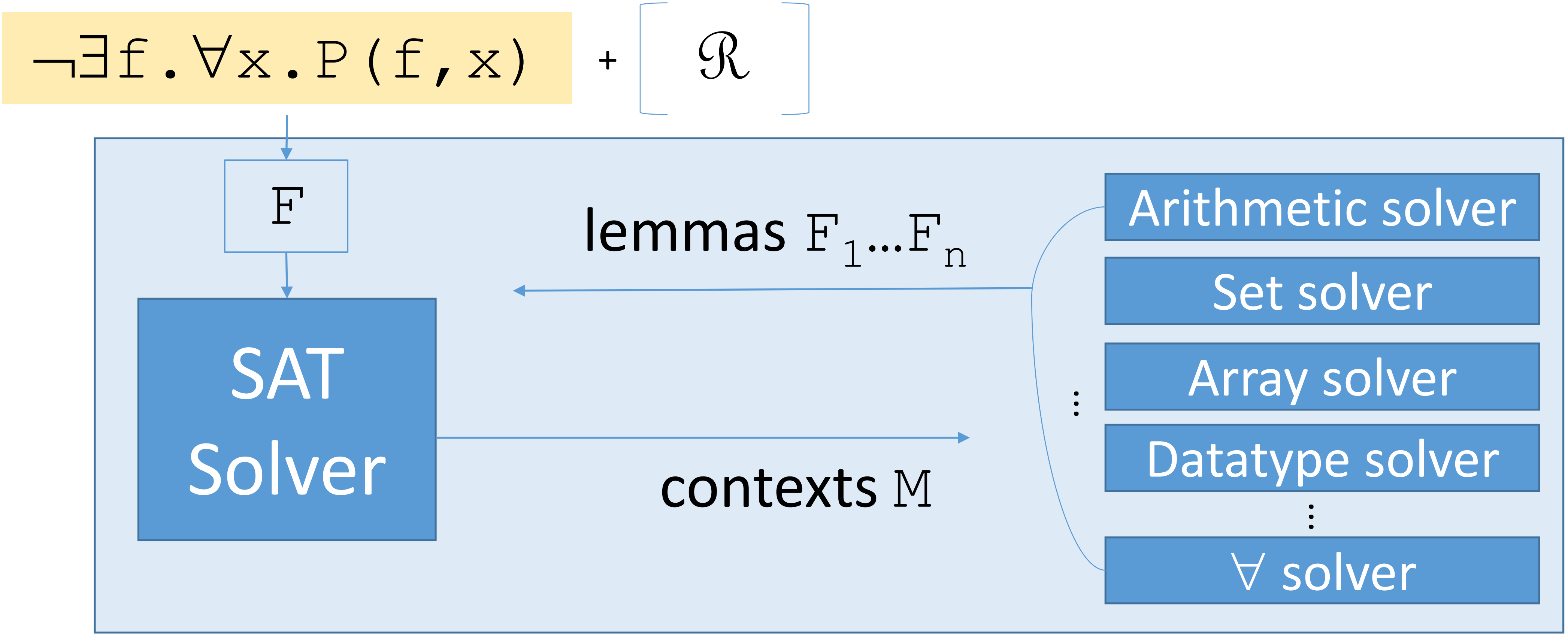
$\neg\exists f.\forall x.P(f,x)$ + $\mathfrak{R}$

Negated Synthesis Conjecture
(+ syntactic restrictions $\mathfrak{R}$)

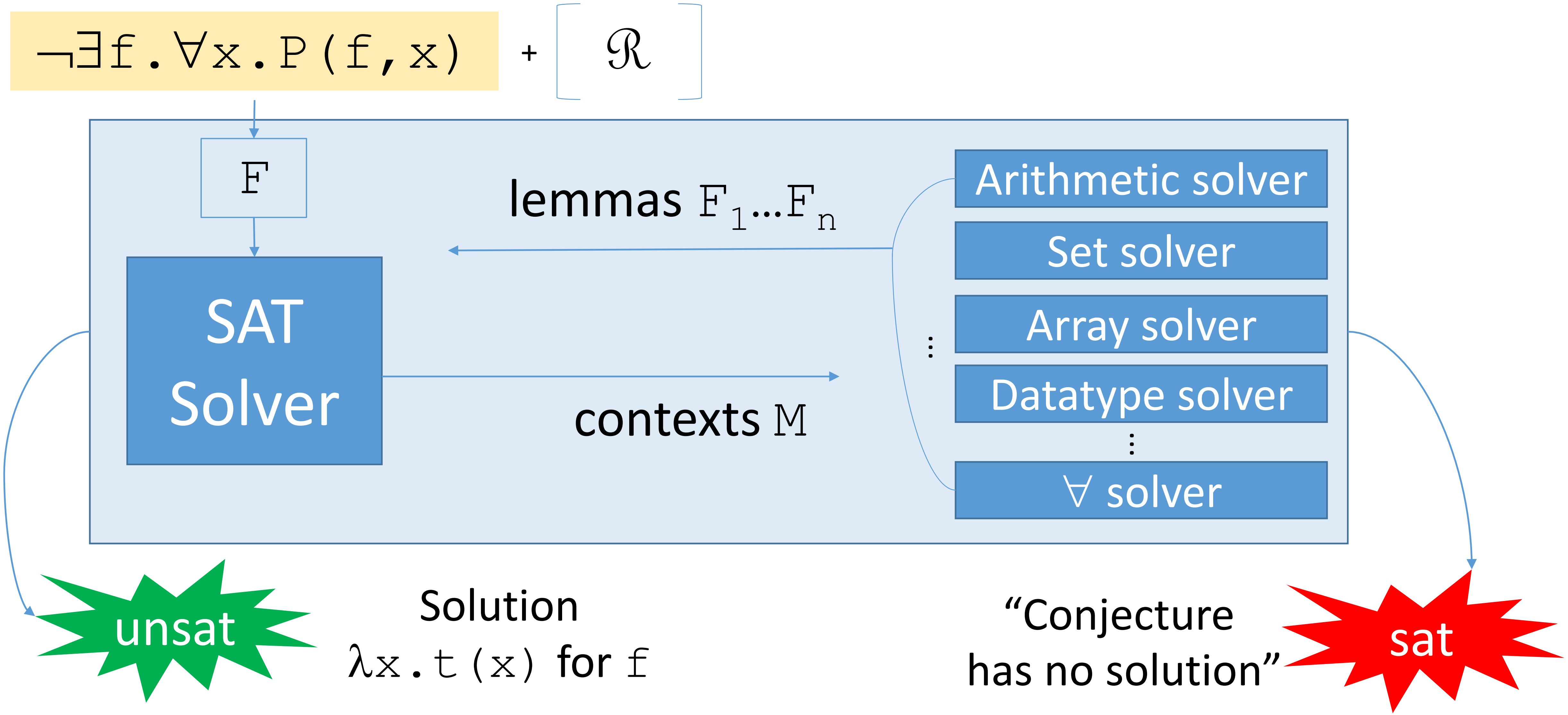
DPLL(T)-based SMT Solvers for Synthesis



DPLL(T)-based SMT Solvers for Synthesis



DPLL(T)-based SMT Solvers for Synthesis



Single Invocation w/o Syntactic Restrictions

With Syntactic Restrictions	?	?	?
Without Syntactic Restrictions	?	?	?
	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures

Single Invocation w/o Syntactic Restrictions

- Some synthesis conjectures are *essentially first-order*

Single Invocation w/o Syntactic Restrictions

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

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

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

Single Invocation w/o Syntactic Restrictions

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

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

All occurrence of `f` are in terms of the form **`f(x,y)`**

$\Rightarrow$ "single invocation" synthesis conjectures

Single Invocation w/o Syntactic Restrictions

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

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

Single Invocation w/o Syntactic Restrictions

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

Int × Int → Int

Anti-skolemize

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

Int

[Reynolds et al CAV2015]

Single Invocation w/o Syntactic Restrictions

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

Int × Int → Int

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

Int

"for each x,y, there exists a return value $\mathbf{z}$ that is the maximum of x and y"

[Reynolds et al CAV2015]

Single Invocation w/o Syntactic Restrictions

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

Int × Int → Int

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

Int

Simplify

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

[Reynolds et al CAV2015]

Single Invocation w/o Syntactic Restrictions

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

$\mathrm{Int} \times \mathrm{Int} \rightarrow \mathrm{Int}$

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

Int

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

First-order linear arithmetic ⇒ Solvable by first-order ∀-instantiation

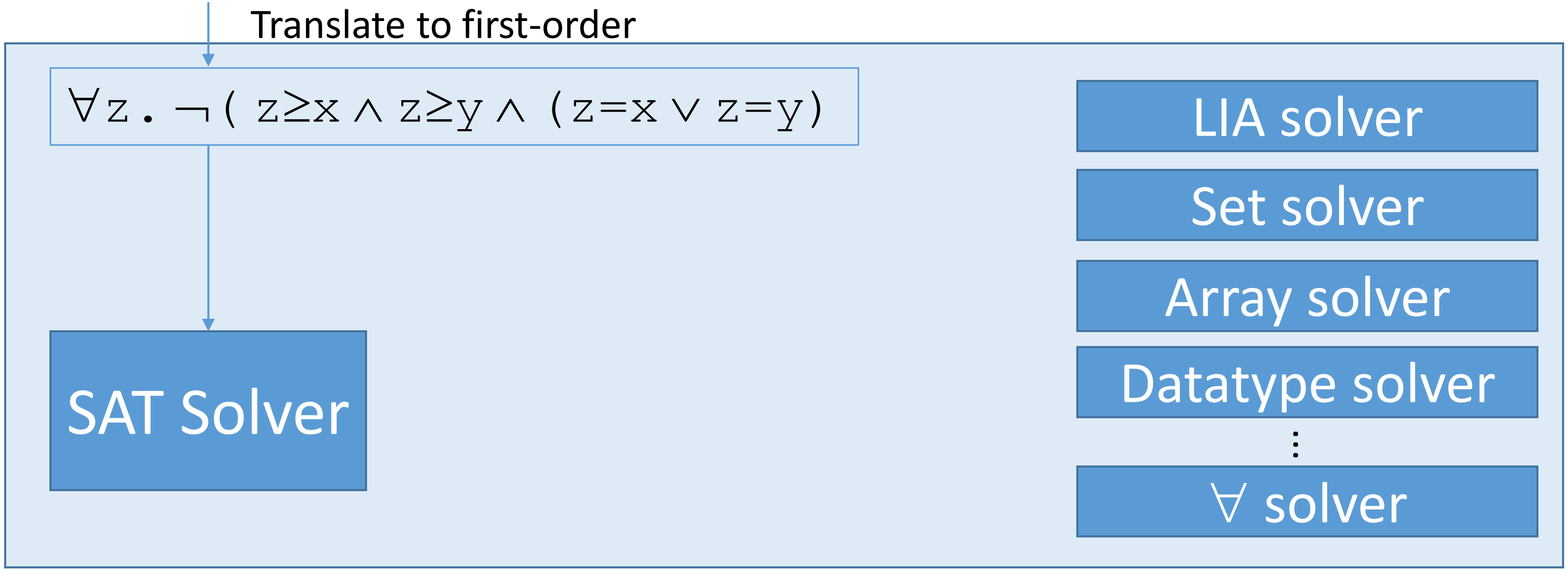
[Reynolds et al CAV2015]

Single Invocation Synthesis in SMT

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

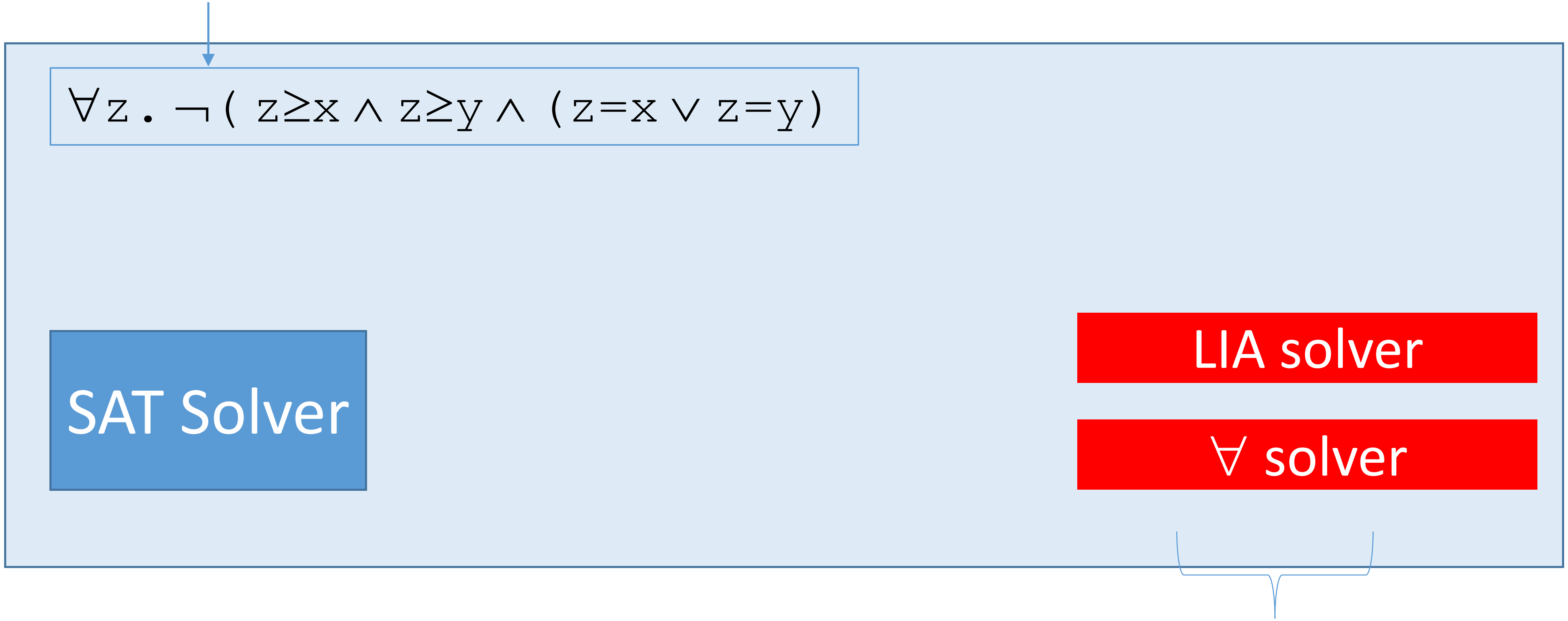
Single Invocation Synthesis in SMT

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



Single Invocation Synthesis in SMT

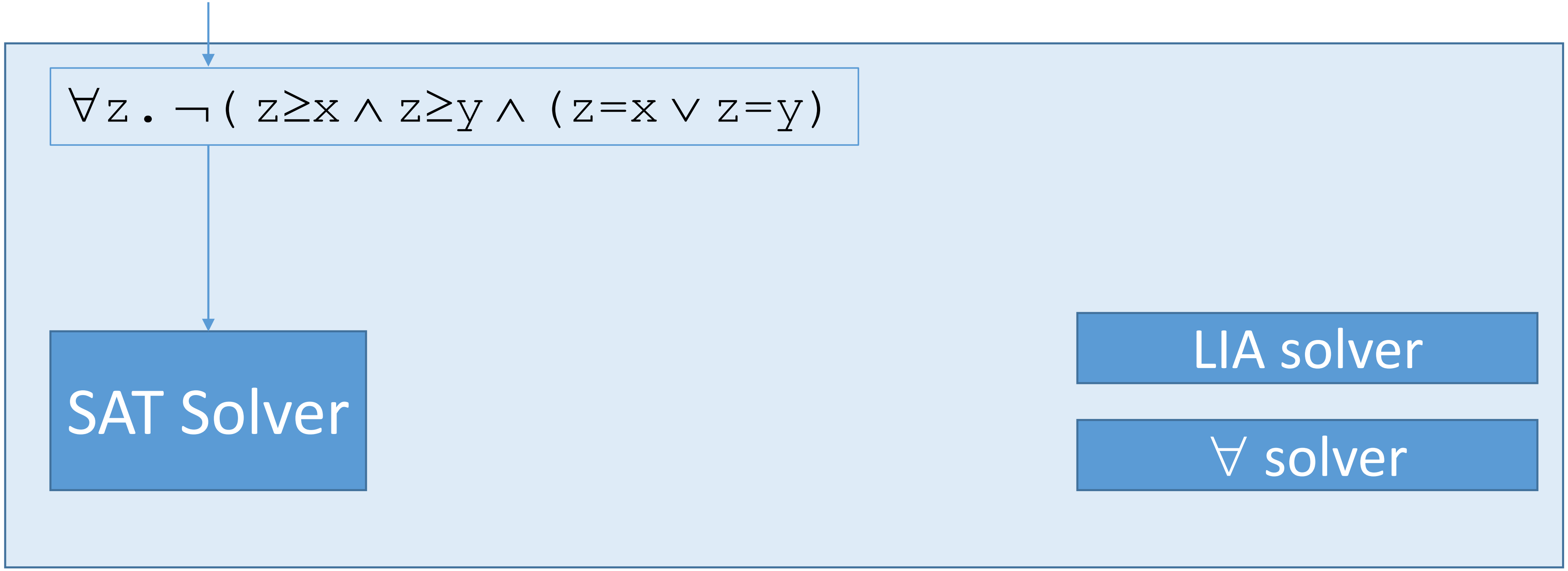
$\neg\exists f.\forall xy.\ f(x,y)\geq x \wedge f(x,y)\geq y \wedge (f(x,y)=x \vee f(x,y)=y)$



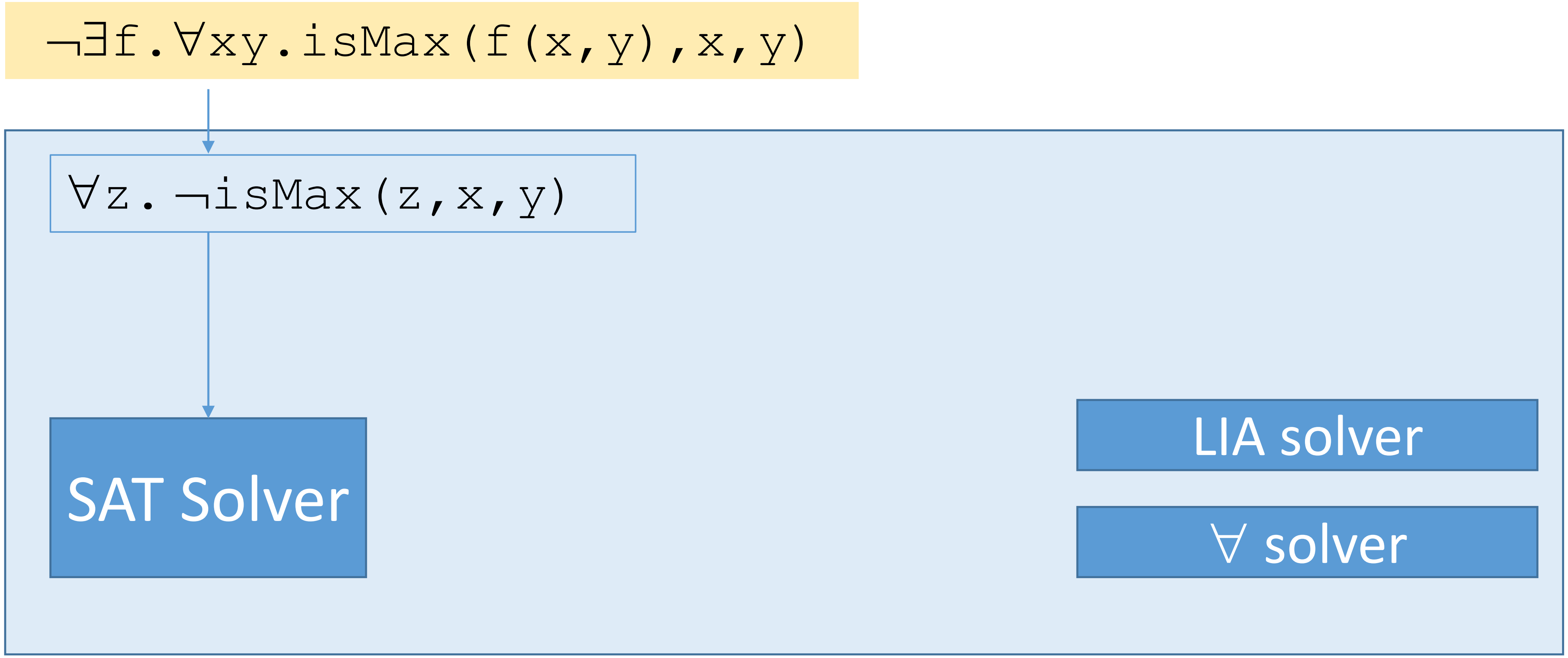
Solve use first-order ∀-instantiation for linear arithmetic (LIA)

Single Invocation Synthesis in SMT

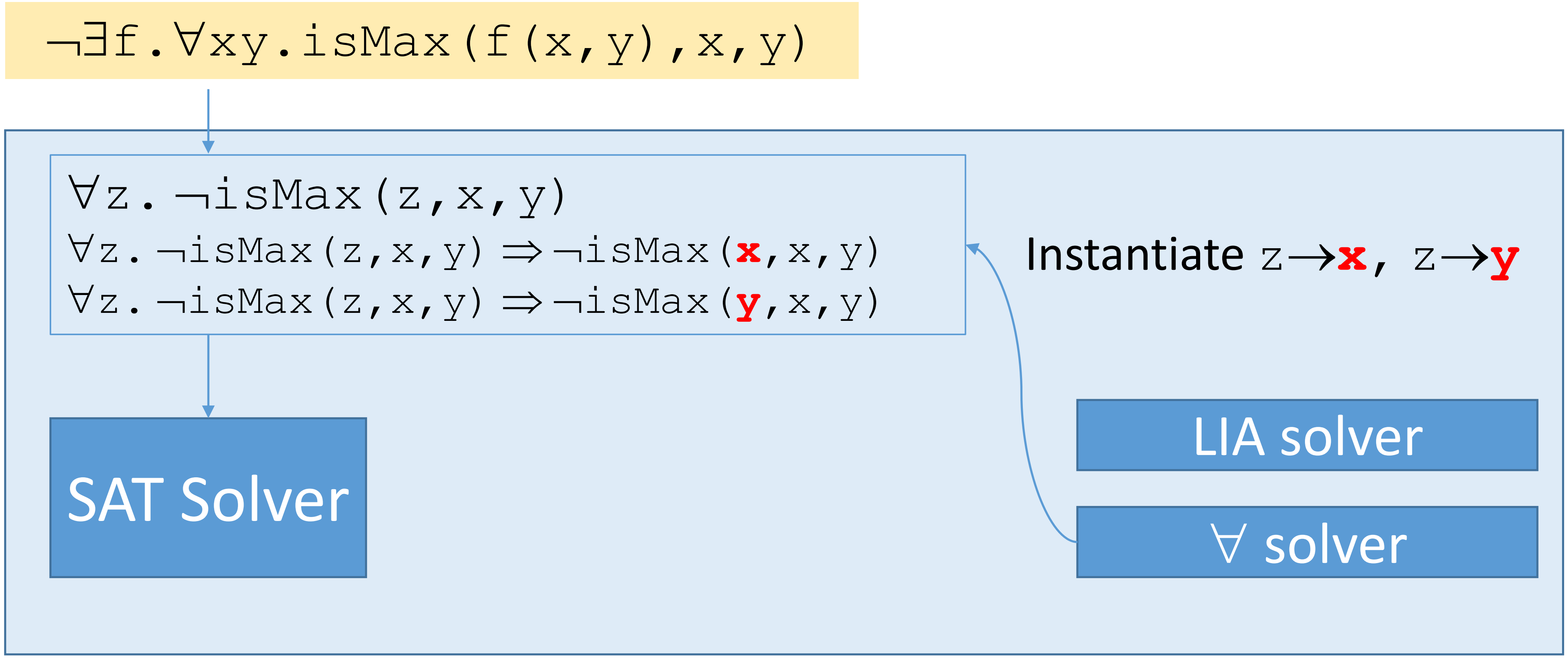
$$\neg\exists f.\forall xy.\ f(x,y)\geq x \wedge f(x,y)\geq y \wedge (f(x,y)=x \vee f(x,y)=y)$$



Single Invocation Synthesis in SMT



Single Invocation Synthesis in SMT



Single Invocation Synthesis in SMT

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

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

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

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

Simplify

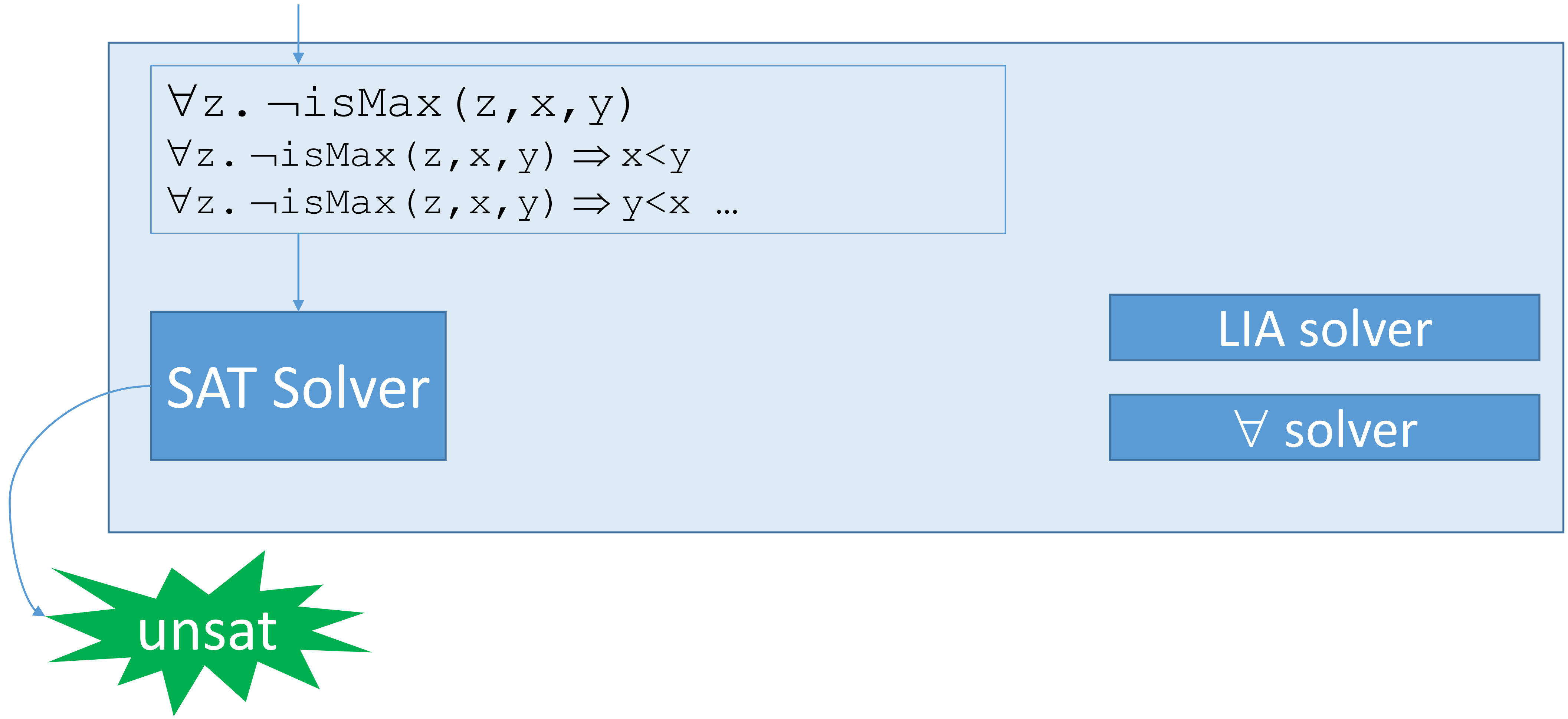
SAT Solver

LIA solver

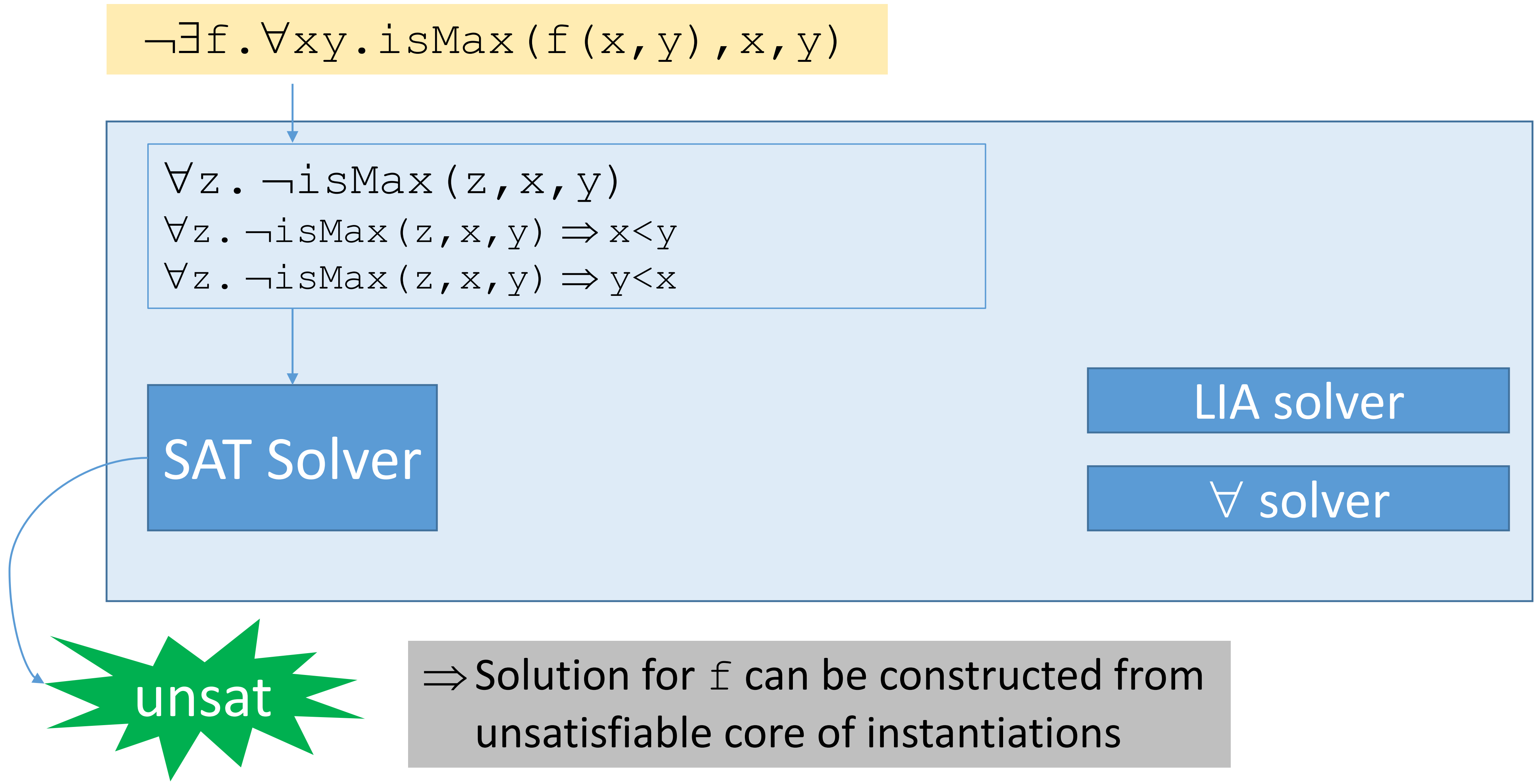
$\forall$ solver

Single Invocation Synthesis in SMT

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

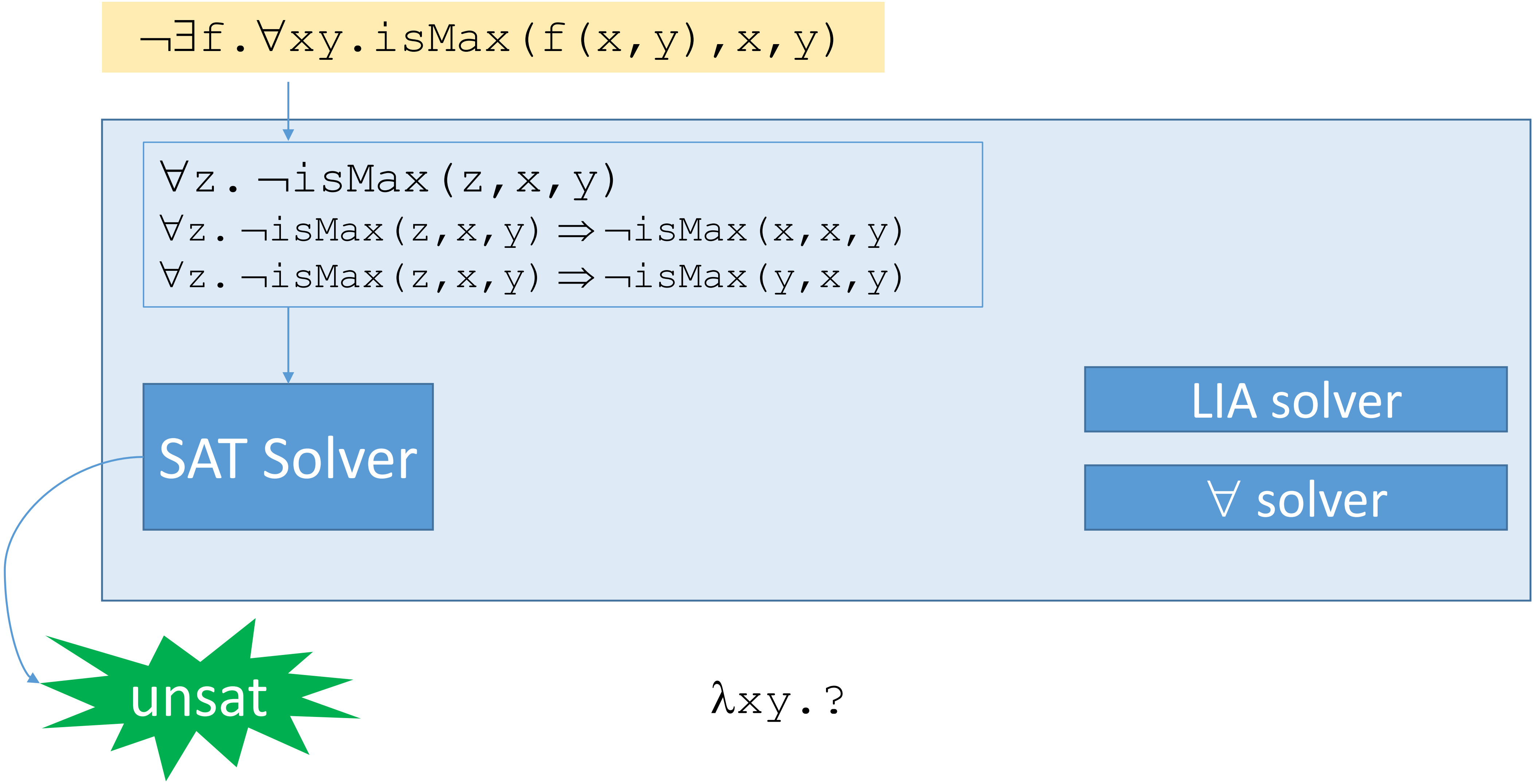


Single Invocation Synthesis in SMT

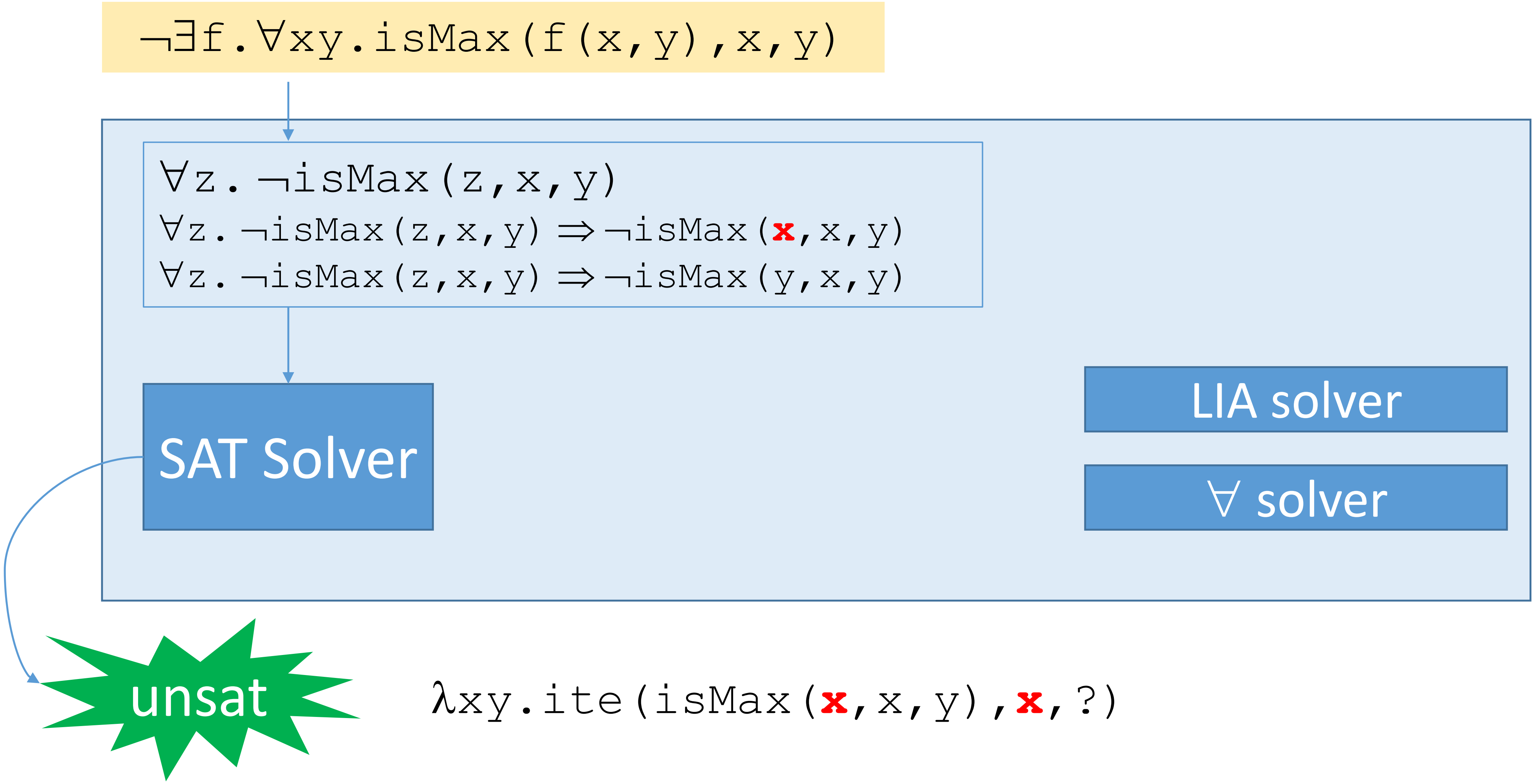


⇒ Solution for f can be constructed from unsatisfiable core of instantiations

Single Invocation Synthesis in SMT

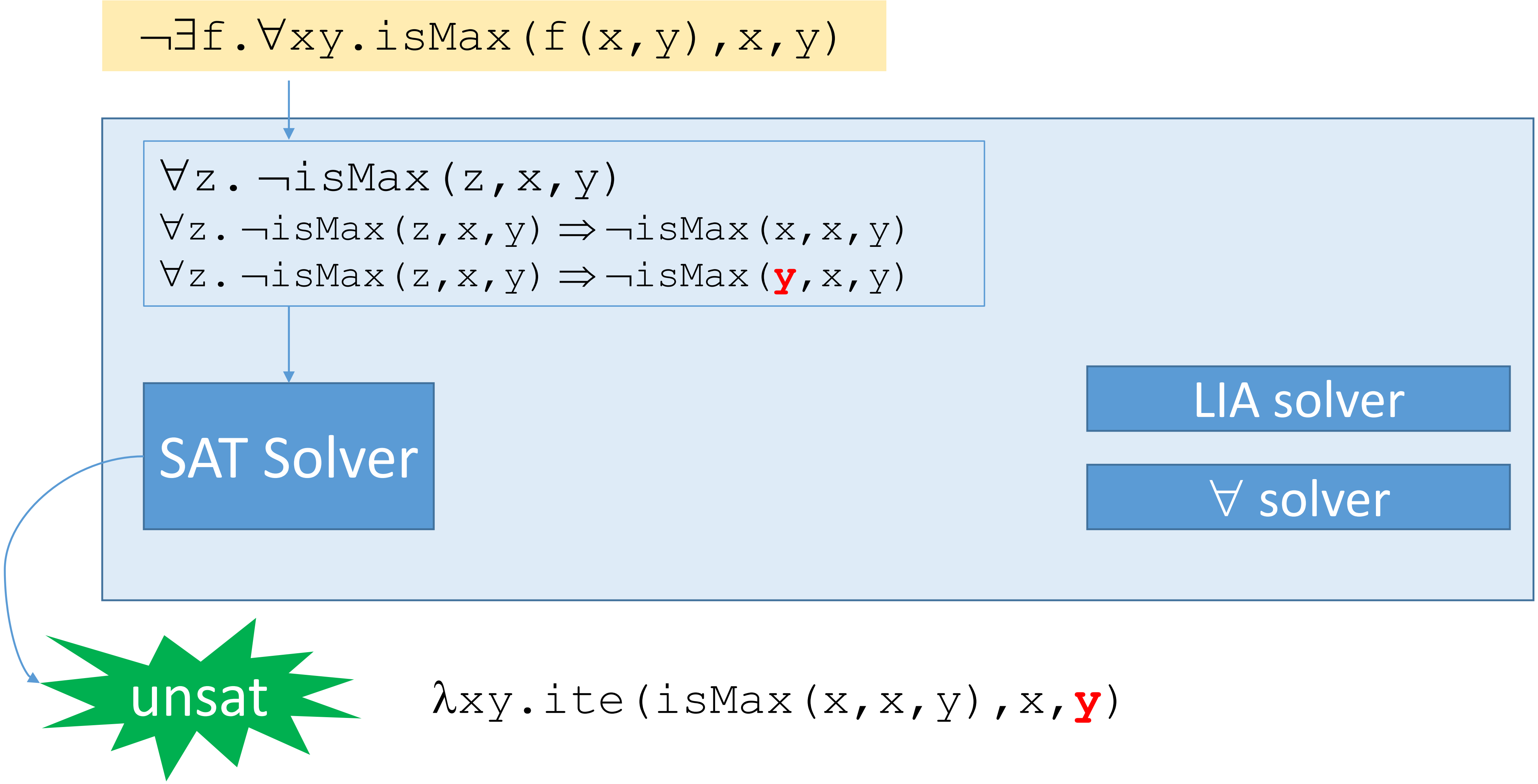


Single Invocation Synthesis in SMT

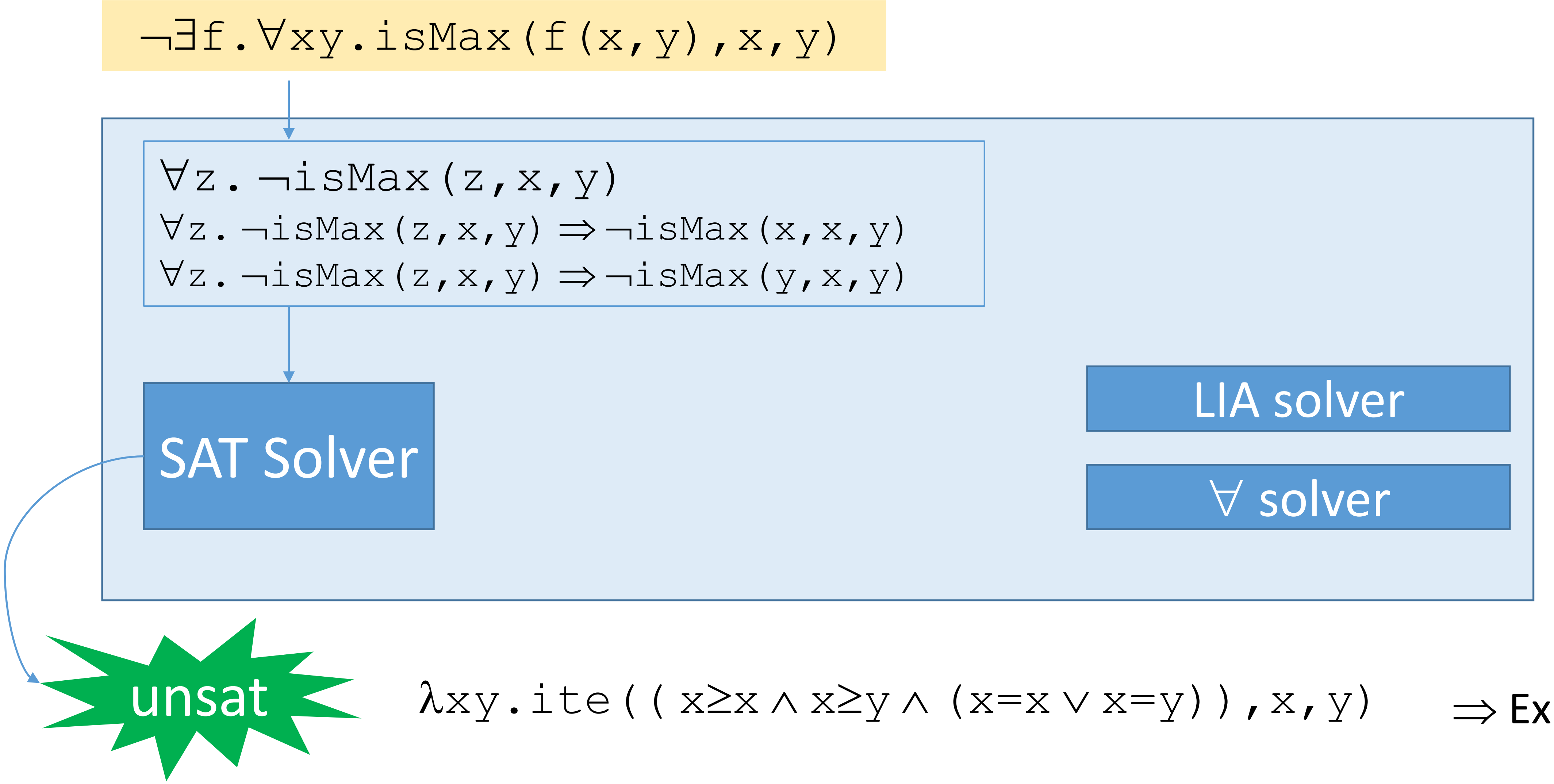


λxy.ite(isMax(**x**,x,y),**x**,?)

Single Invocation Synthesis in SMT

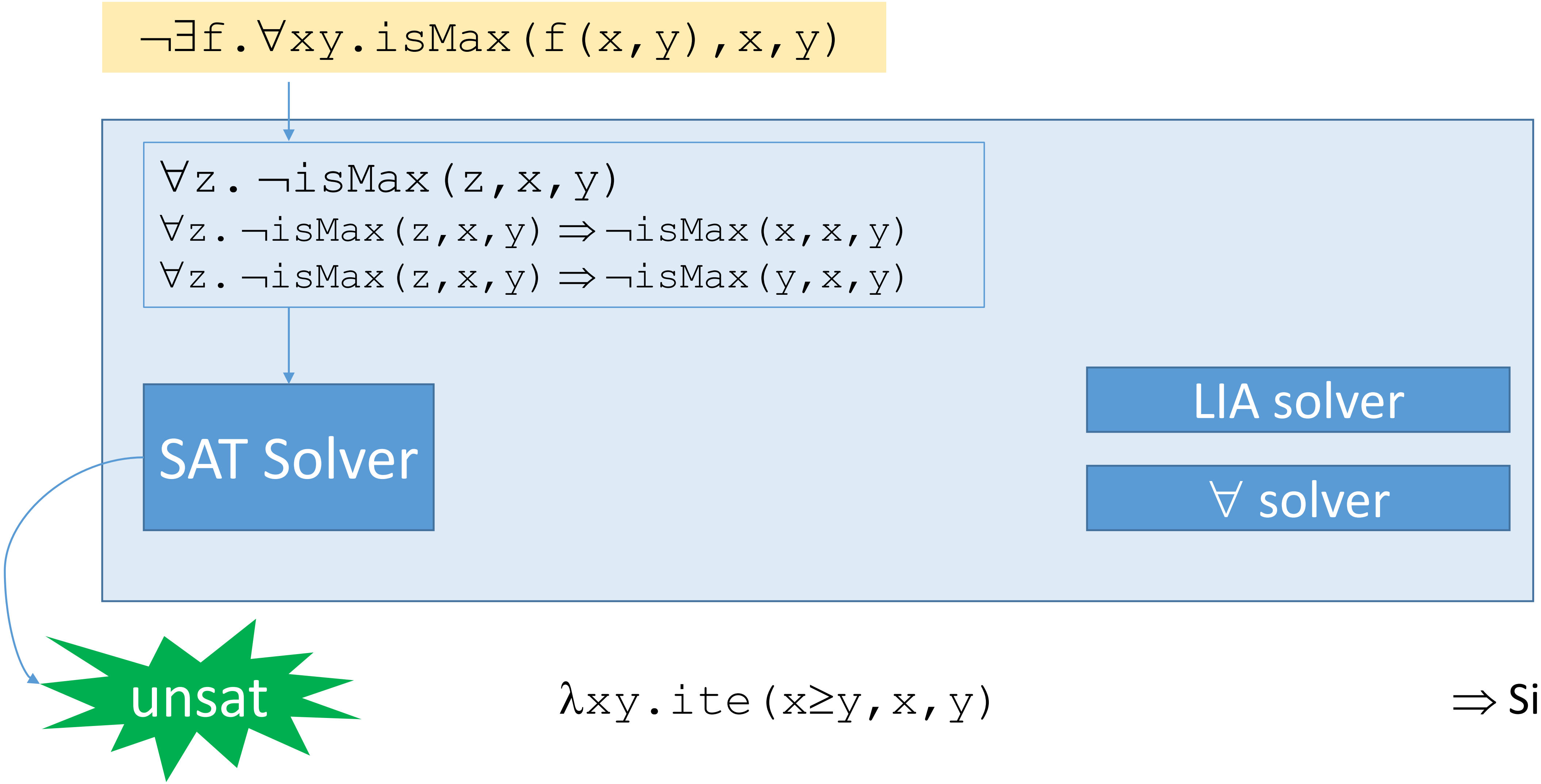


Single Invocation Synthesis in SMT



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

Single Invocation Synthesis in SMT

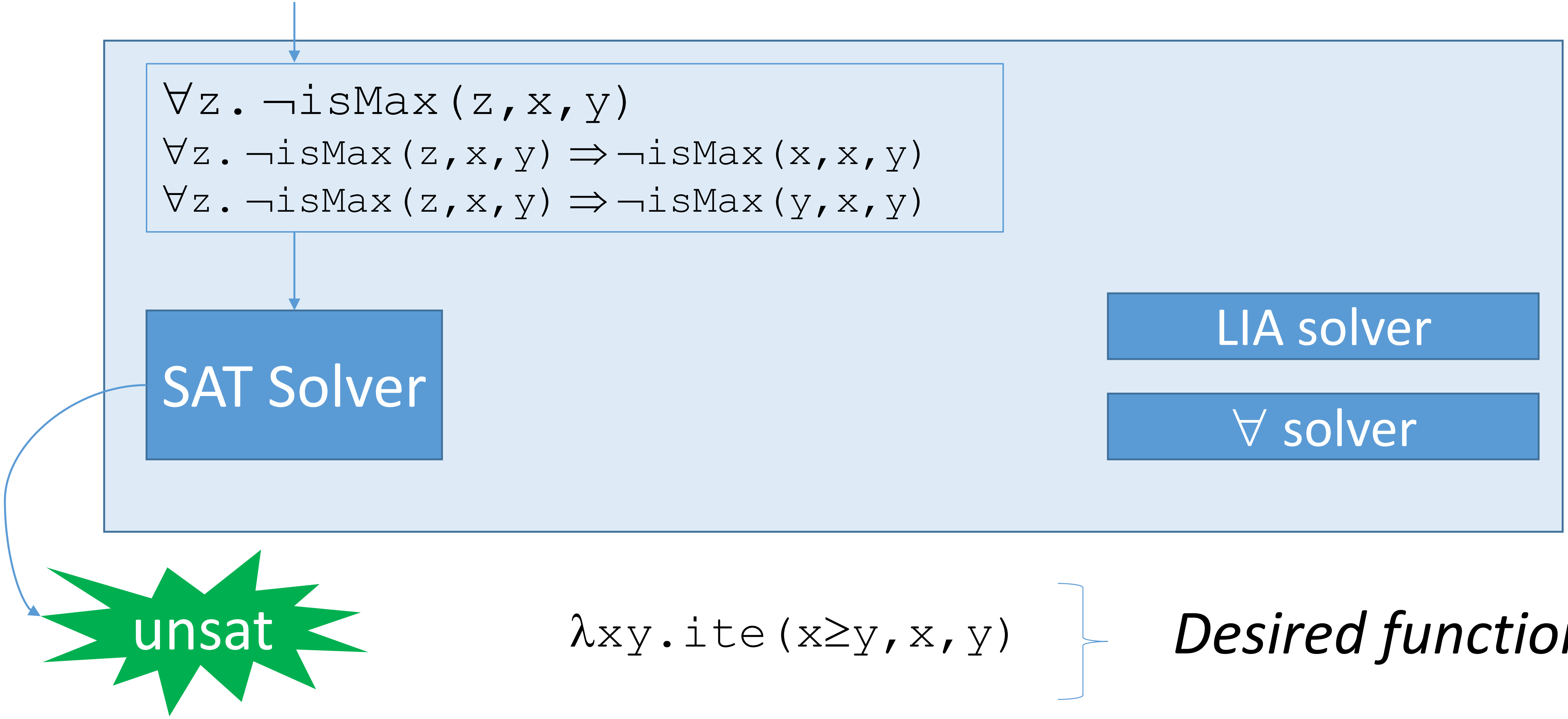


$\lambda xy.\mathrm{ite}(x \geq y, x, y)$

⇒ Simplify

Single Invocation Synthesis in SMT

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



$\lambda xy.\texttt{ite}(x \geq y, x, y)$ } *Desired function*

Single Invocation Synthesis in SMT

- Requires: method for selecting a term **?t** for instantiation

$$(\forall z.\ \neg(z \geq x \wedge z \geq y \wedge (z = x \vee z = y)) \Rightarrow$$
$$\neg(?t \geq x \wedge ?t \geq y \wedge (?t = x \vee ?t = y))$$

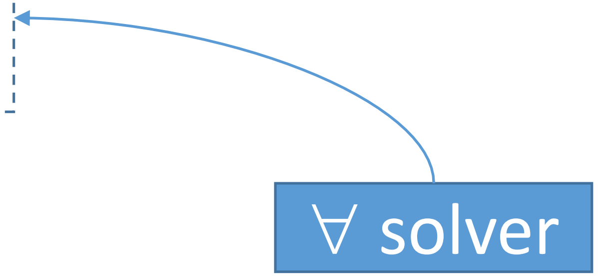
∀ solver

Single Invocation Synthesis in SMT

- Requires: method for selecting a term **?t** for instantiation
  - Use *counterexample-guided quantifier instantiation* (CEGQI)
  - General idea used in a number of related works:

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

$$(\forall z.\ \neg(z \geq x \wedge z \geq y \wedge (z=x \vee z=y)) \Rightarrow$$
$$\neg(\mathbf{?t} \geq x \wedge \mathbf{?t} \geq y \wedge (\mathbf{?t}=x \vee \mathbf{?t}=y))$$



Counterexample-Guided ∀-Instantiation

$(\forall \mathbf{z}.\ \neg(\mathbf{z} \geq x \wedge \mathbf{z} \geq y \wedge (\mathbf{z}=x \vee \mathbf{z}=y)) \Rightarrow$

$\neg(?t \geq x \wedge ?t \geq y \wedge (?t=x \vee ?t=y))$

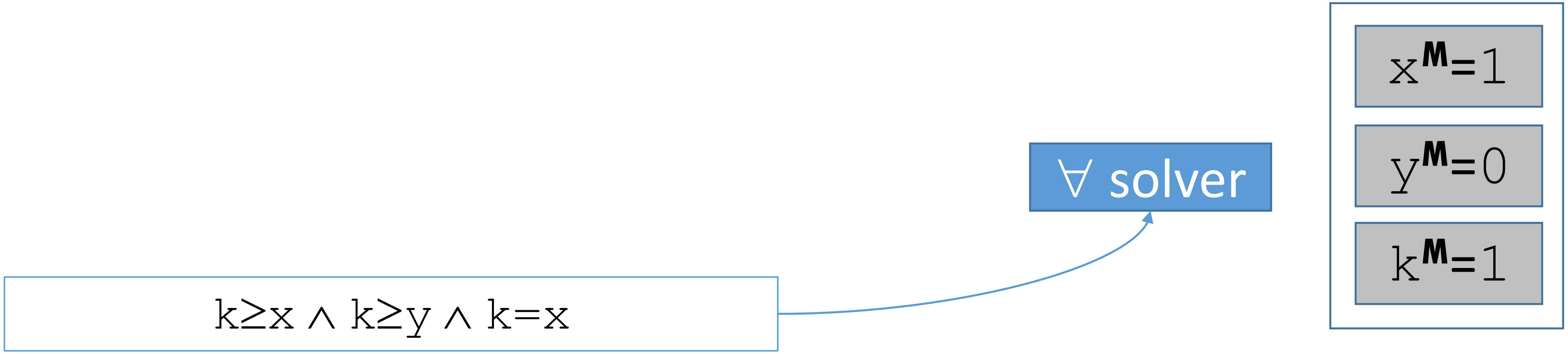
∀ solver

$\exists \mathbf{k}.\ \mathbf{k} \geq x \wedge \mathbf{k} \geq y \wedge (\mathbf{k}=x \vee \mathbf{k}=y)$

⇒ Consider conjecture's negation

- **k** is counterexample to conjecture

Counterexample-Guided ∀-Instantiation



Context satisfying negation of conjecture

Find model ***M***

Counterexample-Guided ∀-Instantiation

Consider lower bounds for k

k≥x ∧ **k≥y** ∧ k=x

∀ solver

$x^{M}=1$

$y^{M}=0$

$k^{M}=1$

Counterexample-Guided ∀-Instantiation

x is the ***maximal lower bound*** for k in ***M***

∀ solver

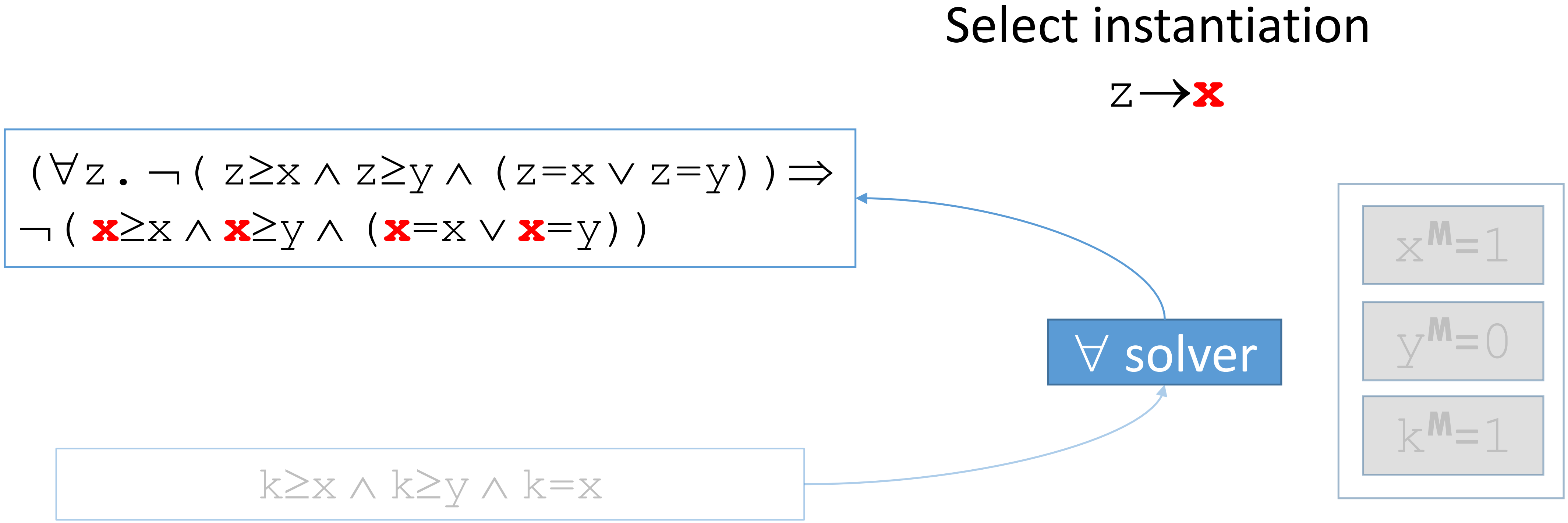
$x^M=1$

$y^M=0$

$k^M=1$

$k \geq x \land k \geq y \land k = x$

Counterexample-Guided ∀-Instantiation



Counterexample-Guided ∀-Instantiation

Requires a ***selection function*** (M,**M**, k) → x

Select instantiation

$z \rightarrow x$

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

∀ solver

$x^{\mathbf{M}}=1$

$y^{\mathbf{M}}=0$

$k^{\mathbf{M}}=1$

$k \geq x \wedge k \geq y \wedge k=x$

Counterexample-Guided ∀-Instantiation

Quantifier Elimination Procedures

$\Leftarrow(\Rightarrow)^?$

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

$\Longleftrightarrow$

Synthesis procedures for single-invocation properties

CEGQI Selection Functions

- Can devise **CEGQI selection functions** for:
  - Linear real arithmetic (LRA)
    - Maximal lower (minimal upper) bounds $l_1 < k, \ldots, l_n < k \rightarrow \{ x \rightarrow l_{max} + \delta \}$

      Analogous to **[Loos+Wiespfenning 93]** *…may involve virtual terms δ, ∞*
    - Interior point method: $l_{max} < k < u_{min} \rightarrow \{ x \rightarrow (l_{max} + u_{min}) / 2 \}$

      Analogous to **[Ferrante+Rackoff 79]**
  - Linear integer arithmetic (LIA)
    - Maximal lower (minimal upper) bounds (+c) $l_1 < k, \ldots, l_n < k \rightarrow \{ x \rightarrow l_{max} + c \}$

      Analogous to **[Cooper 72]**
  - Bitvectors/finite domains
    - (Naively) Value instantiations … $\rightarrow \{ x \rightarrow k^{\mathbf{M}} \}$
  - Datatypes, …

CEGQI Selection Functions

- Can devise **CEGQI selection functions** for:
  - Linear real arithmetic (LRA)
    - Maximal lower (minimal upper) bounds $l_1<k, \ldots, l_n<k \rightarrow \{x \rightarrow l_{max}+\delta\}$
      Analogous to **[Loos+Wiespfenning 93]** *...may involve virtual terms δ, ∞*
    - Interior point method: $l_{max}<k<u_{min} \rightarrow \{x \rightarrow (l_{max}+u_{min})/2\}$
      Analogous to **[Ferrante+Rackoff 79]**
  - Linear integer arithmetic (LIA)
    - Maximal lower (minimal upper) bounds (+c) $l_1<k, \ldots, l_n<k \rightarrow \{x \rightarrow l_{max}+c\}$
      Analogous to **[Cooper 72]**
  - Bitvectors/finite domains
    - (Naively) Value instantiations … $\rightarrow \{x \rightarrow k^{\mathbf{M}}\}$
  - Datatypes, …

$\Rightarrow$ *Each gives a **sound** and **complete** procedure for single invocation conjectures*

CEGQI in **CVC4**

- Highly efficient:
  - for synthesis:
    - CVC4 won SyGuS-Comp GENERAL track 2015, CLIA track 2015-2016
  - and also for automated theorem proving:
    - CVC4 won TFA division of CASC 2014, LIA/LRA divisions of SMT COMP 2014-2016
- Applicable to multiple-function conjectures $\exists f g . \forall x . P(f(x), g(x), x)$
- **Sound** and **complete** for single invocation conjectures over LIA, LRA

  See **[Reynolds/King/Kuncak FMSD2017, to appear]**
- **Disadvantage**: leads to verbose solutions

Overview

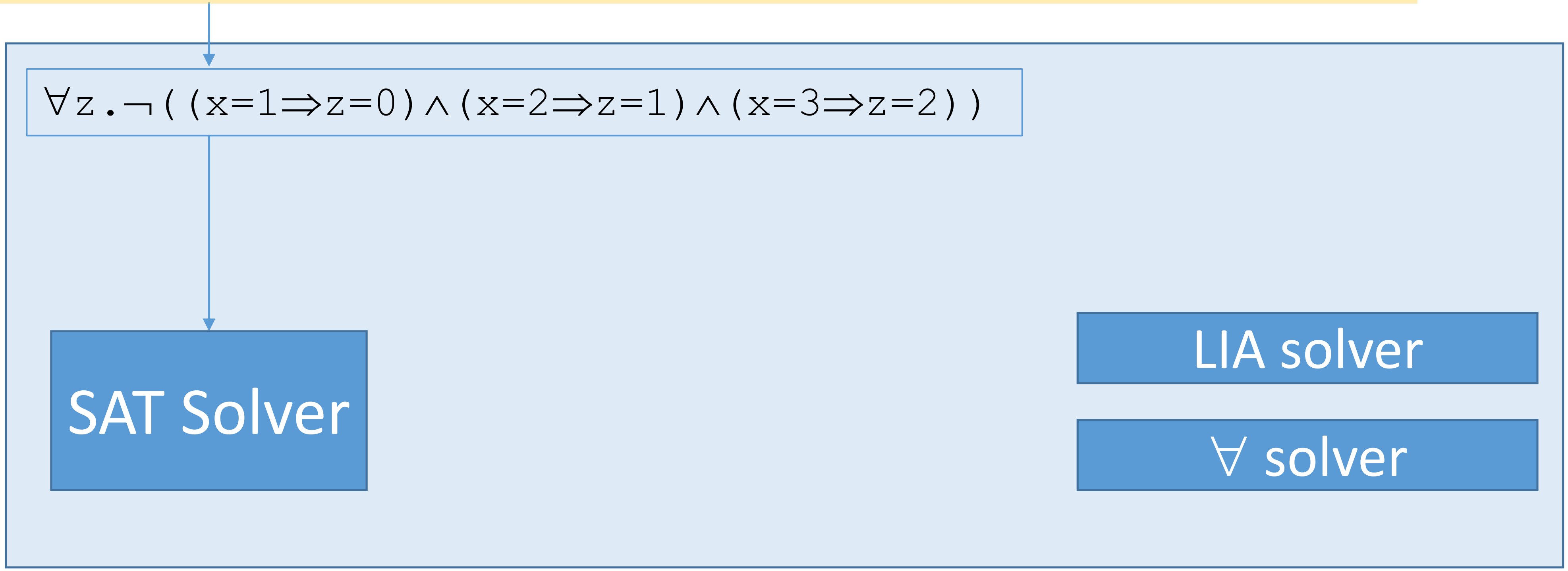
	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	?	?	?
Without Syntactic Restrictions	?	Counterexample Guided ∀-Instantiation	?

What if we apply CEGQI to I/O Examples?

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

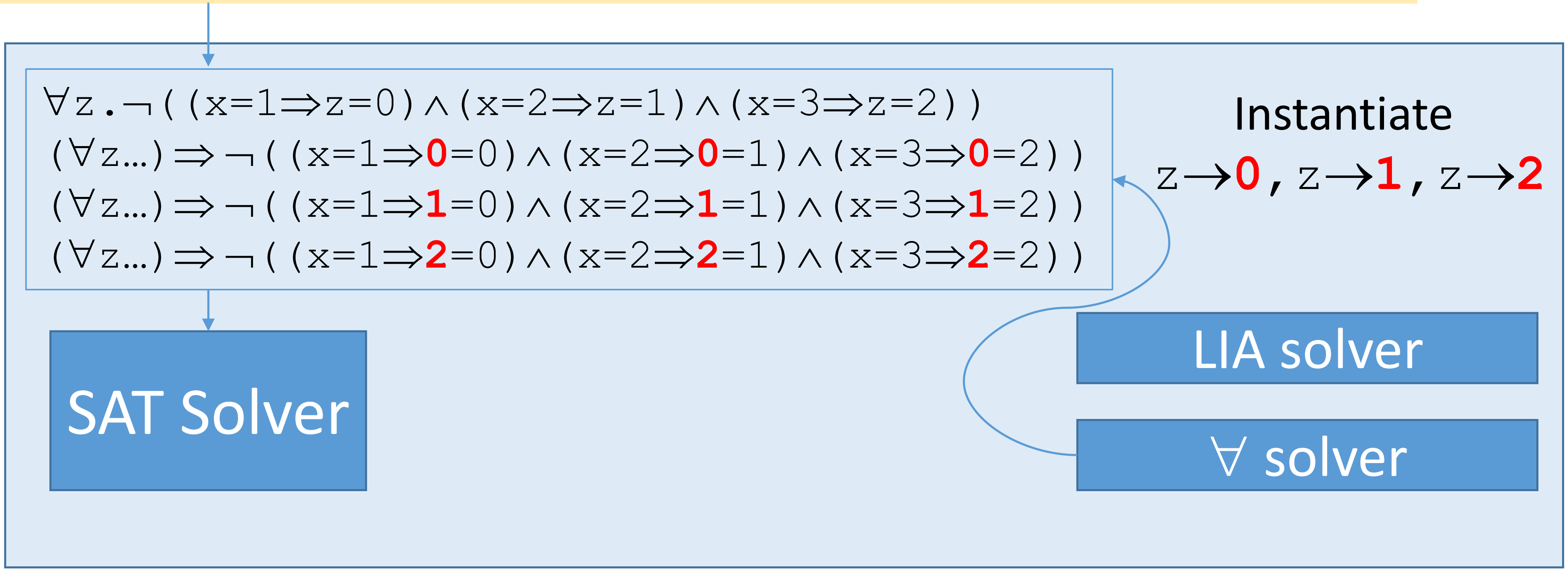
What if we apply CEGQI to I/O Examples?

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



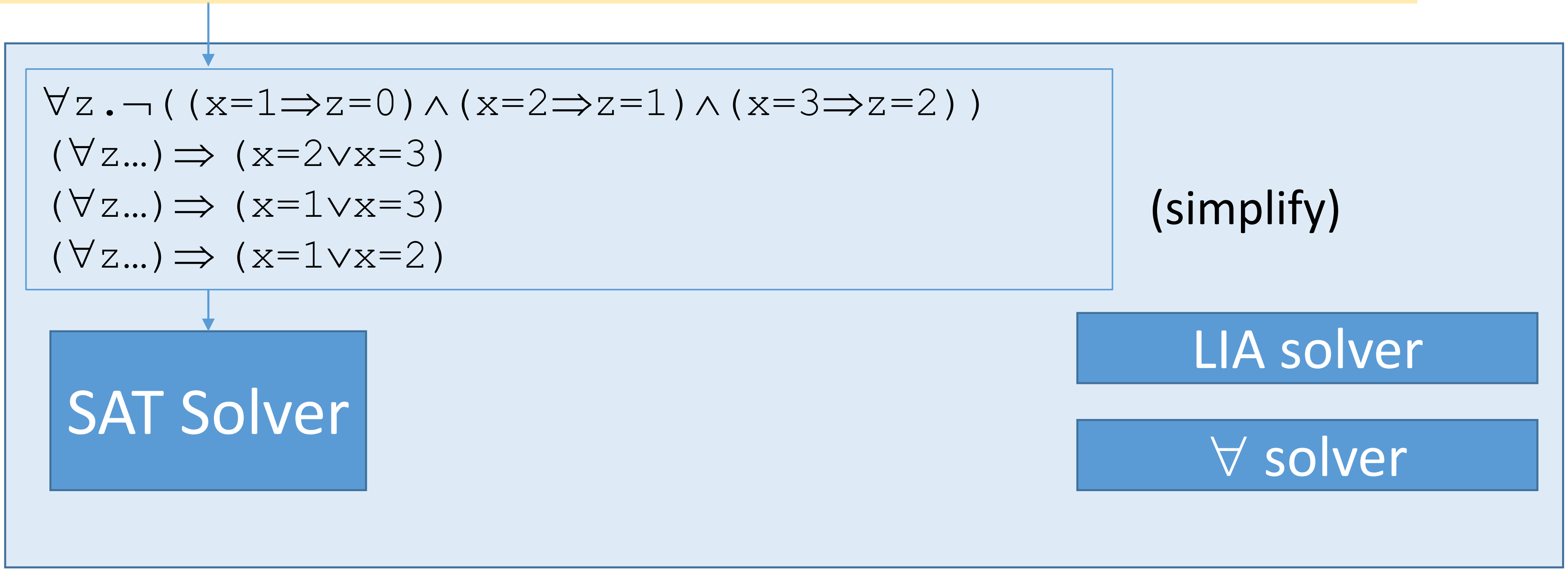
What if we apply CEGQI to I/O Examples?

¬∃f.∀x.(x=1⇒f(x)=0)∧(x=2⇒f(x)=1)∧(x=3⇒f(x)=2)



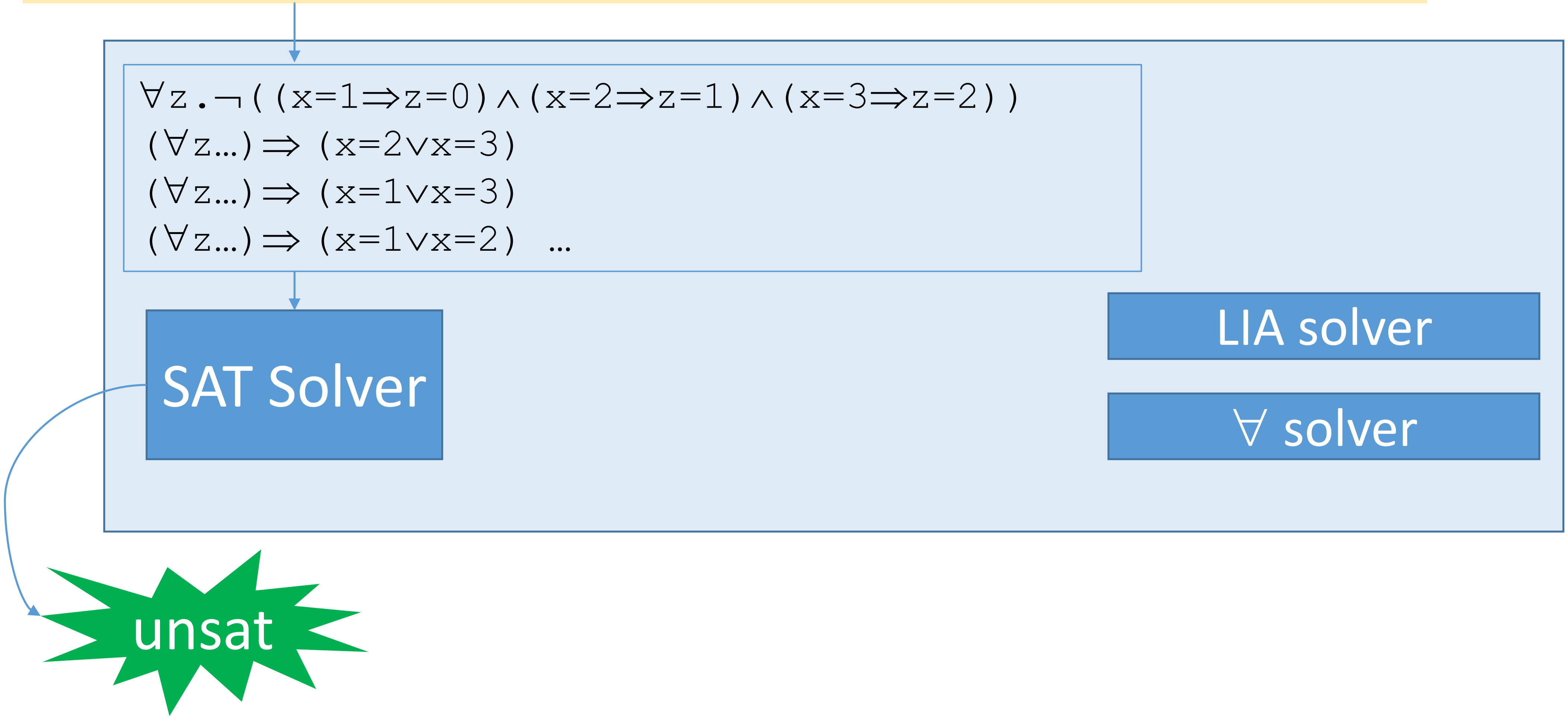
What if we apply CEGQI to I/O Examples?

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

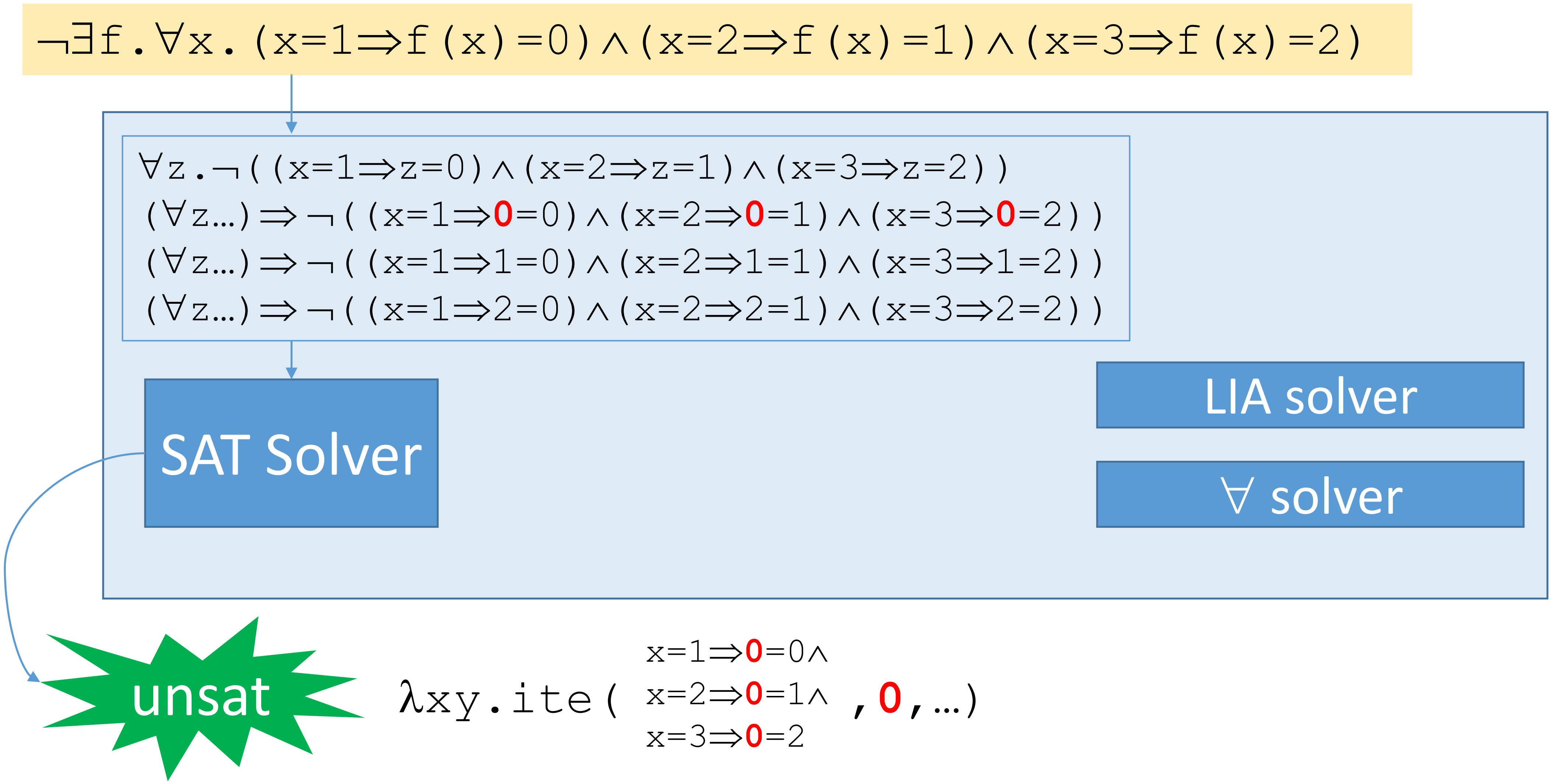


What if we apply CEGQI to I/O Examples?

¬∃f.∀x.(x=1⇒f(x)=0)∧(x=2⇒f(x)=1)∧(x=3⇒f(x)=2)

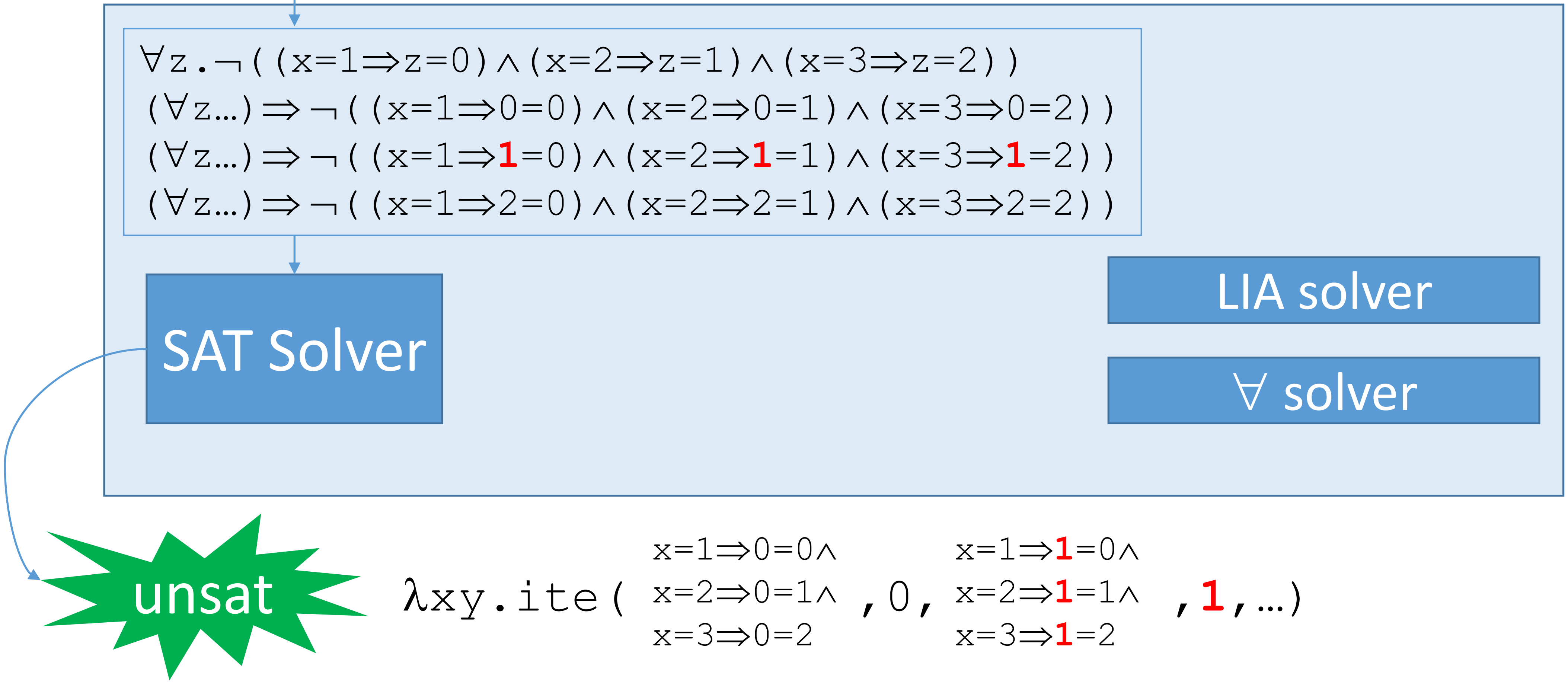


What if we apply CEGQI to I/O Examples?



What if we apply CEGQI to I/O Examples?

¬∃f.∀x.(x=1⇒f(x)=0)∧(x=2⇒f(x)=1)∧(x=3⇒f(x)=2)



What if we apply CEGQI to I/O Examples?

¬∃f.∀x.(x=1⇒f(x)=0)∧(x=2⇒f(x)=1)∧(x=3⇒f(x)=2)

∀z.¬((x=1⇒z=0)∧(x=2⇒z=1)∧(x=3⇒z=2))

(∀z…)⇒¬((x=1⇒0=0)∧(x=2⇒0=1)∧(x=3⇒0=2))

(∀z…)⇒¬((x=1⇒1=0)∧(x=2⇒1=1)∧(x=3⇒1=2))

(∀z…)⇒¬((x=1⇒**2**=0)∧(x=2⇒**2**=1)∧(x=3⇒**2**=2))

SAT Solver

LIA solver

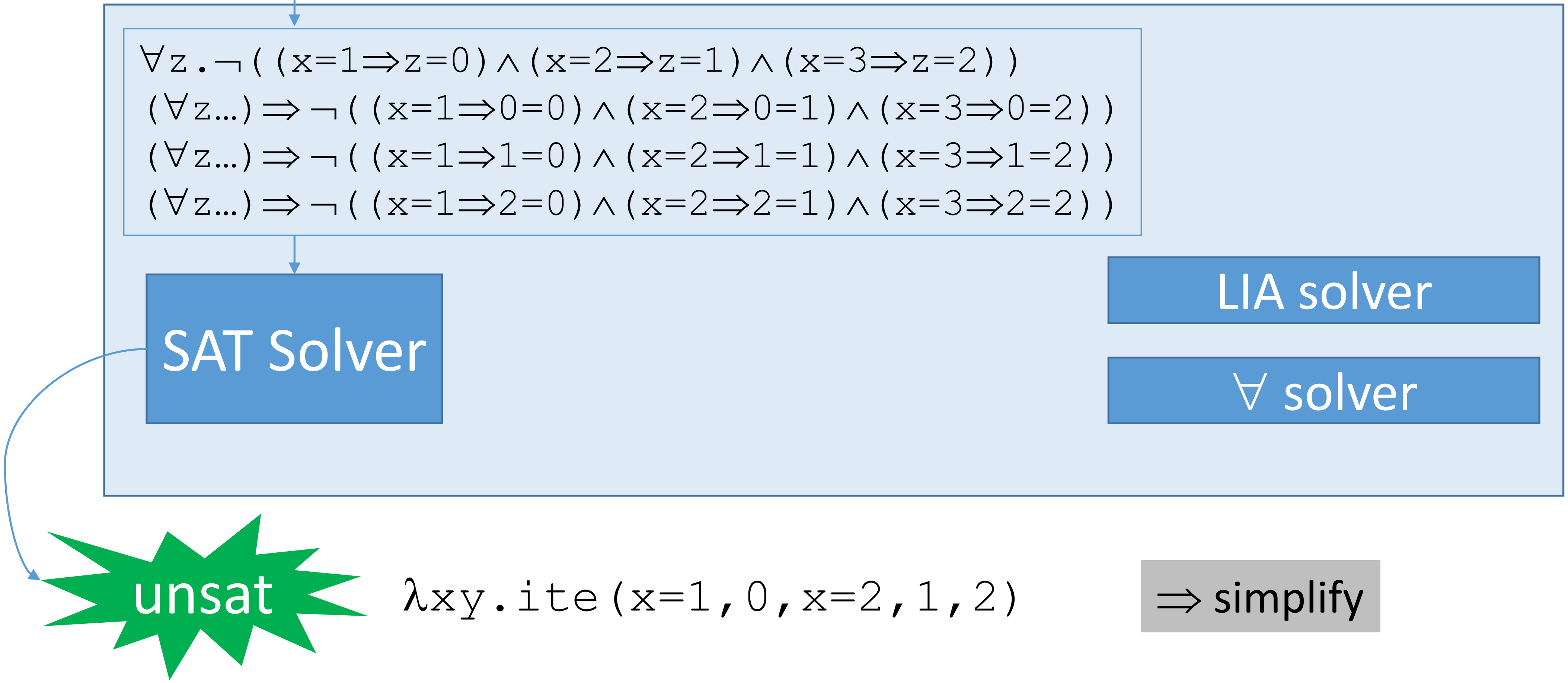
∀ solver

unsat

λxy.ite(x=1⇒0=0∧ x=2⇒0=1∧ x=3⇒0=2 ,0, x=1⇒1=0∧ x=2⇒1=1∧ x=3⇒1=2 ,1,**2**)

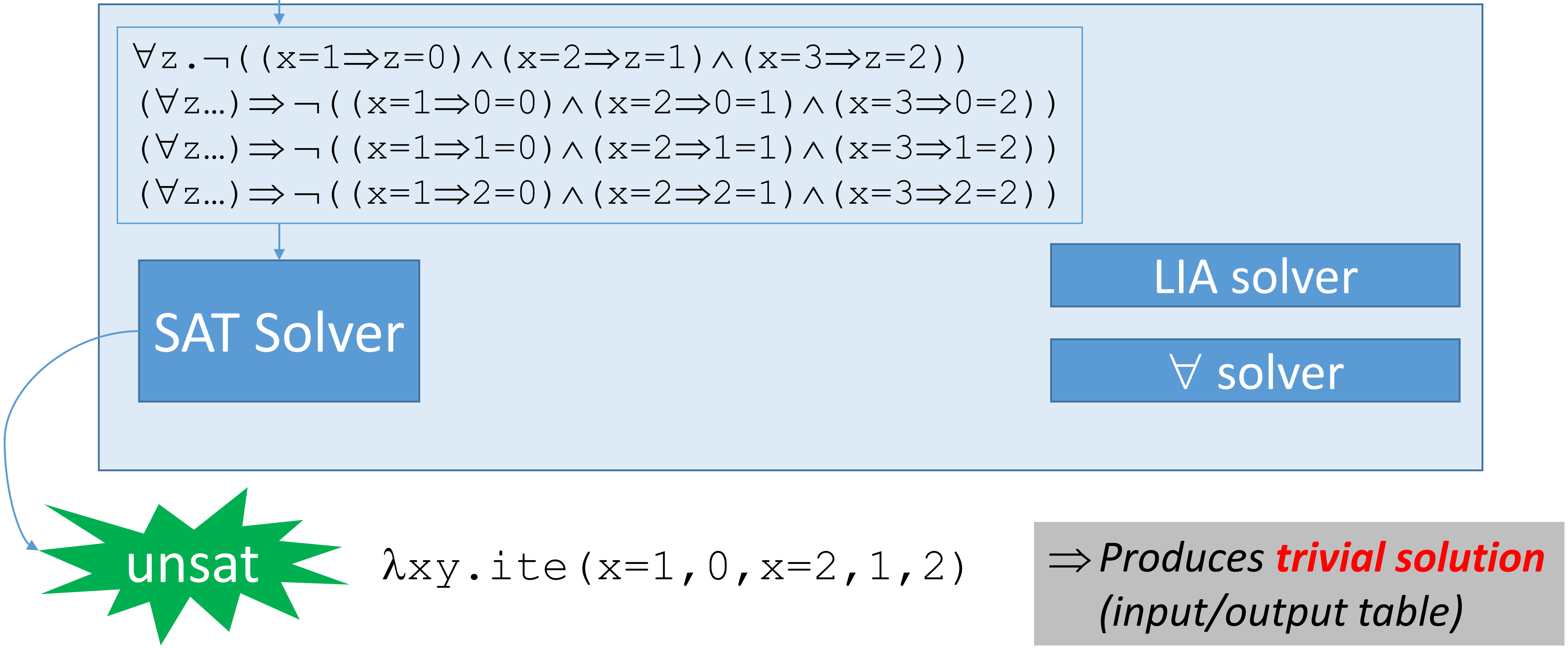
What if we apply CEGQI to I/O Examples?

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



What if we apply CEGQI to I/O Examples?

¬∃f.∀x.(x=1⇒f(x)=0)∧(x=2⇒f(x)=1)∧(x=3⇒f(x)=2)



Overview

	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	? …although prefer solutions here	?	?
Without Syntactic Restrictions	CEGQI (trivially)	Counterexample Guided ∀-Instantiation	?

What if there are syntactic restrictions?

```
¬∃f.∀xy.isMax(f(x,y),x,y)
```

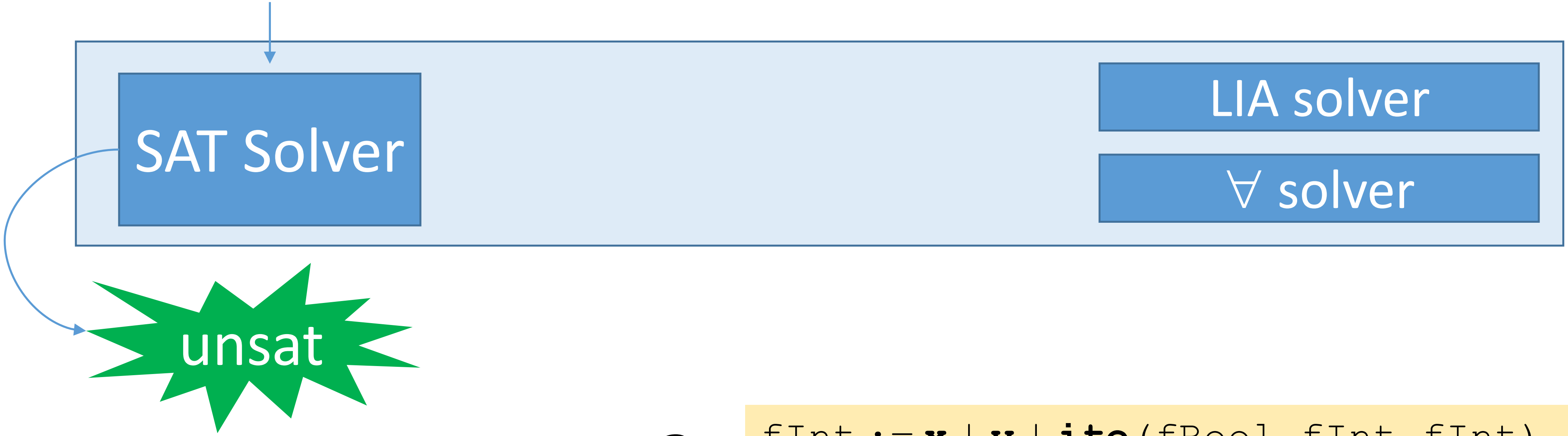
where solution meets syntactic restrictions $\mathfrak{R}$:

$\mathfrak{R}$:

```
fInt  := x | y | ite(fBool,fInt,fInt)
fBool := >(fInt,fInt) | =(fInt,fInt) | ¬(fBool)
```

What if there are syntactic restrictions?

¬∃f.∀xy.isMax(f(x,y),x,y)



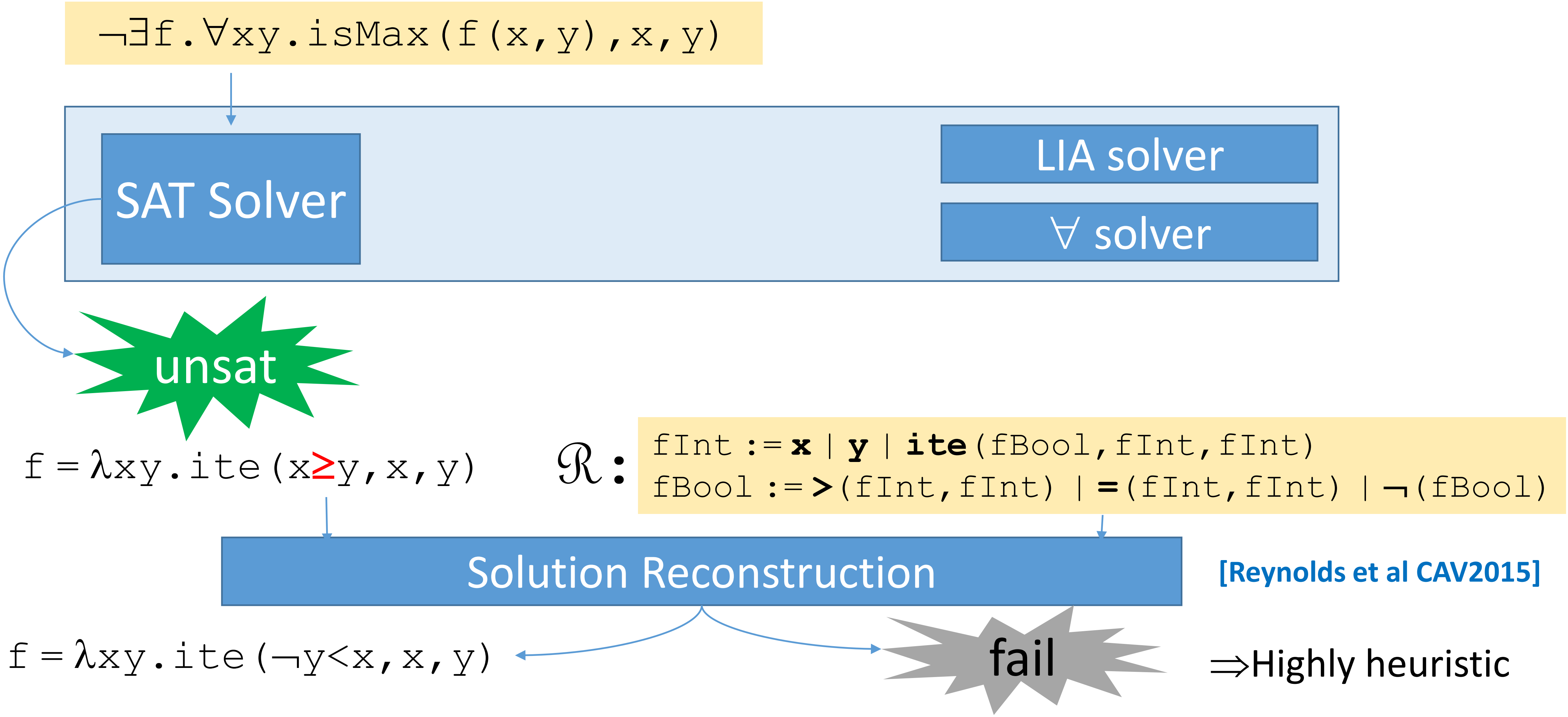
f = λxy.ite(x≥y,x,y)

ℛ:

fInt := **x** | **y** | **ite**(fBool,fInt,fInt)

fBool := **>**(fInt,fInt) | **=**(fInt,fInt) | **¬**(fBool)

What if there are syntactic restrictions?



Overview

	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	?	? CEGQI + reconstruction	?
Without Syntactic Restrictions	CEGQI (trivially)	Counterexample Guided $\forall$-Instantiation	?

For Everything Else…

	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	Enumerative SyGuS	Enumerative SyGuS CEGQI + reconstruction	Enumerative SyGuS
Without Syntactic Restrictions	CEGQI (trivially)	Counterexample Guided ∀-Instantiation	Enumerative SyGuS (using default restrictions)

Enumerative Syntax-Guided Synthesis

Conjecture

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

Test ←

```
0
1
x
1+1
x+1
x+x
ite(x>0,0,1)
.
.
.
```

← Enumerate

Syntactic Restrictions $\mathfrak{R}$

fInt := **x** | **0** | **1** | **+**(fInt,fInt) | **ite**(fBool,fInt,fInt)

fBool := **>**(fInt,fInt) | **=**(fInt,fInt) | **¬**(fBool)

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

Enumerative Syntax-Guided Synthesis in SMT

Conjecture

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

Test ←

```
0
1
x
1+1
x+1
x+x
ite(x>0,0,1)
.
.
.
```

← Enumerate

Syntactic Restrictions $\mathfrak{R}$

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

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

$\Rightarrow$ **In this talk:** how this approach can be integrated in a DPLL(T) SMT solver

Enumerative Syntax-Guided Synthesis in SMT

Conjecture

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

$\mathcal{R}$:

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

Enumerative Syntax-Guided Synthesis in SMT

Conjecture

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

$\mathcal{R}$:

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

Construct *inductive datatypes* I,B corresponding to $\mathcal{R}$

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I)
```

$\Rightarrow$ Involves flattening, minimization

Enumerative Syntax-Guided Synthesis in SMT

Conjecture

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

Int × Int → Int

$\mathfrak{R}$:

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

Encode conjecture using *deep embedding* involving I

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

I

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I)

Enumerative Syntax-Guided Synthesis in SMT

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: $I \times Int \times Int \rightarrow Int$

$\mathcal{R}$:

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

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I)

- **E**(d,x,y) *evaluates* d for arguments x,y, e.g. E(**+**(**x**,**y**),2,3)=5

Enumerative Syntax-Guided Synthesis in SMT

Conjecture

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

$\mathcal{R}$:

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

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

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I)
```

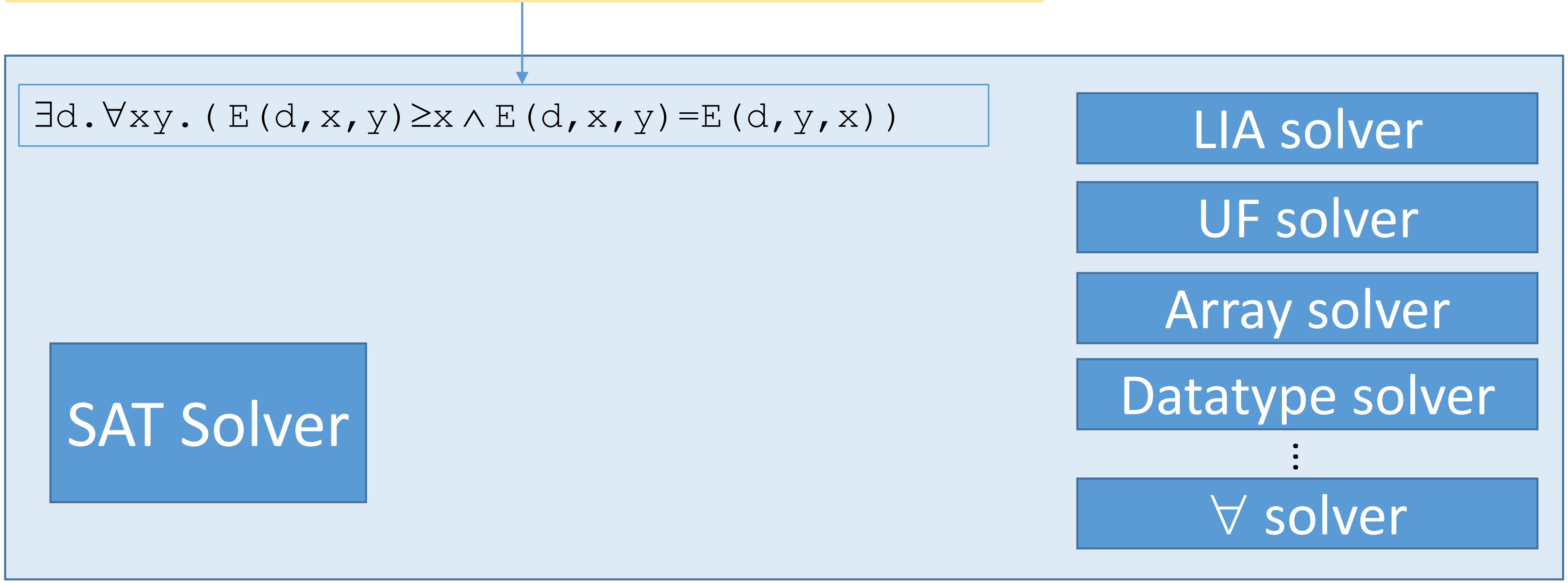
$\Rightarrow$ *Solvable by combination of datatypes, LIA, UF, and $\forall$-instantiation*

Enumerative Syntax-Guided Synthesis in SMT

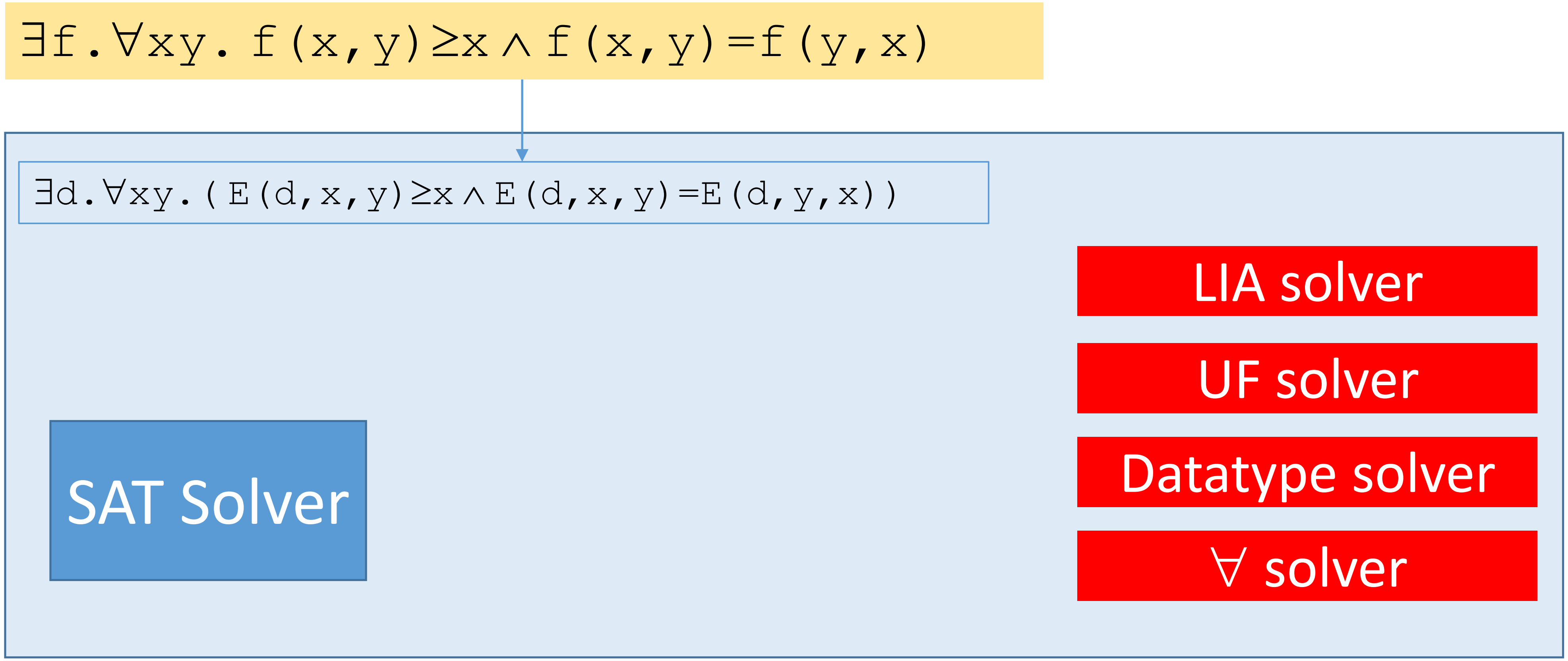
$\exists f.\forall xy.\ f(x,y) \geq x \wedge f(x,y) = f(y,x)$

Enumerative Syntax-Guided Synthesis in SMT

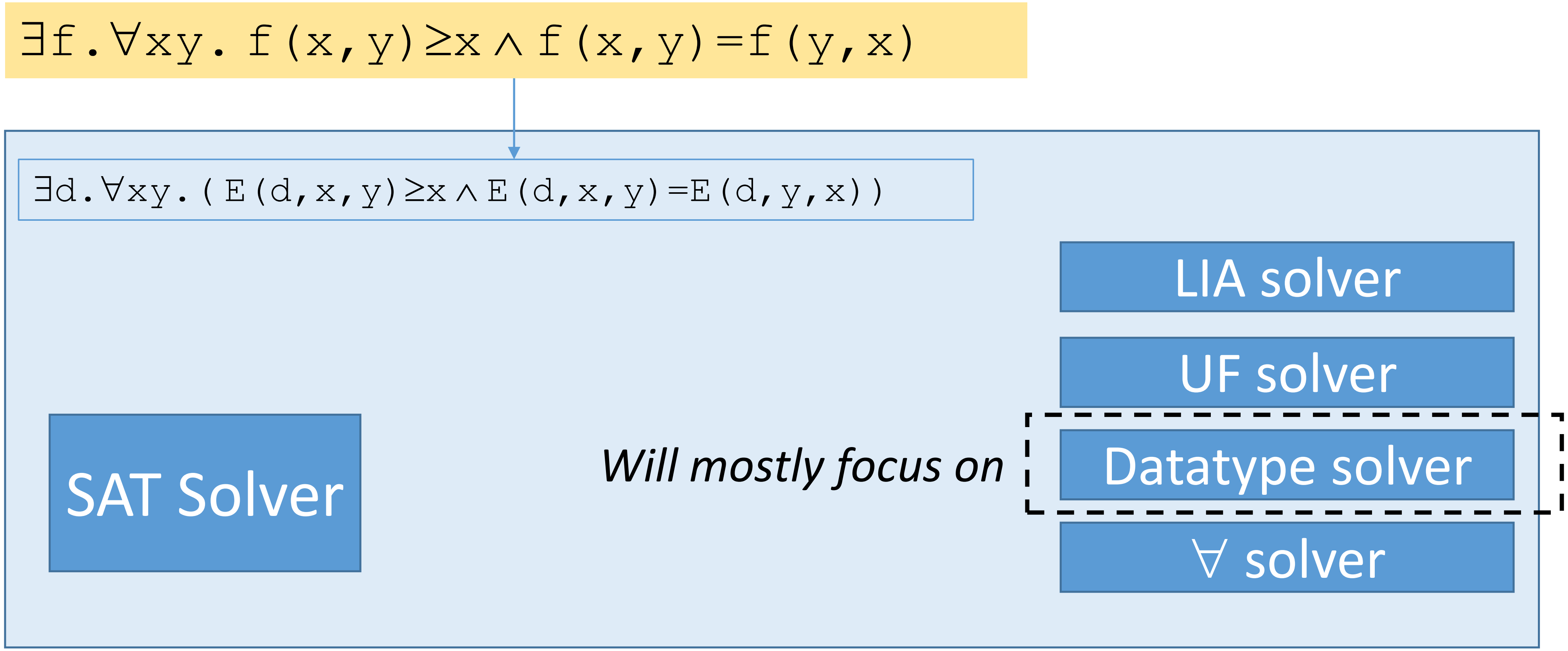
$\exists f. \forall xy.\ f(x,y) \geq x \wedge f(x,y) = f(y,x)$



Enumerative Syntax-Guided Synthesis in SMT



Enumerative Syntax-Guided Synthesis in SMT



Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

- **Idea:** Return candidate solutions based on value of $d^{\mathbf{M}}$ in models $\mathbf{M}$ where `d` is of type `I`

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | ¬(B)

$is_x(d) \lor is_y(d) \lor is_0(d) \lor is_1(d) \lor is_+(d) \lor is_{ite}(d)$

Split on top symbol of d

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

$\underline{\mathbf{is}_{x}\mathbf{(d)}} \vee is_{y}(d) \vee is_{0}(d) \vee is_{1}(d) \vee is_{+}(d) \vee is_{ite}(d)$

SAT Solver

Datatype solver

$is_{x}(d)$

Find satisfying context

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$

SAT Solver

Datatype solver

$is_x(d)$

$d^{M} = x$

Find model ***M***

Enumerative Syntax-Guided Synthesis in SMT

$I := \mathbf{x} \mid \mathbf{y} \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{+}(I,I) \mid \mathbf{ite}(B,I,I)$
$B := \geq(I,I) \mid =(I,I) \mid \neg(B)$

$is_x(d) \lor is_y(d) \lor is_0(d) \lor is_1(d) \lor is_+(d) \lor is_{ite}(d)$
$is_x(d) \Rightarrow \forall xy.(E(\mathbf{x},x,y) \geq x \land E(\mathbf{x},x,y) = E(\mathbf{x},y,x))$

Check conjecture for candidate $d \rightarrow \mathbf{x}$

SAT Solver

Datatype solver

$\forall$ solver

$d^M = \mathbf{x}$

Enumerative Syntax-Guided Synthesis in SMT

$I := \mathbf{x} \mid \mathbf{y} \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{+}(I,I) \mid \mathbf{ite}(B,I,I)$
$B := \mathbf{\geq}(I,I) \mid \mathbf{=}(I,I) \mid \mathbf{\neg}(B)$

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$
$is_x(d) \Rightarrow \forall xy.(x \geq x \wedge x = y)$

Simplify

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$

$\neg is_x(d)$

Simplify

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

$I := \mathbf{x} \mid \mathbf{y} \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{+}(I,I) \mid \mathbf{ite}(B,I,I)$
$B := \mathbf{\geq}(I,I) \mid \mathbf{=}(I,I) \mid \neg(B)$

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$
$\neg is_x(d)$

SAT Solver

Datatype solver

$\neg is_x(d)$
$is_y(d)$

Find next satisfying context

Enumerative Syntax-Guided Synthesis in SMT

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$
$\neg is_x(d)$

SAT Solver

Datatype solver

$d^M = y$

$\neg is_x(d)$
$is_y(d)$

Find next model ***M***

Enumerative Syntax-Guided Synthesis in SMT

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$
$\neg is_x(d)$
$\neg is_y(d)$

SAT Solver

Datatype solver

∀ solver

$d^M = \ldots$

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$

$\neg is_x(d)$

$\neg is_y(d)$

...

$is_x(d.1) \vee is_y(d.1) \vee is_0(d.1) \vee is_1(d.1) \vee is_+(d.1) \vee is_{ite}(d.1)$

$is_x(d.2) \vee is_y(d.2) \vee is_0(d.2) \vee is_1(d.2) \vee is_+(d.2) \vee is_{ite}(d.2)$

$\neg is_+(d) \vee \neg is_x(d.1) \vee \neg is_x(d.2)$

…

…and repeat

SAT Solver

Datatype solver

∀ solver

$d^M = \ldots$

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

$is_x(d) \vee is(d) \vee is(d) \vee is(d) \vee is(d) \vee is(d)$

$\neg is_x($

…

Optimization:

Only consider terms d^M *whose* ***analog***
is unique up to theory-specific simplification $\downarrow$

SAT Solver

Datatype solver

∀ solver

$d^M = \ldots$

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

$is_x(d) \vee is_y(d) \vee is_0(d) \vee is_1(d) \vee is_+(d) \vee is_{ite}(d)$
$\neg is_x($
…

SAT

Optimization:

Only consider terms d^M *whose* ***analog*** *is unique up to theory-specific simplification* $\downarrow$

Enumerate

x	…	x	$=\downarrow$	x
y	…	y	$=\downarrow$	y
+(1,y)	…	1+y	$=\downarrow$	y+1
+(x,0)	…	x+0	$=\downarrow$	x
+(y,1)	…	y+1	$=\downarrow$	y+1

=…

Enumerative Syntax-Guided Synthesis in SMT

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

$is_x(d) \vee is$ … $(d) \vee is$ … $(d) \vee is$ … $(d) \vee is$ … $(d) \vee is$ … (d)
$\neg is_x($ …
…

SAT

…=…

Optimization:

Only consider terms d^M *whose* ***analog***
is unique up to theory-specific simplification ↓

Enumerate ↓

x	…	x	=↓	x
y	…	y	=↓	y
+(1,y)	…	1+y	=↓	y+1
~~+(x,0)~~	…	x+0	=↓	x
~~+(y,1)~~	…	y+1	=↓	y+1

Enumerative Syntax-Guided Synthesis in SMT

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

...

Maintain a database of candidates whose simplified analogs are pairwise unique

I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

Datatype solver

SAT Solver

Enumerative Syntax-Guided Synthesis in SMT

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

…

SAT Solver

$is_{+}(d)$
$is_{x}(d.1)$
$is_{0}(d.2)$

Datatype solver

d^{M}=**+(x,0)**

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

...

I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

SAT Solver

Datatype solver

d^M=+(x,0)

is₊(d)
is_x(d.1)
is_0(d.2)

compute simplified analog

+(x,0)	x+0	x

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

...

I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

SAT Solver

Datatype solver

$is_{+}(d)$
$is_{x}(d.1)$
$is_{0}(d.2)$

d^{M}=+(x,0)

...not unique

+(x,0)	x+0	x

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

…

$\neg is_{+}(d) \vee \neg is_{x}(d.1) \vee \neg is_{0}(d.2)$

- Return "symmetry breaking" clause

  For details, see **[Reynolds et al FMSD2017]**

I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

SAT Solver

Datatype solver

d^{M}=**+(x,0)**

$is_{+}(d)$
$is_{x}(d.1)$
$is_{0}(d.2)$

+(x,0)	x+0	x

SyGuS in SMT : Symmetry Breaking Clauses

$\neg \mathtt{is}_{+}(\mathtt{d}) \vee \neg \mathtt{is}_{x}(\mathtt{d.1}) \vee \neg \mathtt{is}_{0}(\mathtt{d.2})$

"Do not consider solutions where `d` is **`+(x,0)`**"

SyGuS in SMT : Symmetry Breaking Clauses

$\neg is_{+}(d) \vee \neg is_{x}(d.1) \vee \neg is_{0}(d.2)$

"Do not consider solutions where d is **+(x,0)**"

- Can be applied for any subterm of d:

$\neg is_{+}(d.1) \vee \neg is_{x}(d.1.1) \vee \neg is_{0}(d.1.2)$

"Do not consider solutions where the first child of d is **+(x,0)**"

SyGuS in SMT : Symmetry Breaking Clauses

$\neg\mathtt{is}_{+}(\mathtt{d}) \vee \neg\mathtt{is}_{\mathtt{x}}(\mathtt{d.1}) \vee \neg\mathtt{is}_{0}(\mathtt{d.2})$

"Do not consider solutions where d is **+(x,0)**"

- Can be applied for any subterm of d:

$\neg\mathtt{is}_{+}(\mathtt{d.1}) \vee \neg\mathtt{is}_{\mathtt{x}}(\mathtt{d.1.1}) \vee \neg\mathtt{is}_{0}(\mathtt{d.1.2})$

"Do not consider solutions where the first child of d is **+(x,0)**"

- Can be generalized:

$\neg\mathtt{is}_{+}(\mathtt{d}) \vee \neg\mathtt{is}_{0}(\mathtt{d.2})$

"Do not consider solutions where d is of the form +(t,0) for any t"

$\Rightarrow$ Leads to stronger search space pruning

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

I	Int	Int↓
		x
		y
		y+1

Optimization for I/O examples:

Only consider terms d^M *whose **analog** is unique up to evaluation on input examples*

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

I	Int	Int↓	Ex. Output
x	x	x	
y	y	y	
+(1,y)	1+y	y+1	

SAT Solver

Datatype solver

Maintain output for examples

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

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

I	Int	Int↓	Ex. Output
x	x	x	
y	y	y	
+(1,y)	1+y	y+1	

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

∃f.∀xy.((x=**1**∧y=**1**)⇒f(x,y)=0)∧
((x=2∧y=1)⇒f(x,y)=1)∧
((x=3∧y=1)⇒f(x,y)=2)

I	Int	Int↓	Ex. Output
x	x	x	1
y	y	y	1
+(1,y)	1+y	y+1	2

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

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

I	Int	Int↓	Ex. Output
x	x	x	1,**2**
y	y	y	1,**1**
+(1,y)	1+y	y+1	2,**2**

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

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

I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

...

SAT Solver

$is_+(d)$
$is_1(d.1)$
$is_1(d.2)$

Datatype solver

d^M=+(1,1)

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

...

I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

SAT Solver

Datatype solver

d^M=+(1,1)

```
is+(d)
is1(d.1)
is1(d.2)
```

compute simplified analog

+(1,1)	1+1	2	

Enumerative SyGuS in SMT for I/O Examples

I := **x** | **y** | **0** | **1** | **+**(I,I) | **ite**(B,I,I)
B := **≥**(I,I) | **=**(I,I) | **¬**(B)

…

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

I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

SAT Solver

Datatype solver

d^M=**+(1,1)**

$is_{+}(d)$
$is_{1}(d.1)$
$is_{1}(d.2)$

…and example outputs

+(1,1)	1+1	**2**	**2,2,2**

Enumerative SyGuS in SMT for I/O Examples

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ≥(I,I) | =(I,I) | ¬(B)
```

…

I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

SAT Solver

Datatype solver

d^M=**+(1,1)**

$is_+(d)$
$is_1(d.1)$
$is_1(d.2)$

If not unique…

+(1,1)	1+1	2	2,2,2

Enumerative SyGuS in SMT for I/O Examples

$I := \mathbf{x} \mid \mathbf{y} \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{+}(I,I) \mid \mathbf{ite}(B,I,I)$
$B := \mathbf{\geq}(I,I) \mid \mathbf{=}(I,I) \mid \neg(B)$

…
$\neg is_{+}(d) \vee \neg is_{1}(d.1) \vee \neg is_{1}(d.2)$

- Return symmetry breaking clause
  $\Rightarrow$ *These also can be generalized and are applicable to any subterm of d*

I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

SAT Solver

Datatype solver

d^M=**+(1,1)**

$is_{+}(d)$
$is_{1}(d.1)$
$is_{1}(d.2)$

+(1,1)	1+1	2	2,2,2

Overview

	Input/Output Examples	Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	Enumerative SyGuS + I/O Symmetry Breaking	Enumerative SyGuS CEGQI + reconstruction	Enumerative SyGuS
Without Syntactic Restrictions	CEGQI (trivially)	Counterexample Guided ∀-Instantiation	Enumerative SyGuS (using default restrictions)

What if conjecture is *Partially Single Invocation?*

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

E.g. invariant synthesis problem for $\mathtt{I}$ w.r.t $\mathtt{pre}$, $\mathtt{T}$, $\mathtt{post}$

What if conjecture is *Partially Single Invocation*?

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

Partition into…

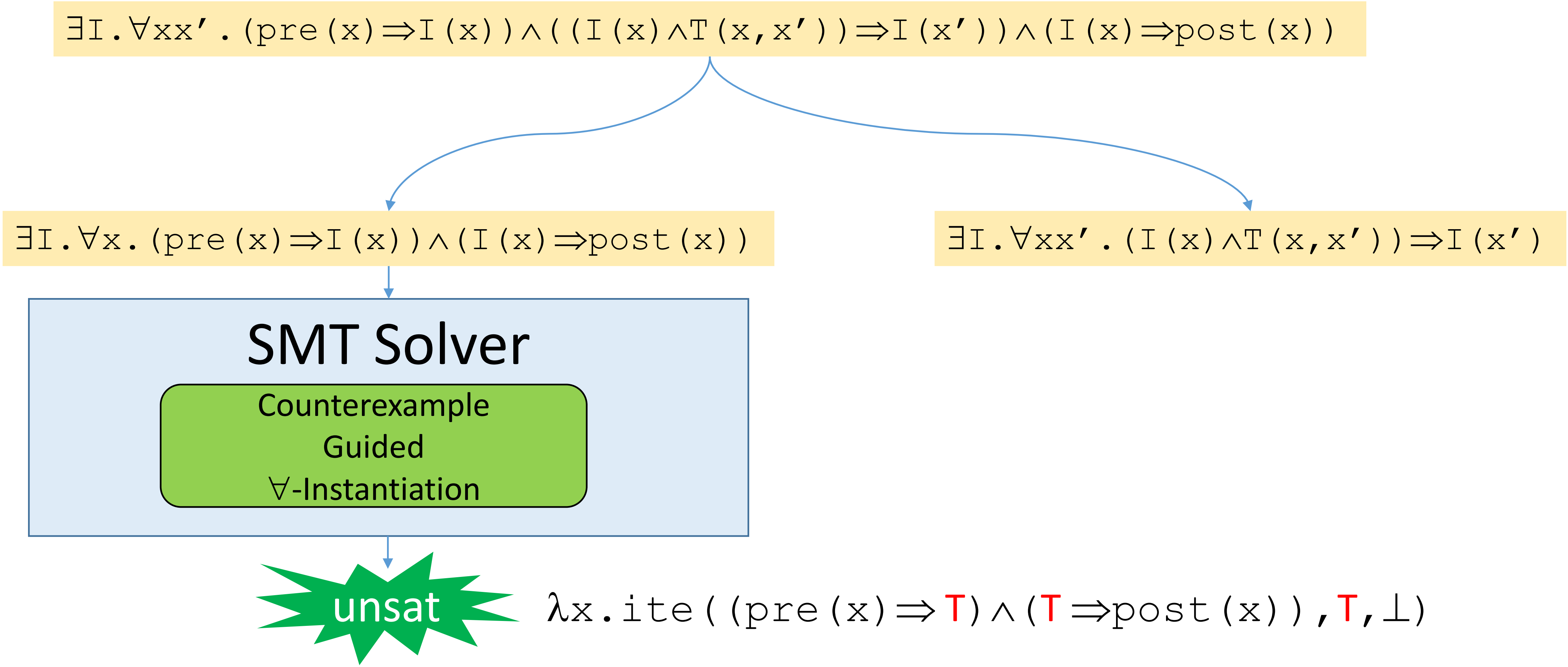
$\exists I. \forall x. (pre(x) \Rightarrow \mathbf{I(x)}) \wedge (\mathbf{I(x)} \Rightarrow post(x))$

Single-invocation portion

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

Non-single-invocation portion

What if conjecture is *Partially Single Invocation*?



What if conjecture is *Partially Single Invocation?*

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

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

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

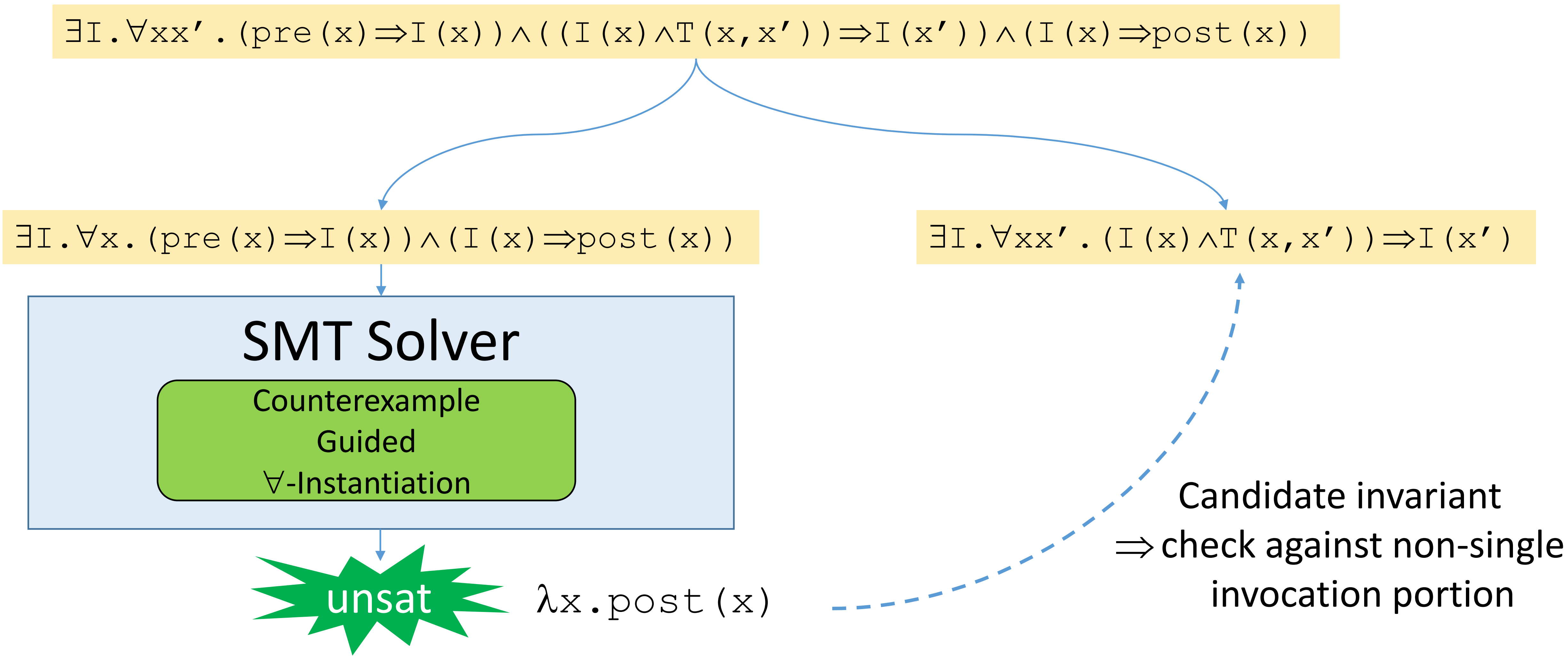
SMT Solver

Counterexample
Guided
∀-Instantiation

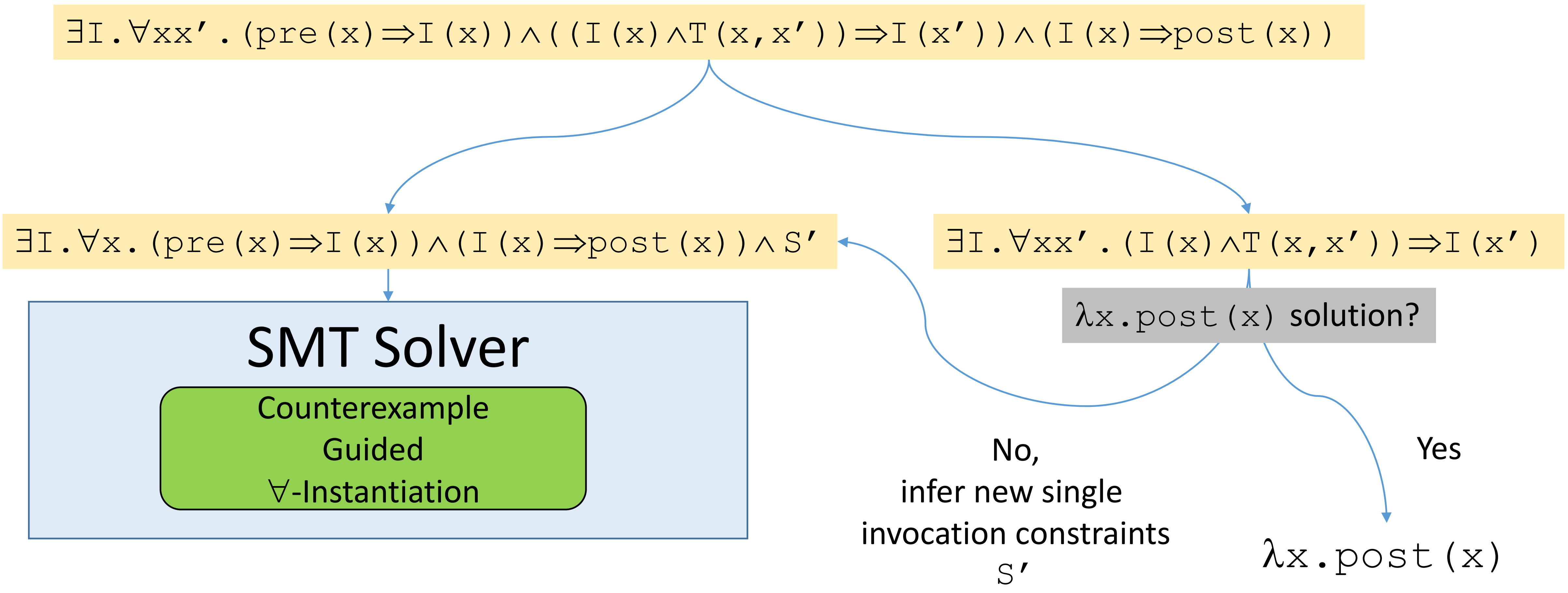
unsat

$\lambda x.post(x)$

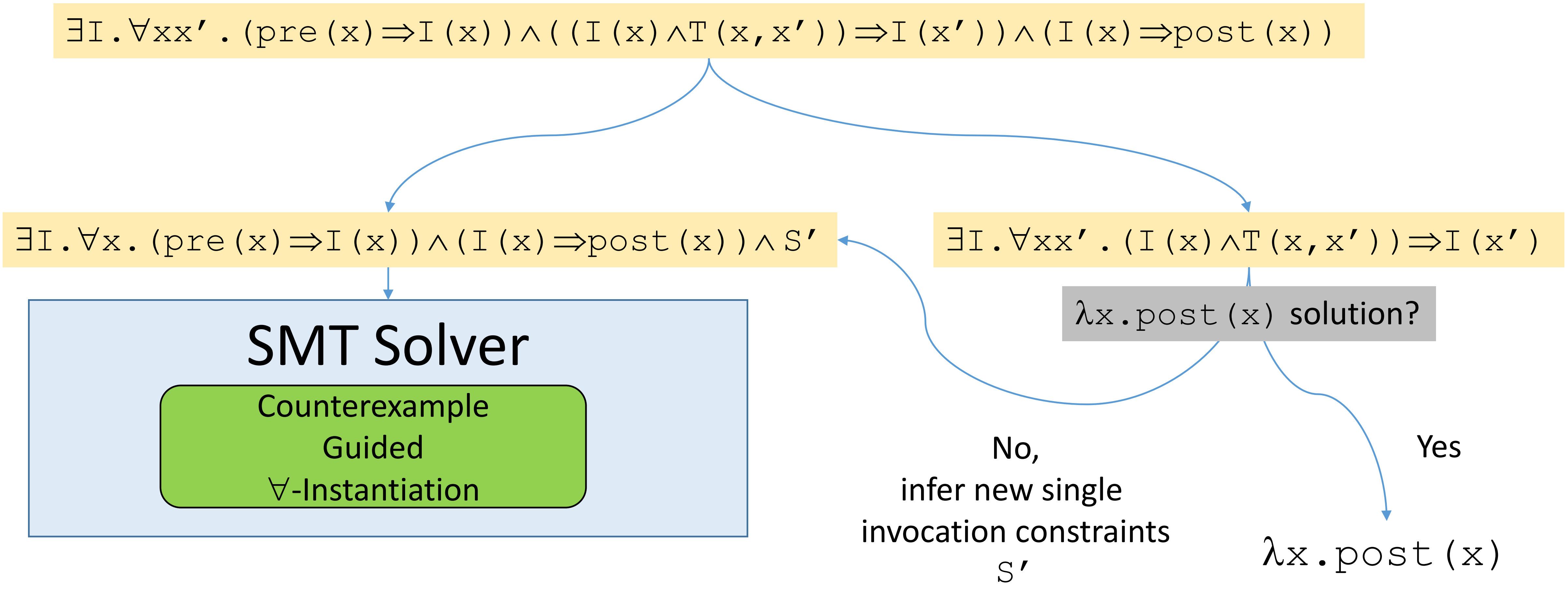
What if conjecture is *Partially Single Invocation?*



What if conjecture is *Partially Single Invocation?*



What if conjecture is *Partially Single Invocation*?

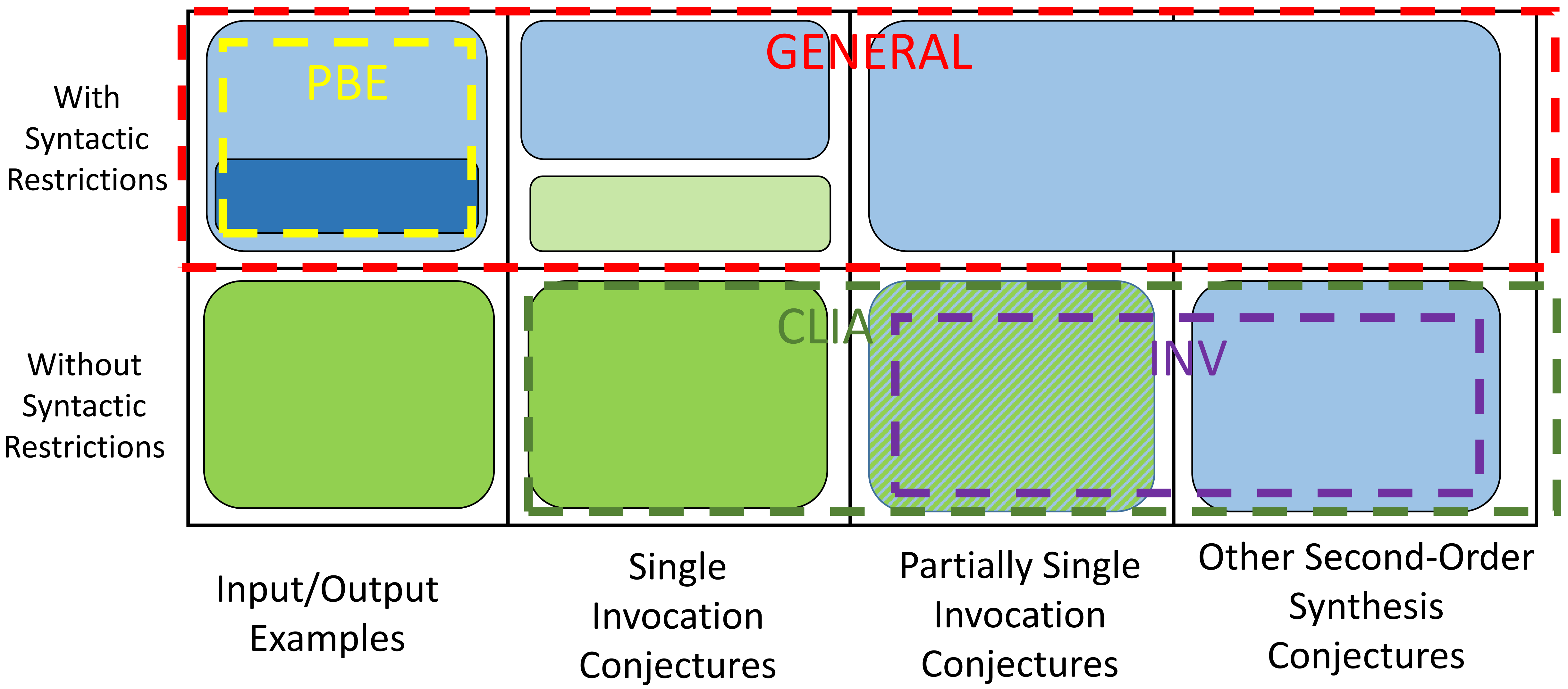


⇒Related to property-directed reachability (PDR) **[Bradley 2011]**

Overview

	Input/Output Examples	Single Invocation Conjectures	Partially Single Invocation Conjectures	Other Second-Order Synthesis Conjectures
With Syntactic Restrictions	Enumerative SyGuS + I/O Symmetry Breaking	Enumerative SyGuS CEGQI + reconstruction	Enumerative SyGuS	Enumerative SyGuS
Without Syntactic Restrictions	CEGQI (trivially)	Counterexample Guided ∀-Instantiation	Hybrid approaches?	Enumerative SyGuS (using default restrictions)

CVC4 for SyGuS Comp 2017:



Topics Not Covered:

- Automatically inferring when a conjecture ⇔ a single-invocation one
- CEGQI solution minimization by proof analysis
  - Simpler proofs ⇒ shorter functions
- Approaches inspired by synthesis by unification **[Alur et al TACAS2017]**
  - Decision tree learning for `ite`-solutions for PBE Bit-Vectors
  - Sequencing algorithm for `concat`-solutions for PBE Strings
- Cases when CEGQI can benefit from syntax-guided synthesis
  - SMT approaches for ∀+BV may benefit from SyGuS **[Preiner et al TACAS2017]**

- …Thanks for listening!

- SyGuS solver in master branch of CVC4:
  - Open source
  - Available at : http://cvc4.cs.stanford.edu/web/
  - Accepts *.smt2, *.sy formats
  - …many other features