SyGuS Techniques in the Core of an SMT Solver

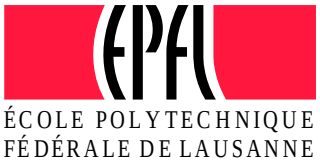
Andrew Reynolds

SYNT Workshop

July 22, 2017

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



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) | Ė(fBool)
```

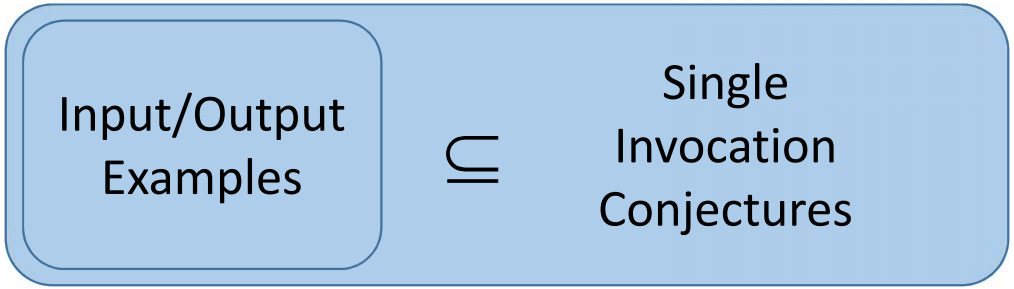
Find solutions `f` = λ`x.t`, where `t` is generated by grammar $\mathcal{R}$

Synthesis Conjectures : Overview

Input/Output Examples

e.g. $\exists f.\forall x.(x=i_1 \Rightarrow f(x)=o_1) \wedge (x=i_2 \Rightarrow f(x)=o_2) \wedge (x=i_3 \Rightarrow f(x)=o_3)$

Synthesis Conjectures : Overview

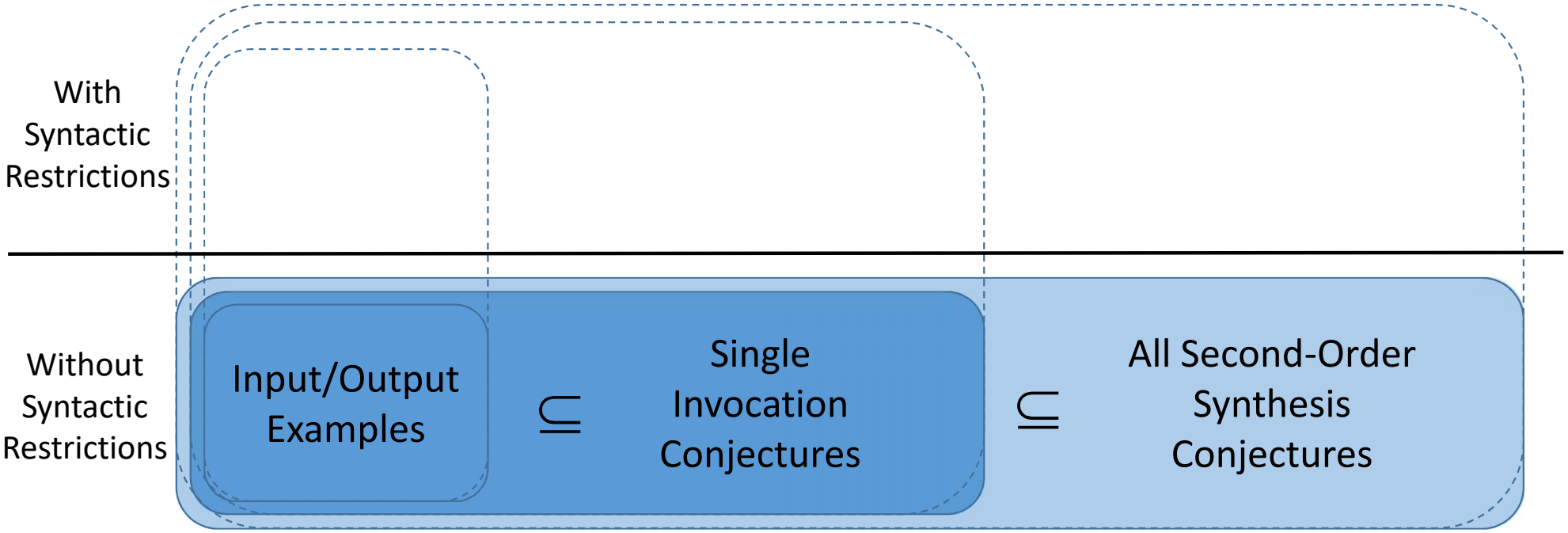


e.g. $\exists f.\forall xy.\ f(x,y) \geq x \wedge f(x,y) \geq y \wedge (f(x,y) = x \vee f(x,y) = y)$

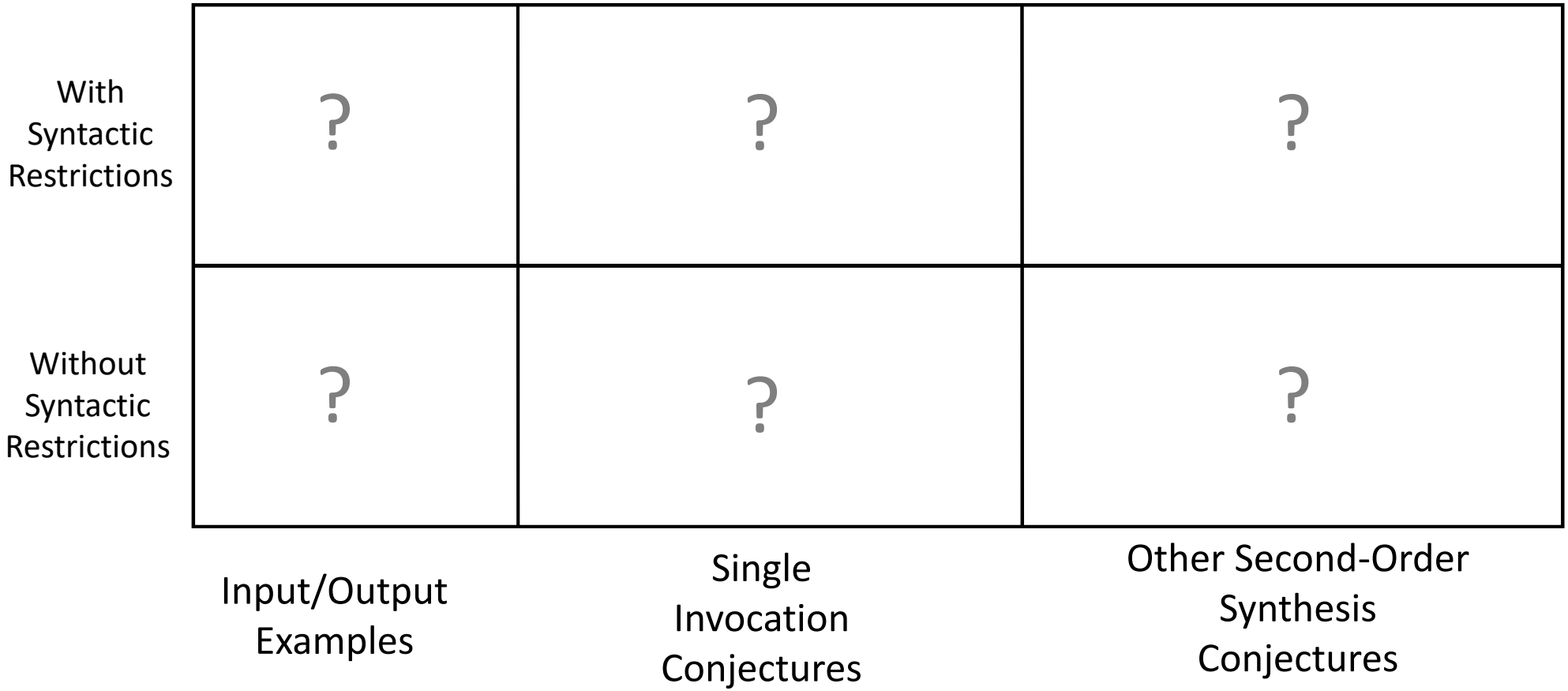
Synthesis Conjectures : Overview

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

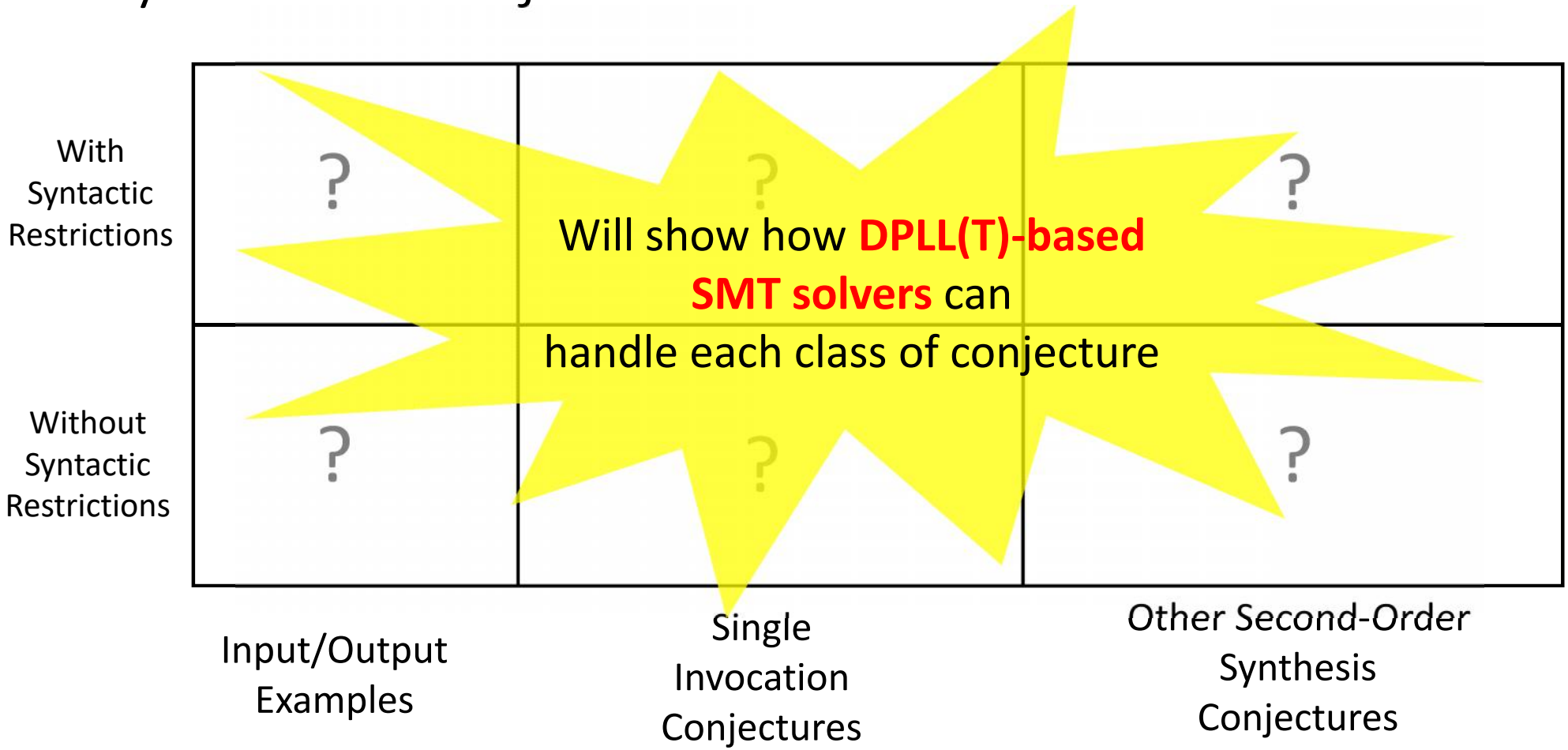
Synthesis Conjectures : Overview



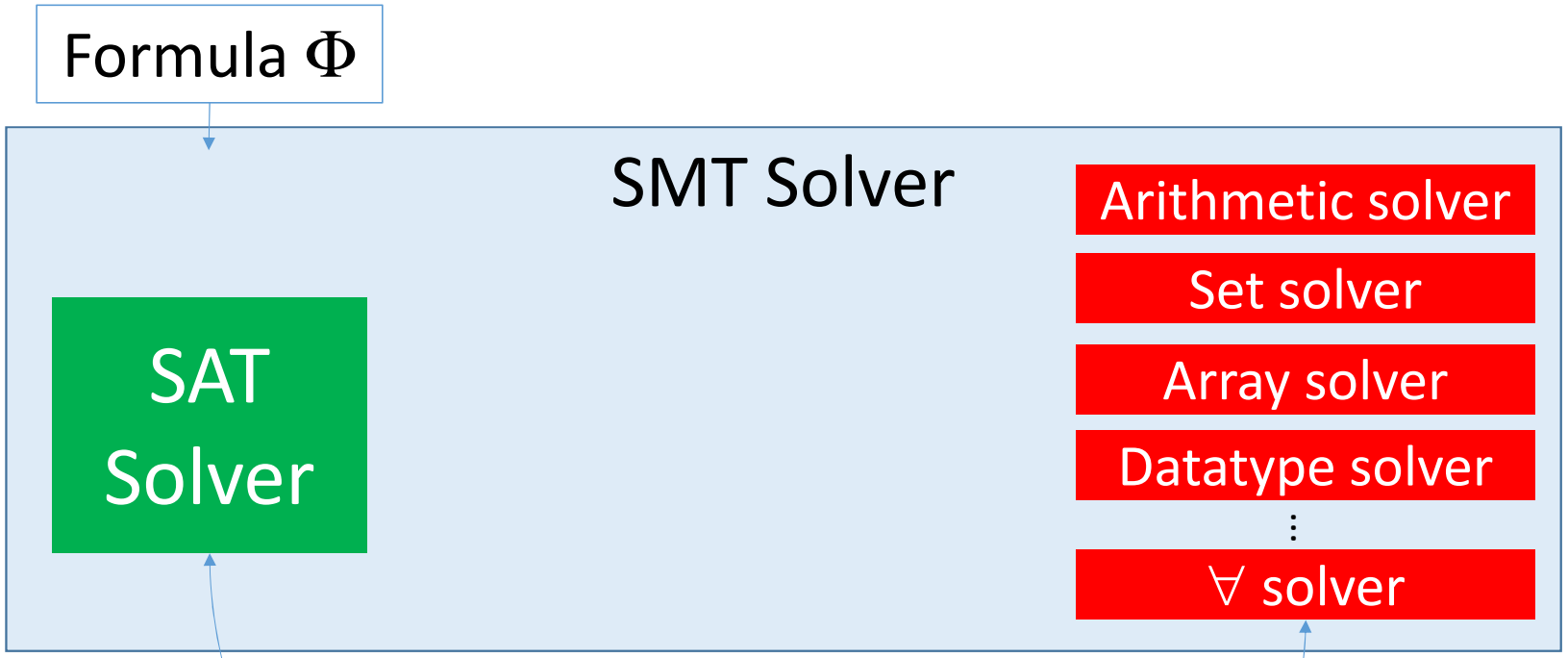
Synthesis Conjectures : Overview



Synthesis Conjectures : Overview



DPLL(T)-based SMT Solvers



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

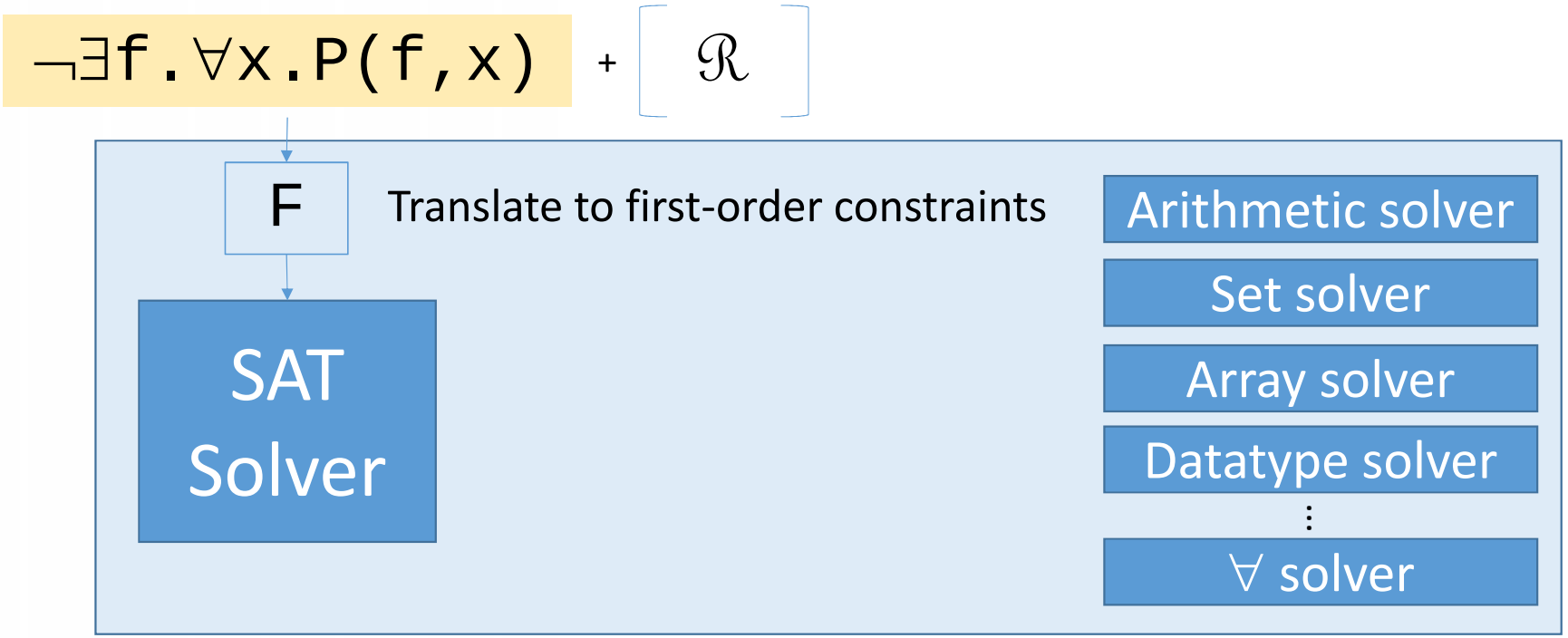
DPLL(T)-based SMT Solvers for Synthesis

DPLL(T)-based SMT Solvers for Synthesis

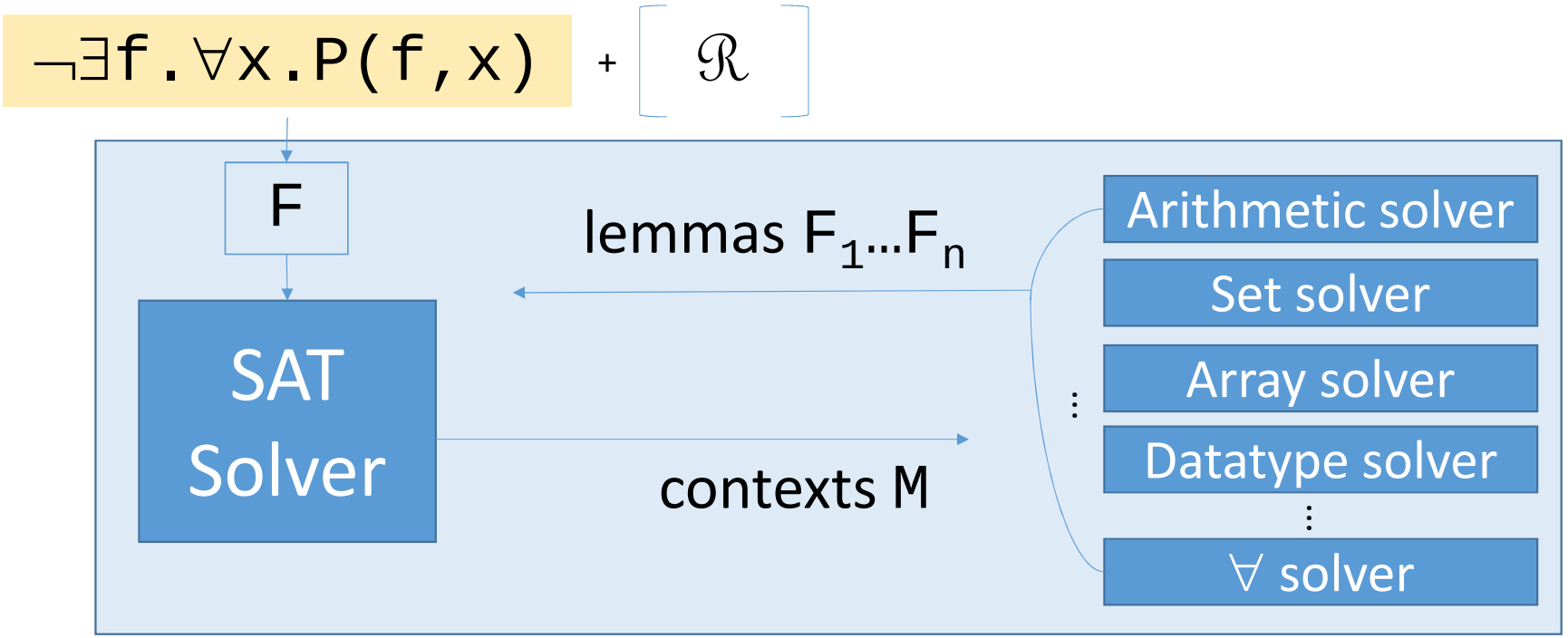
$\dot{E}\exists f.\forall x.P(f,x)$ + $\mathcal{R}$

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

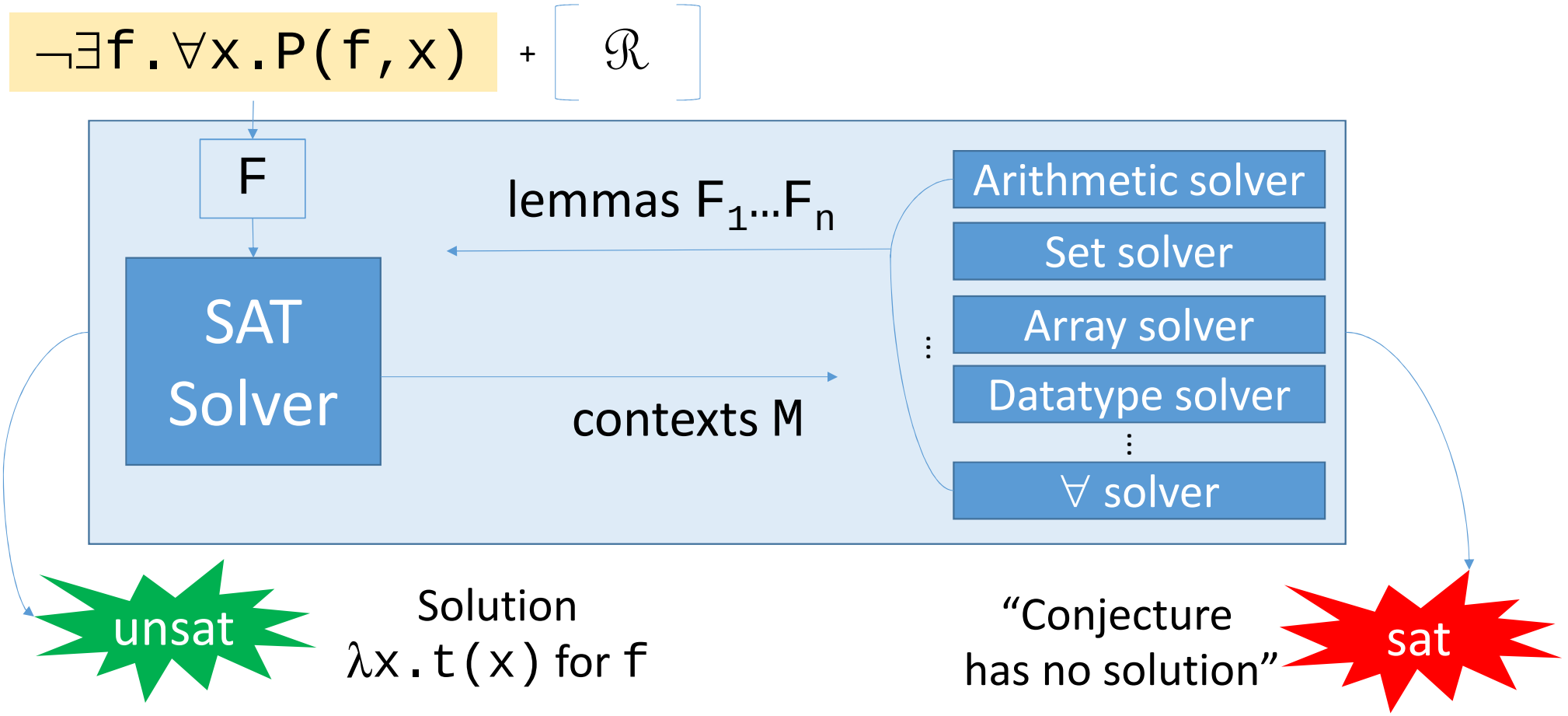
DPLL(T)-based SMT Solvers for Synthesis



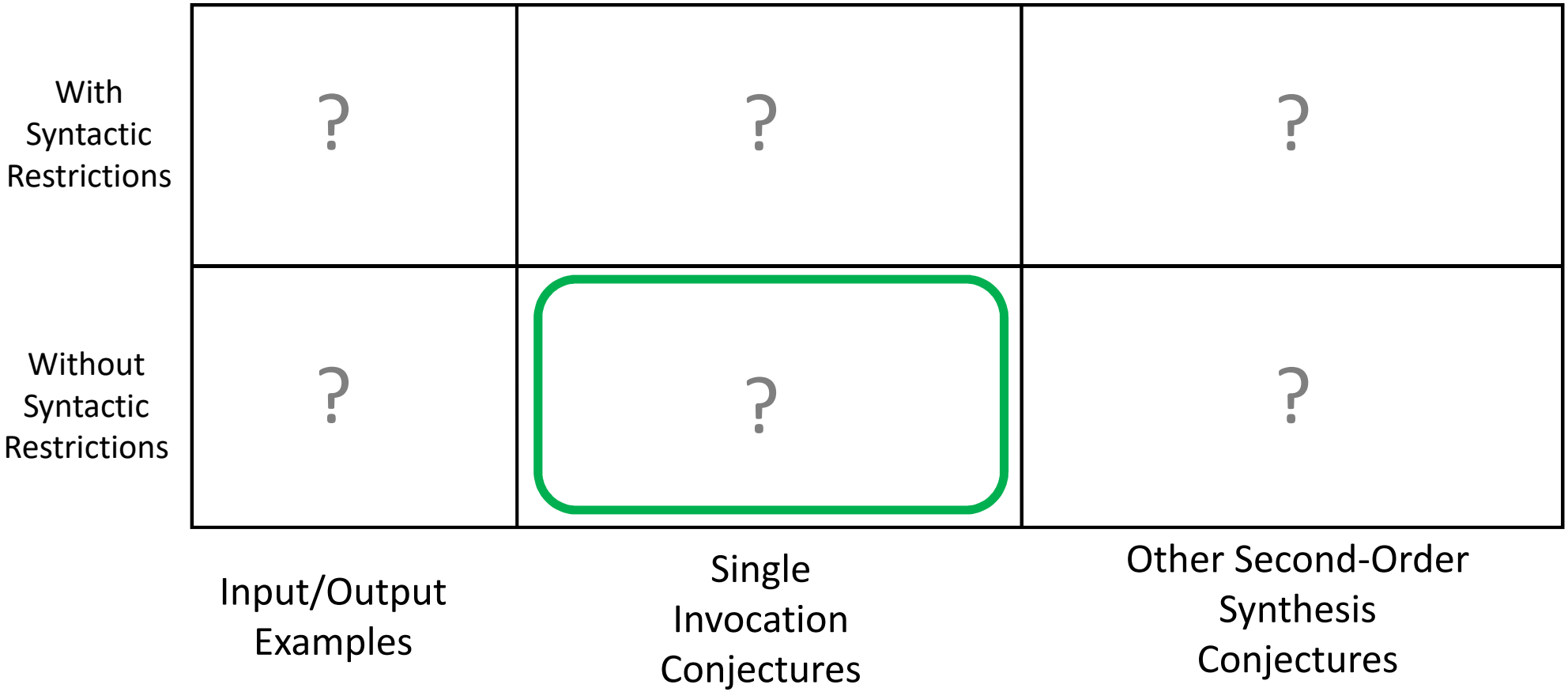
DPLL(T)-based SMT Solvers for Synthesis



DPLL(T)-based SMT Solvers for Synthesis



Single Invocation w/o Syntactic Restrictions



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*:

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

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

Single Invocation w/o Syntactic Restrictions

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

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

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

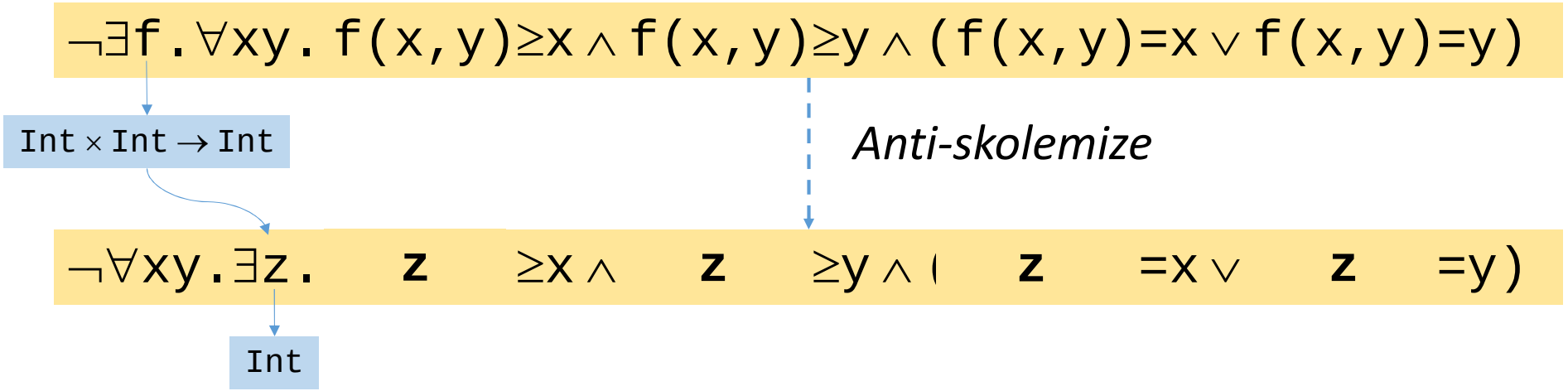
Ĝ “single invocation” synthesis conjectures

Single Invocation w/o Syntactic Restrictions

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

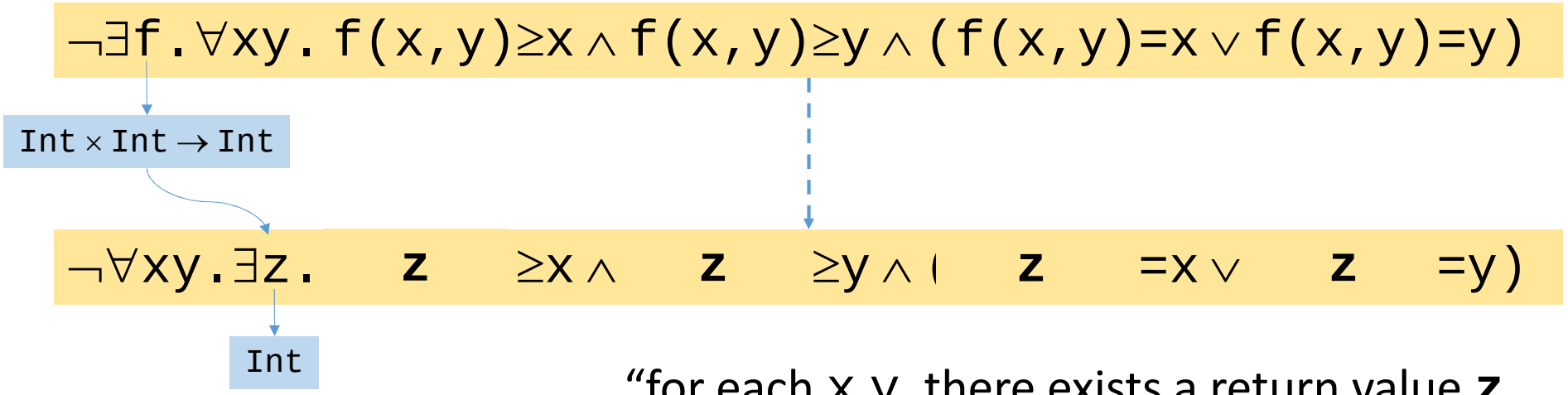
$\texttt{Int} \times \texttt{Int} \to \texttt{Int}$

Single Invocation w/o Syntactic Restrictions



[Reynolds et al CAV2015]

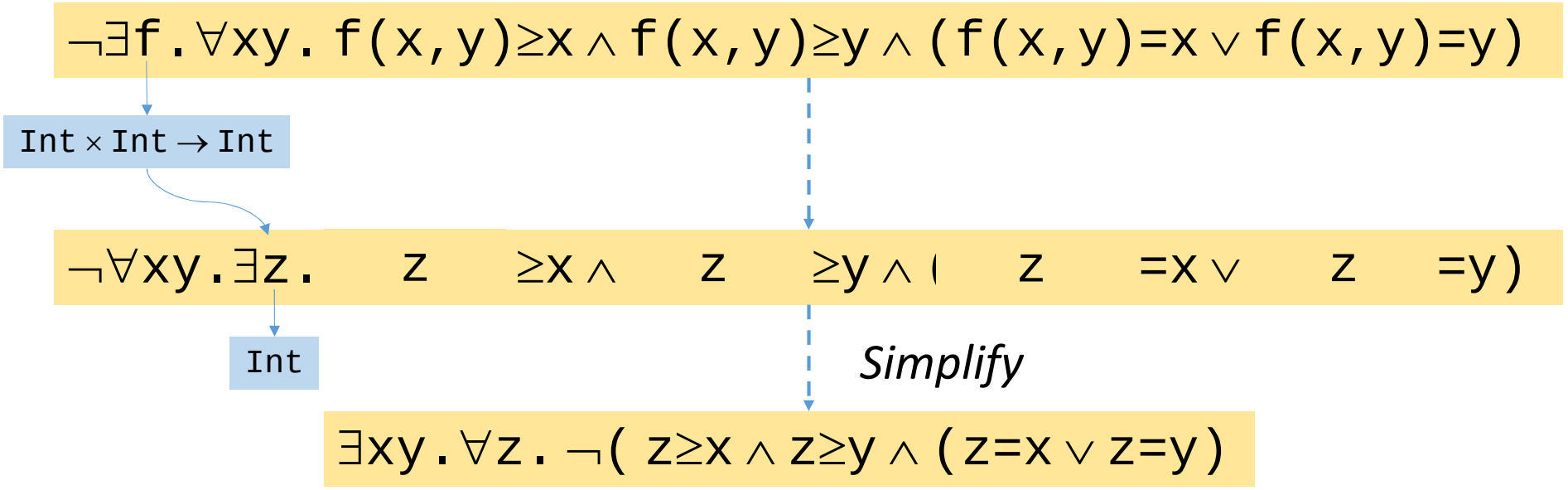
Single Invocation w/o Syntactic Restrictions



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

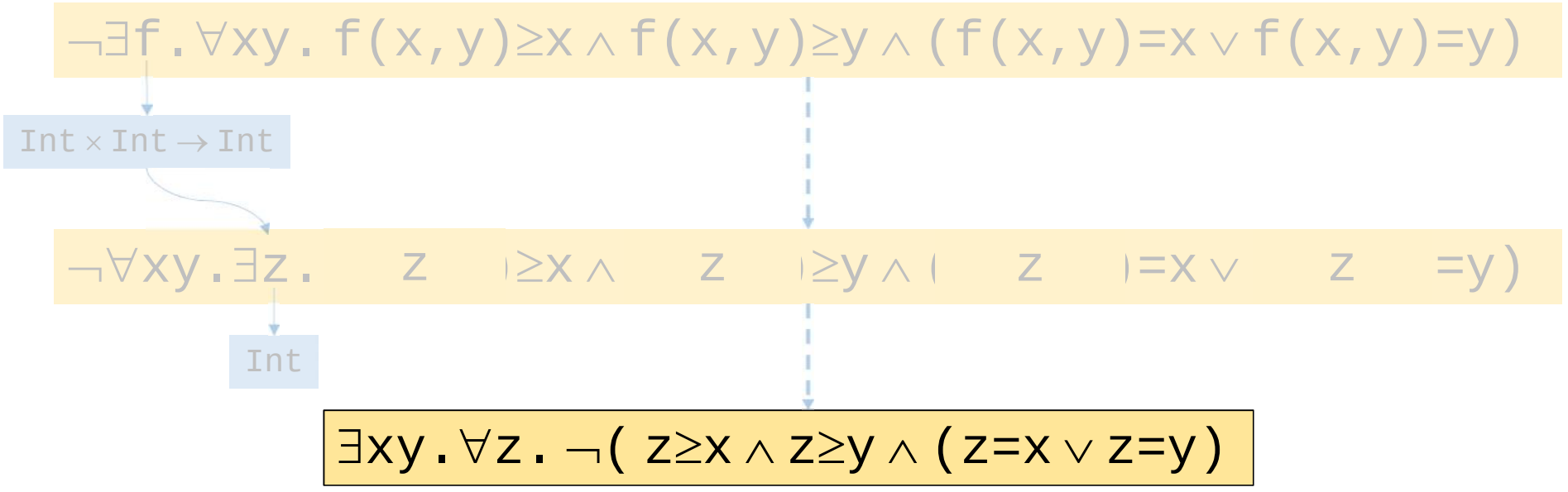
[Reynolds et al CAV2015]

Single Invocation w/o Syntactic Restrictions



[Reynolds et al CAV2015]

Single Invocation w/o Syntactic Restrictions



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

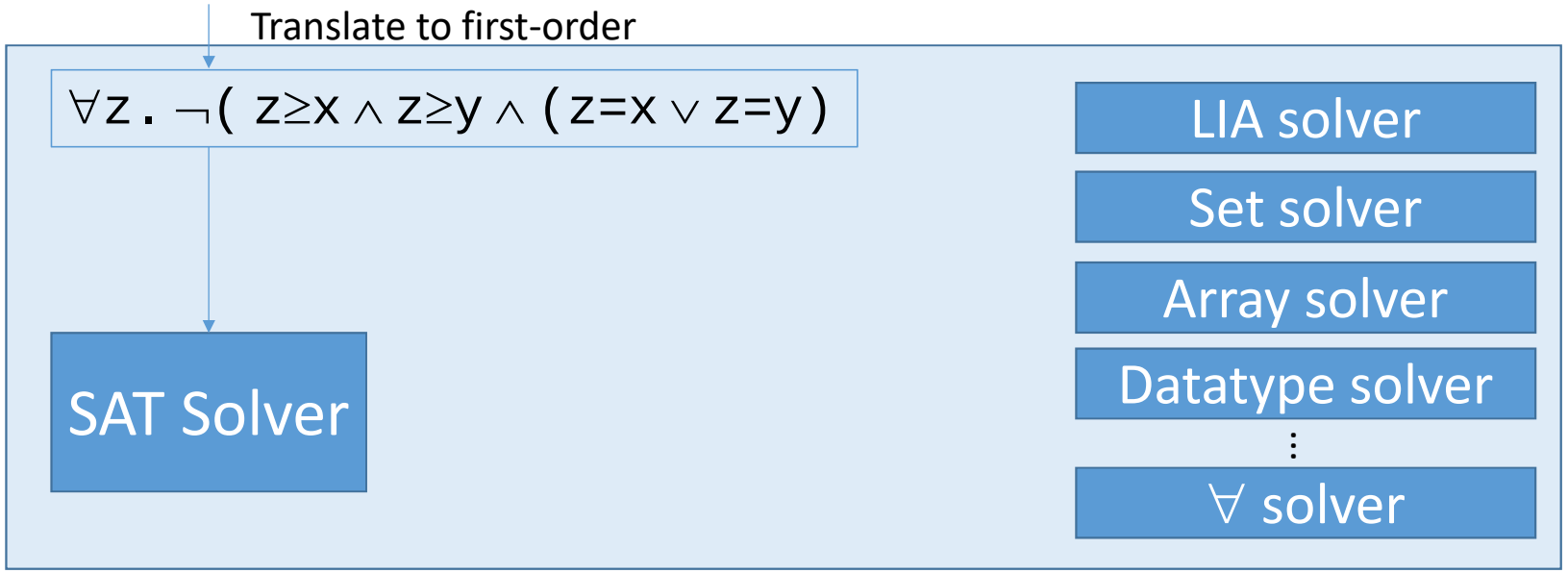
[Reynolds et al CAV2015]

Single Invocation Synthesis in SMT

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

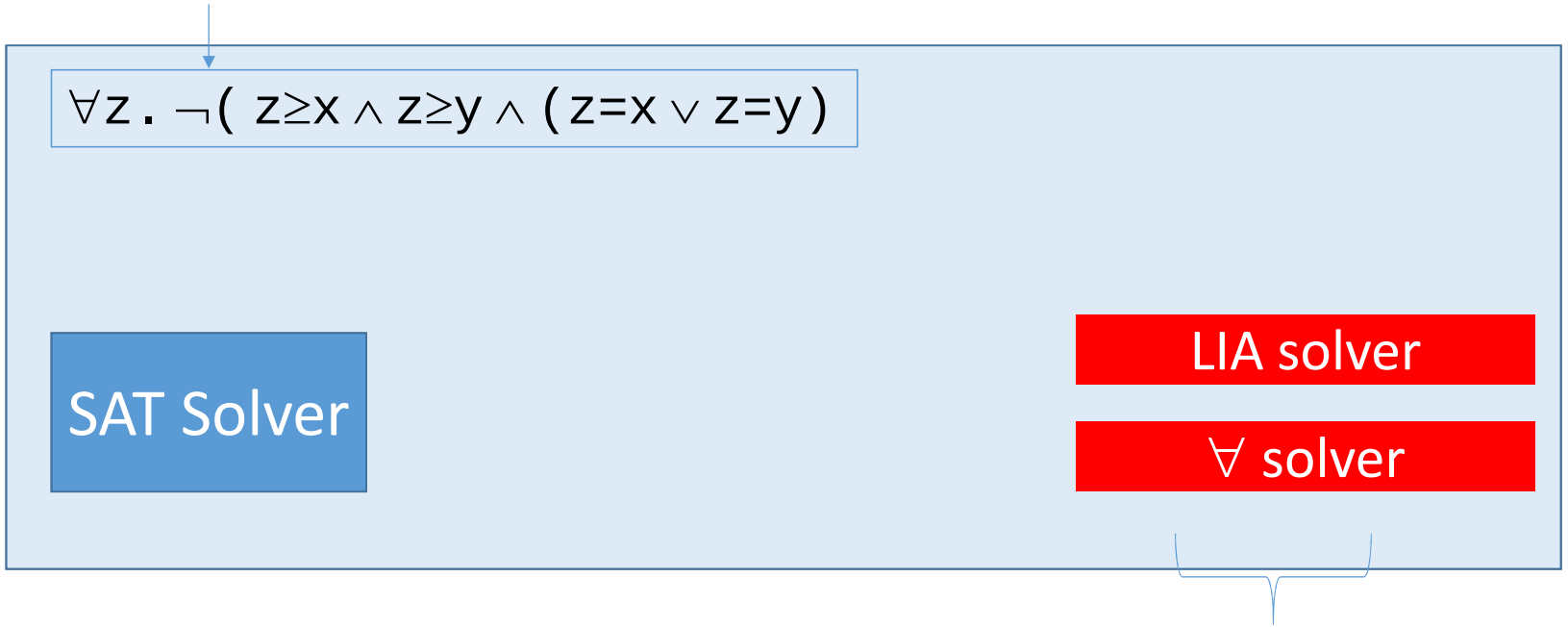
Single Invocation Synthesis in SMT

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



Single Invocation Synthesis in SMT

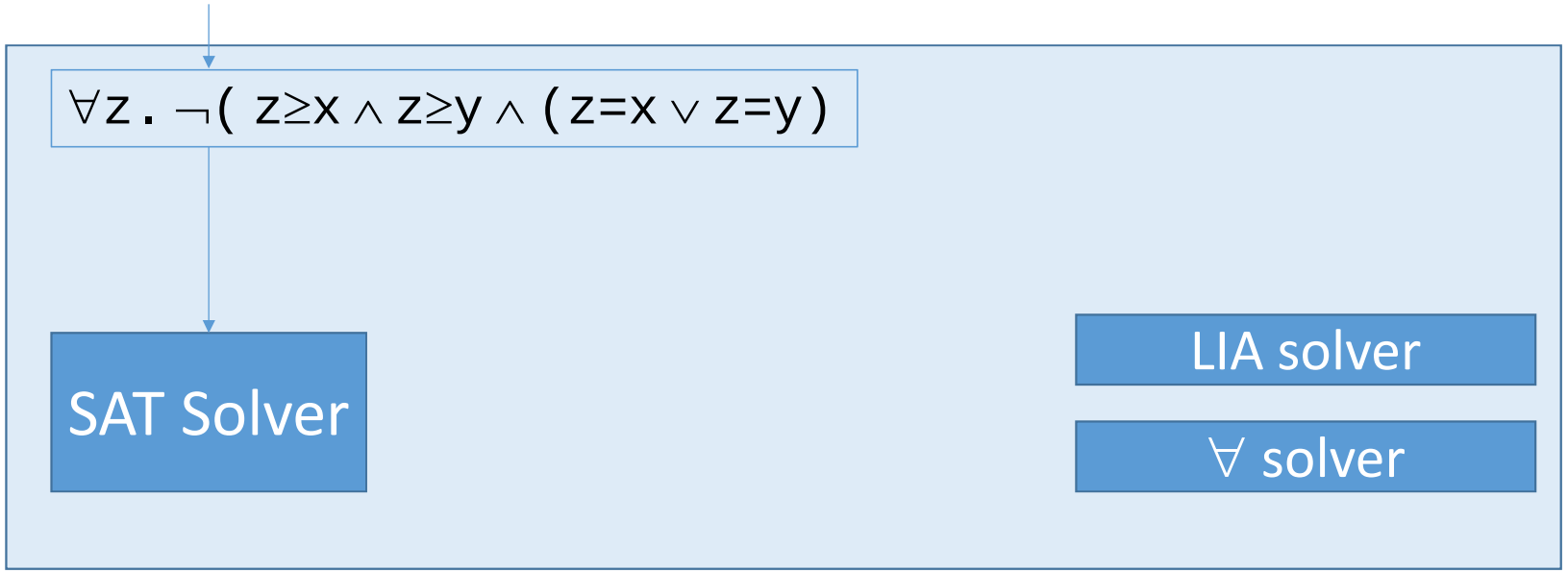
$\neg\exists f.\forall xy.\ f(x,y)\geq x \wedge f(x,y)\geq y \wedge (f(x,y)=x \vee f(x,y)=y)$



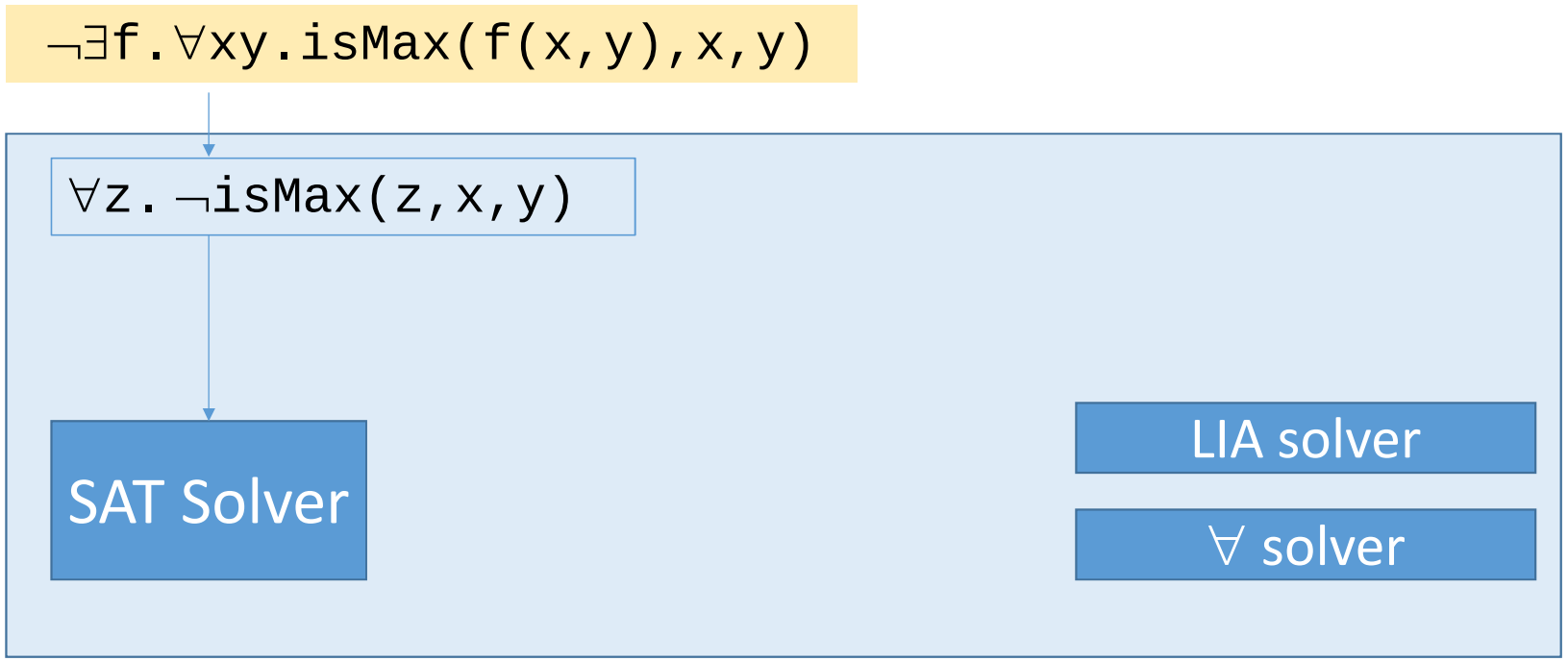
Solve use first-order $\forall$-instantiation for linear arithmetic (LIA)

Single Invocation Synthesis in SMT

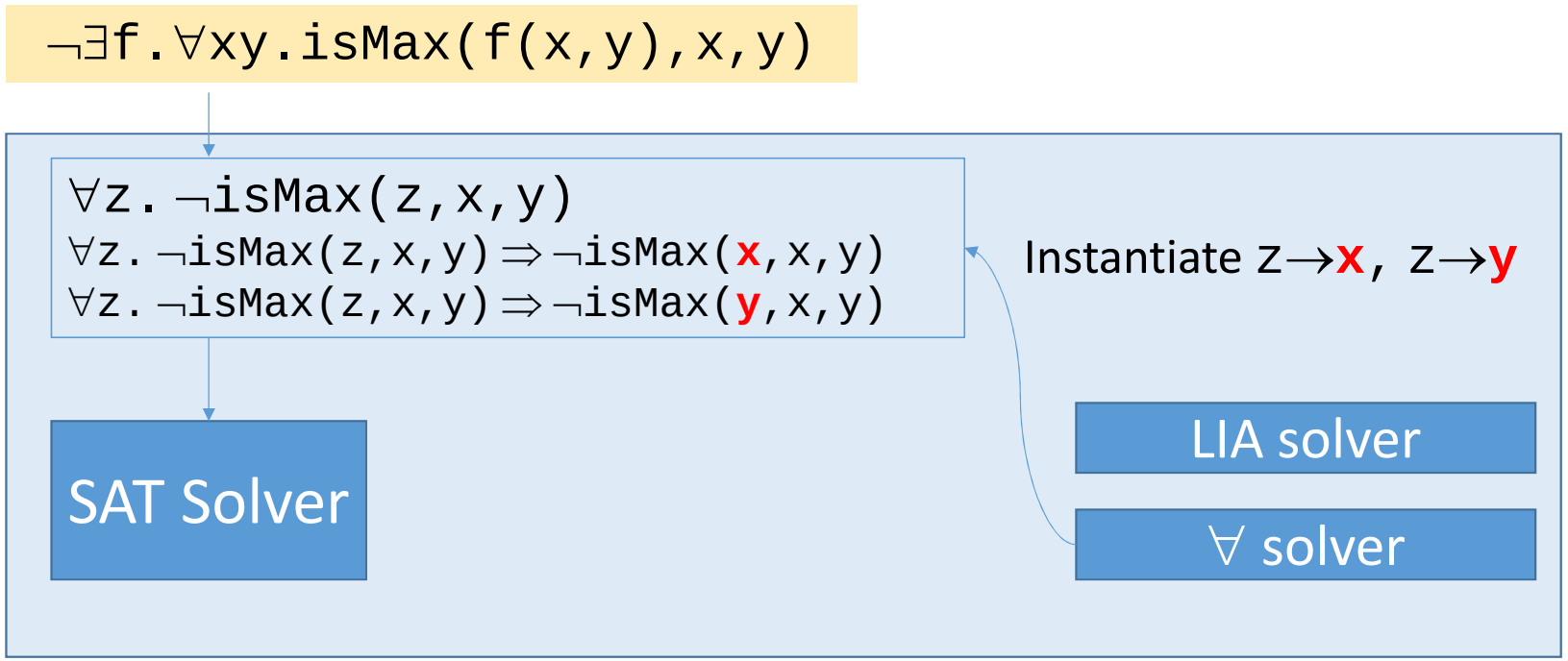
$\neg\exists f.\forall xy.\ f(x,y)\geq x \wedge f(x,y)\geq y \wedge (f(x,y)=x \vee f(x,y)=y)$



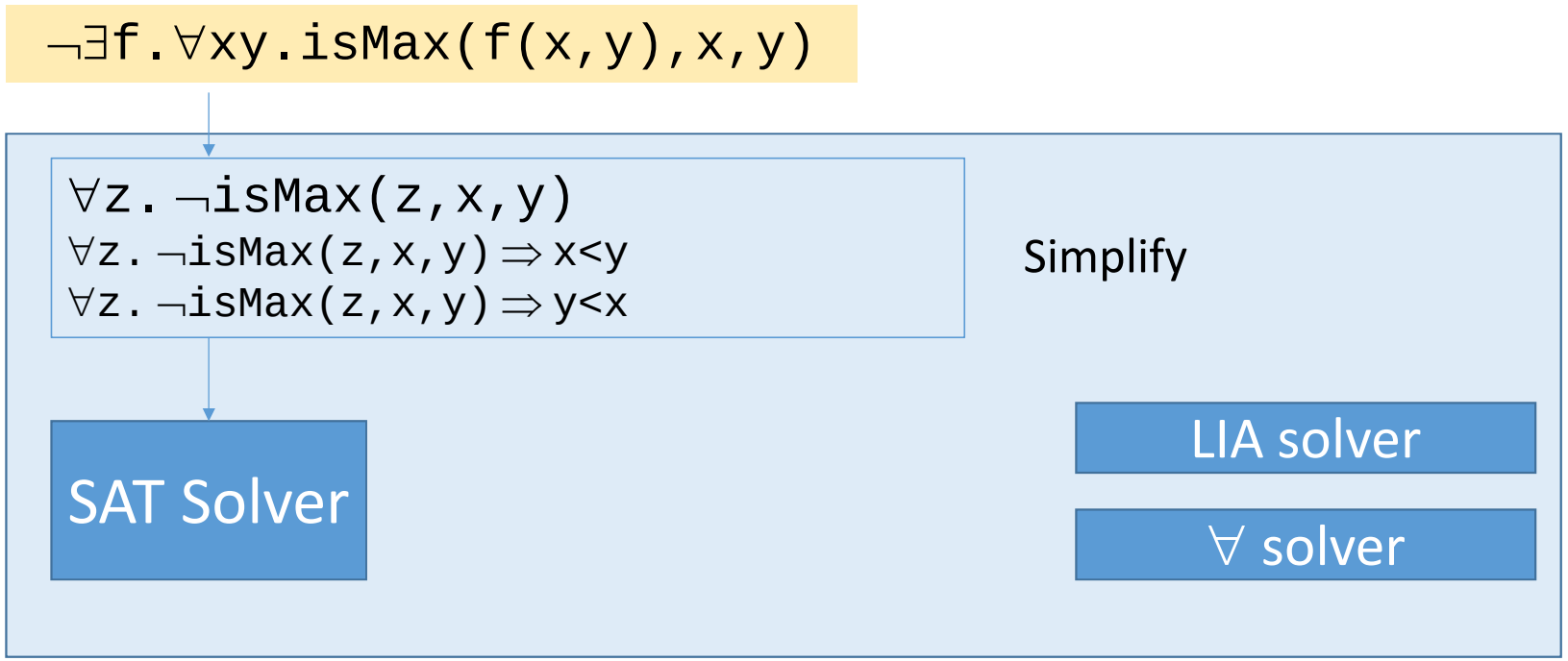
Single Invocation Synthesis in SMT



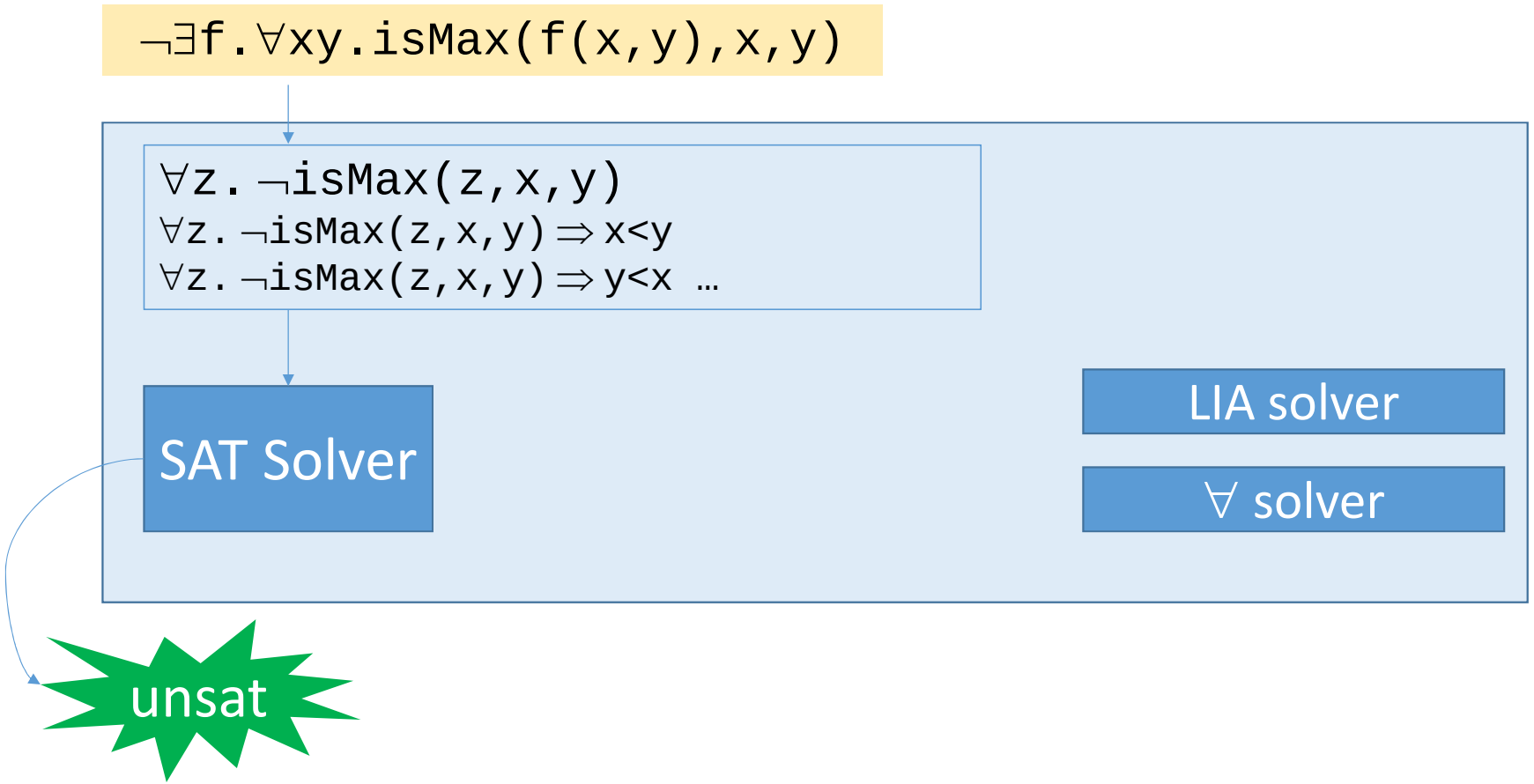
Single Invocation Synthesis in SMT



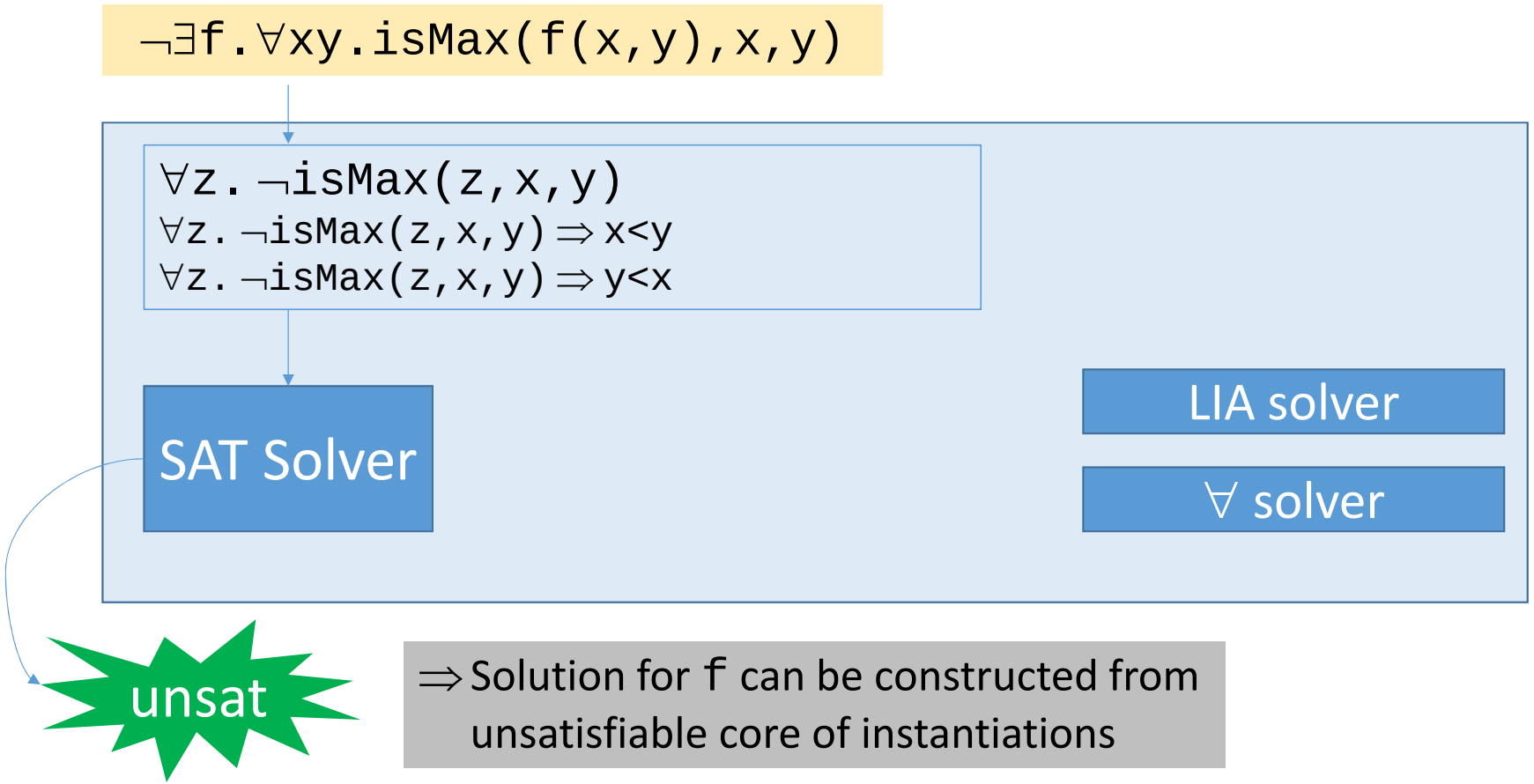
Single Invocation Synthesis in SMT



Single Invocation Synthesis in SMT

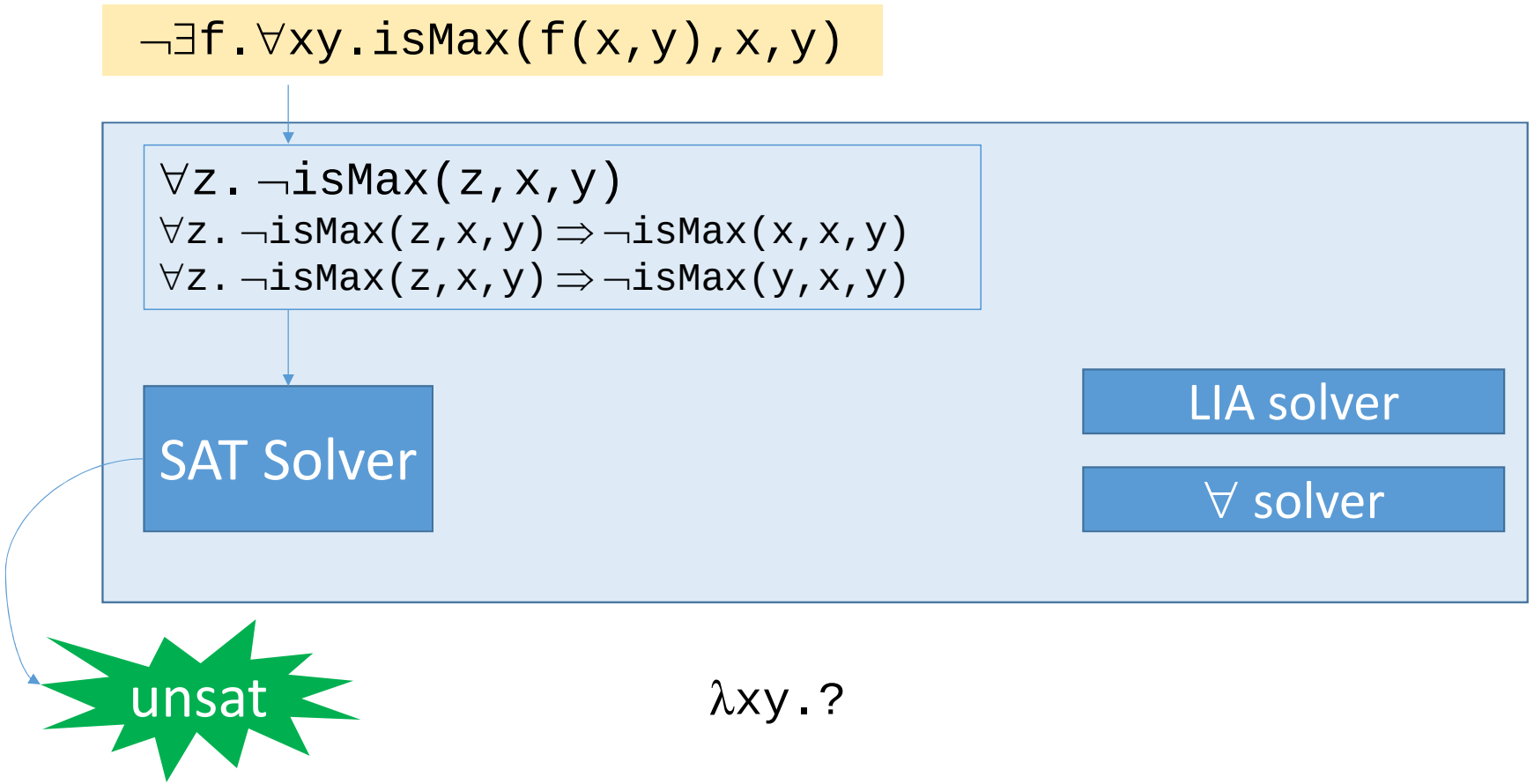


Single Invocation Synthesis in SMT



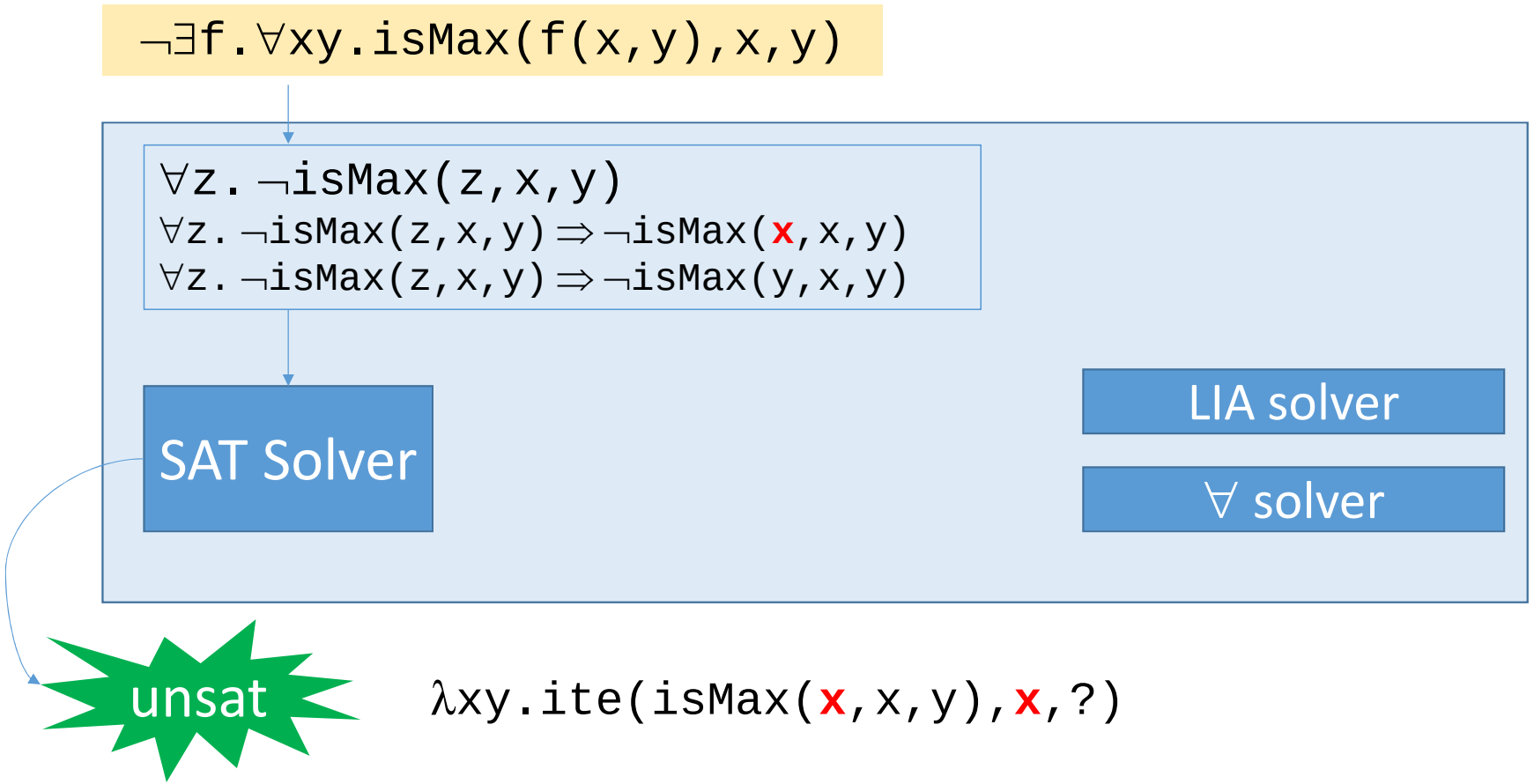
⇒ Solution for f can be constructed from unsatisfiable core of instantiations

Single Invocation Synthesis in SMT

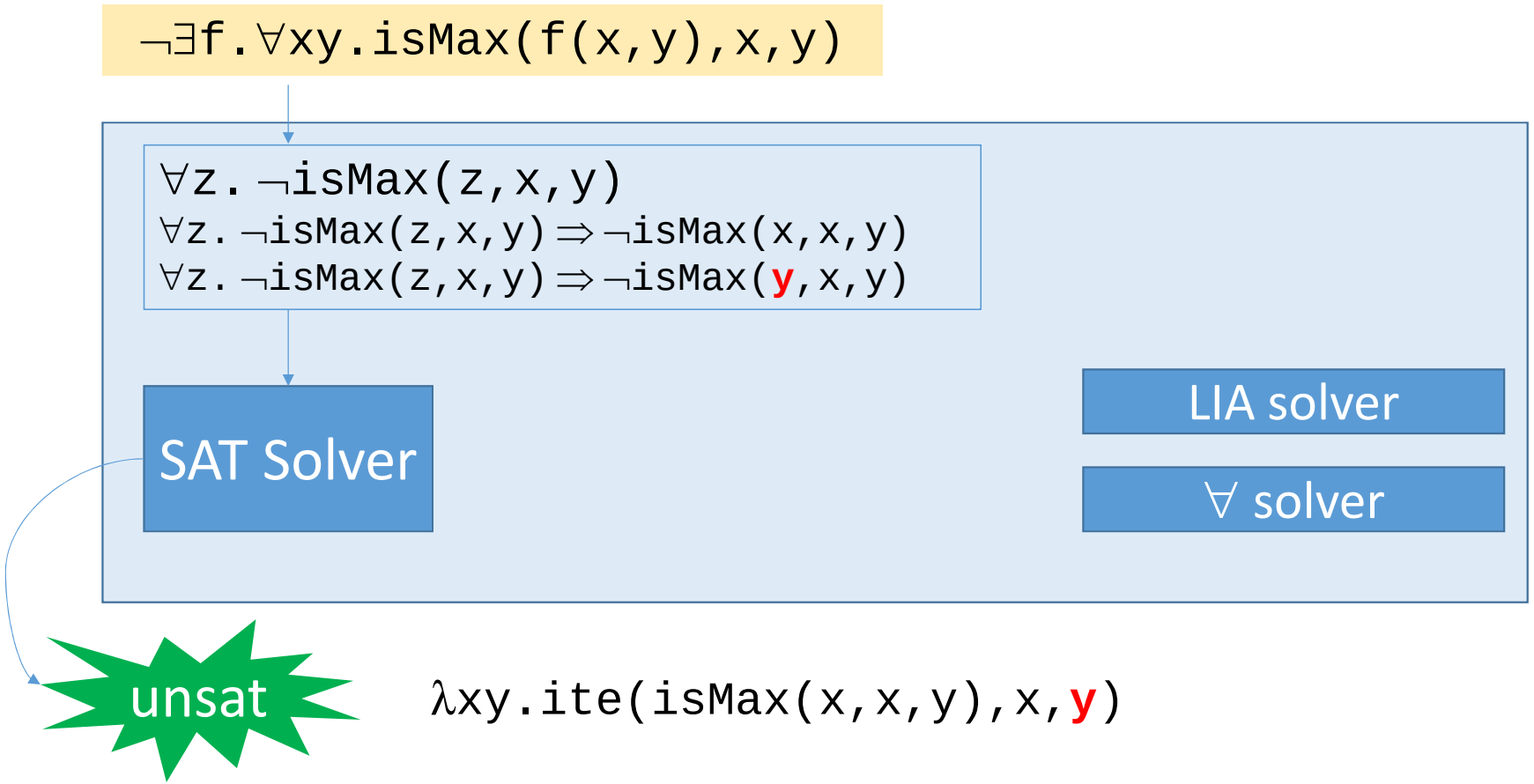


λxy.?

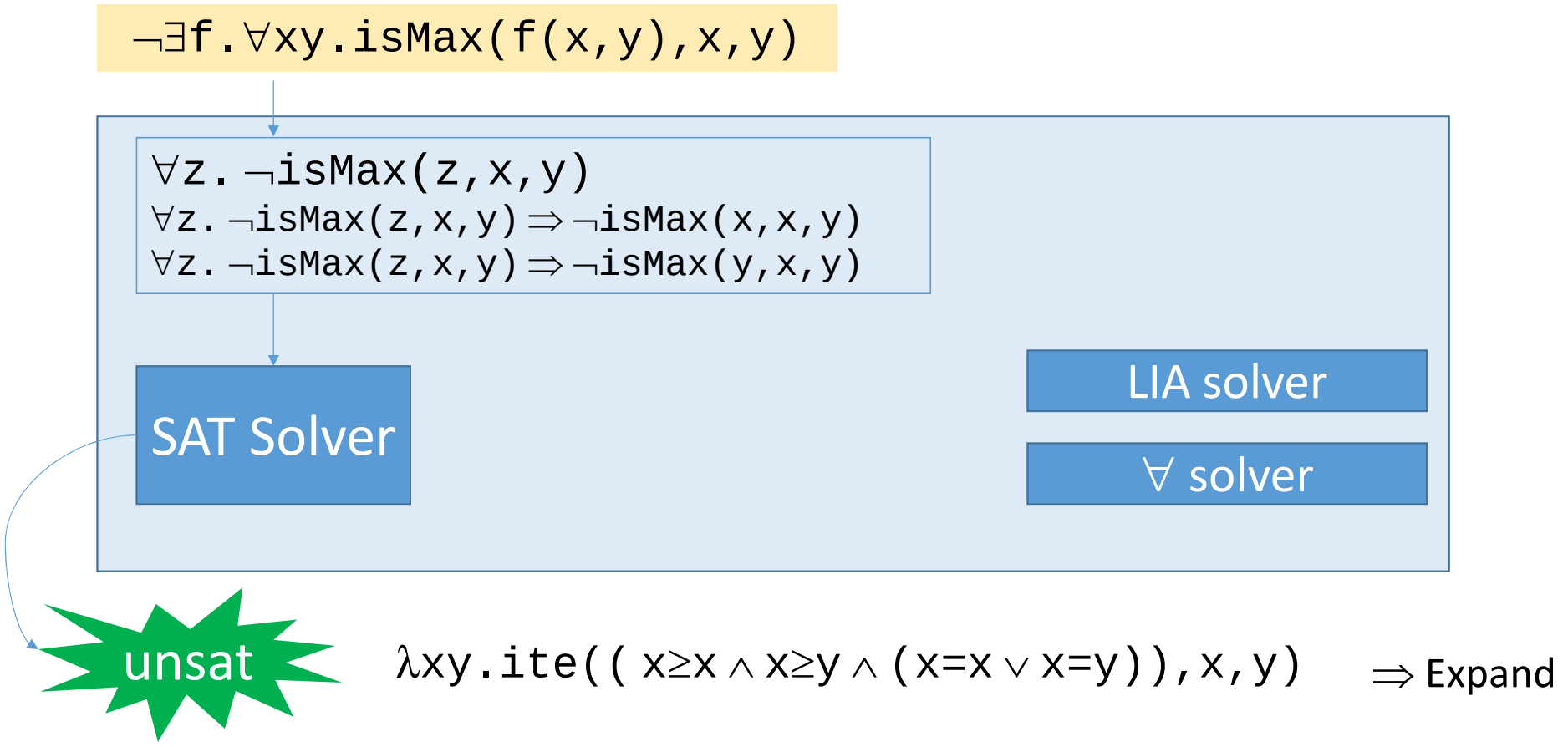
Single Invocation Synthesis in SMT



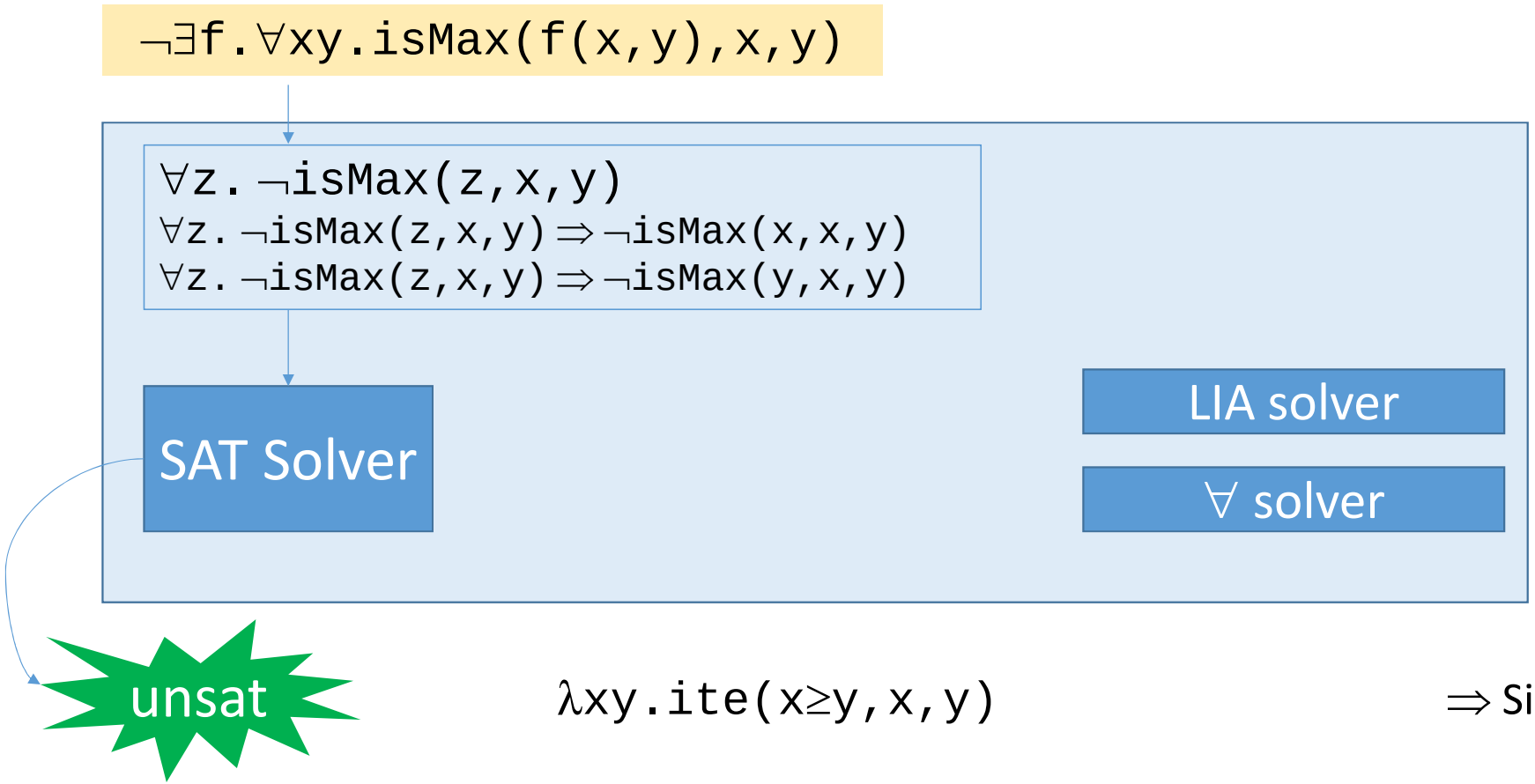
Single Invocation Synthesis in SMT



Single Invocation Synthesis in SMT



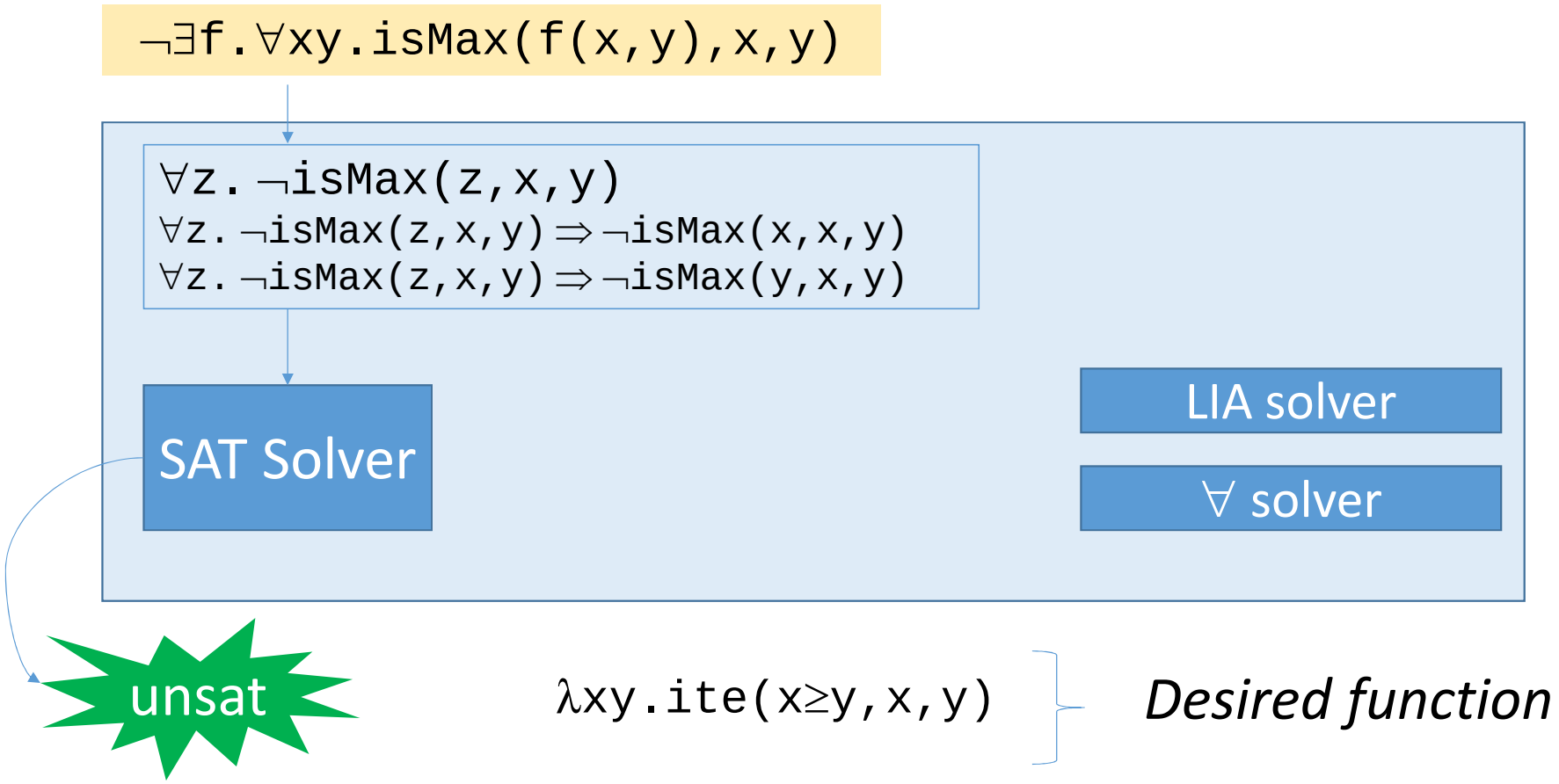
Single Invocation Synthesis in SMT



λxy.ite(x≥y,x,y)

⇒ Simplify

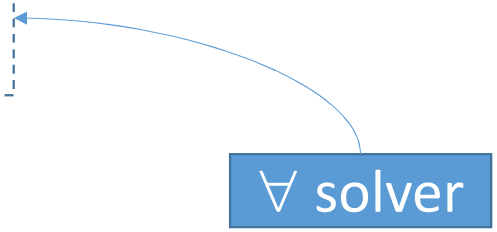
Single Invocation Synthesis in SMT



Single Invocation Synthesis in SMT

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

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

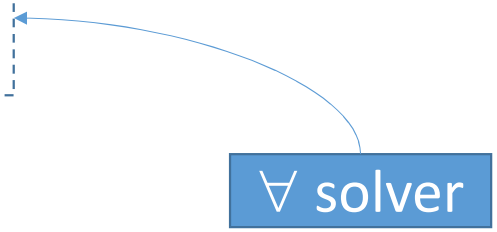


Single Invocation Synthesis in SMT

- 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

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

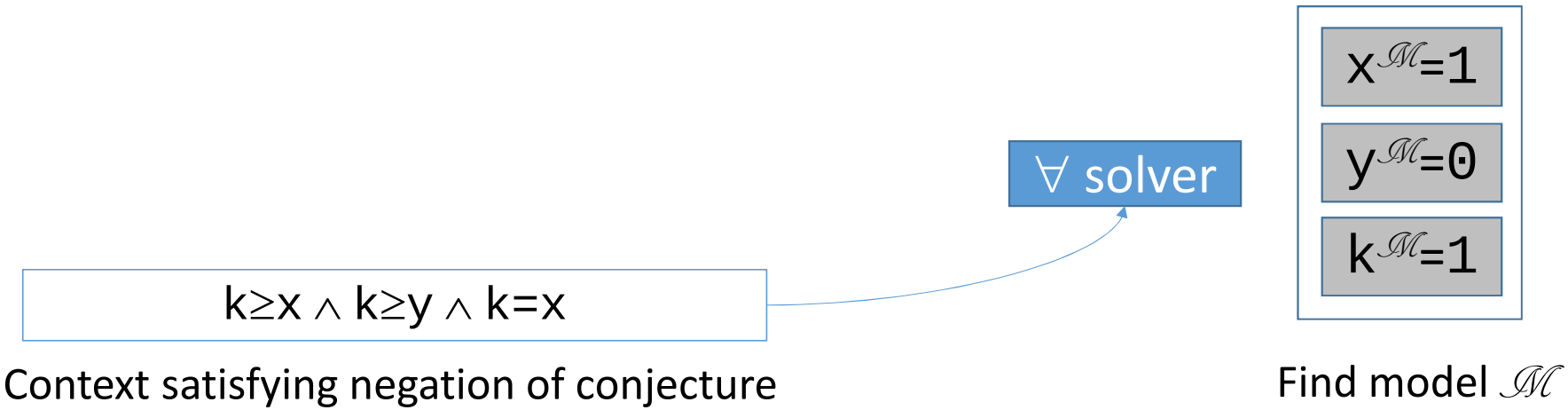
∀ solver

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

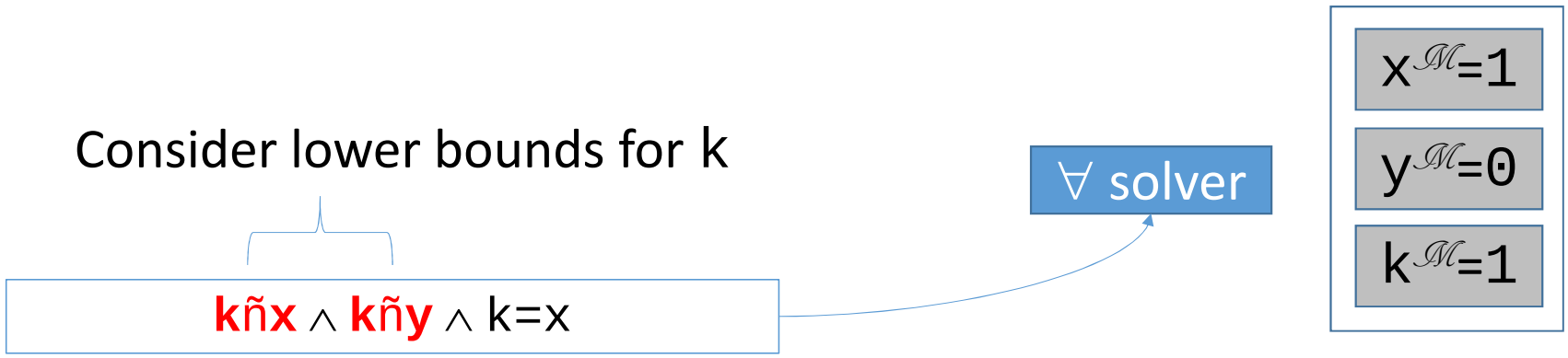
⇒ Consider conjecture's negation

- **k** is counterexample to conjecture

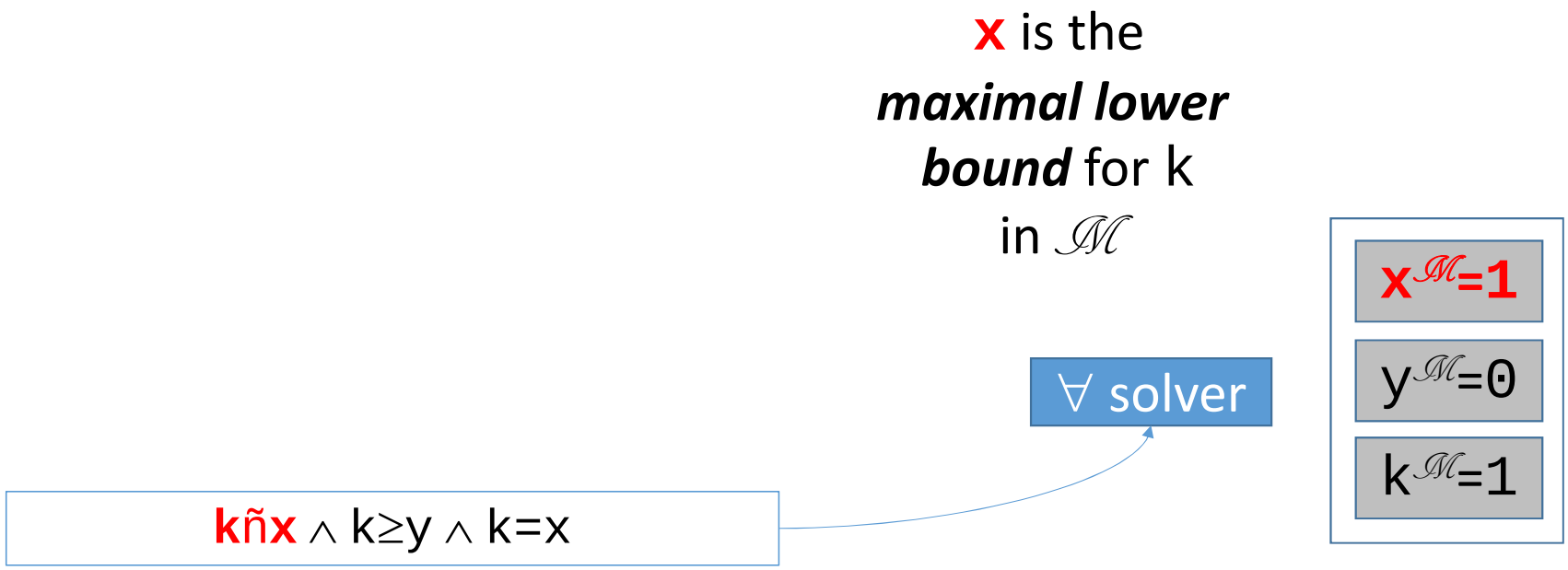
Counterexample-Guided ∀-Instantiation



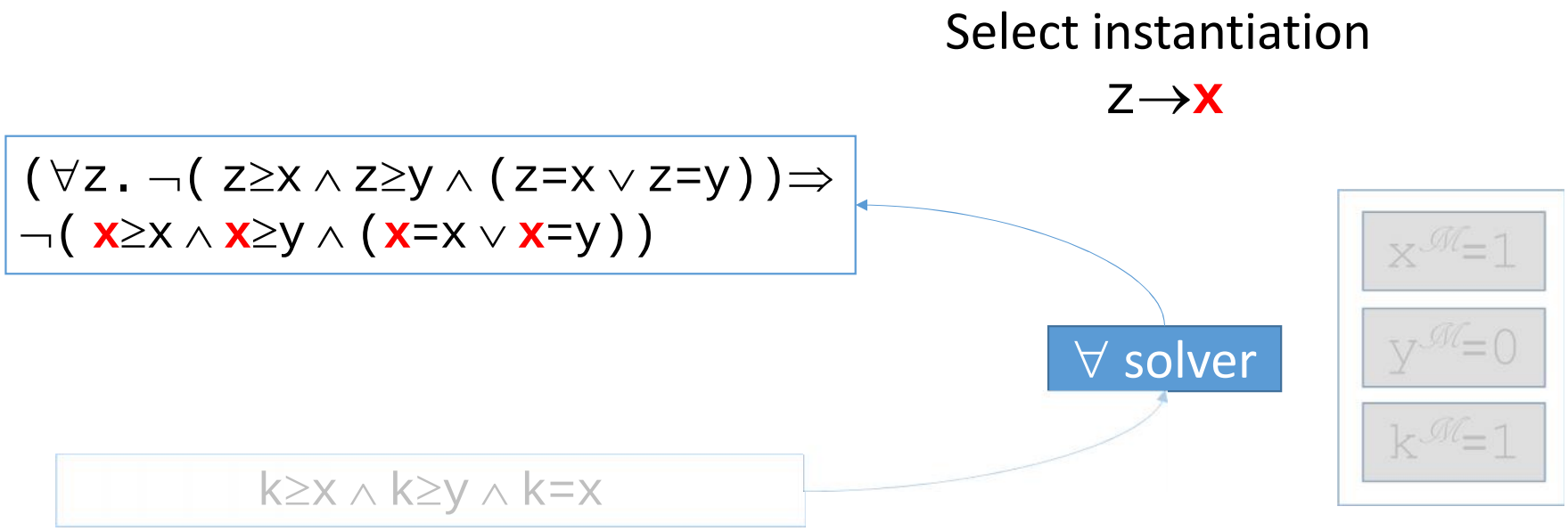
Counterexample-Guided ∀-Instantiation



Counterexample-Guided ∀-Instantiation



Counterexample-Guided ∀-Instantiation



Counterexample-Guided ∀-Instantiation

Requires a selection function $(M, \mathscr{M}, k) \rightarrow x$

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

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

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

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

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

Counterexample-Guided ∀-Instantiation

Quantifier Elimination Procedures

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

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

$\Leftarrow\Rightarrow$

Synthesis procedures for single-invocation properties

CEGQI Selection Functions

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

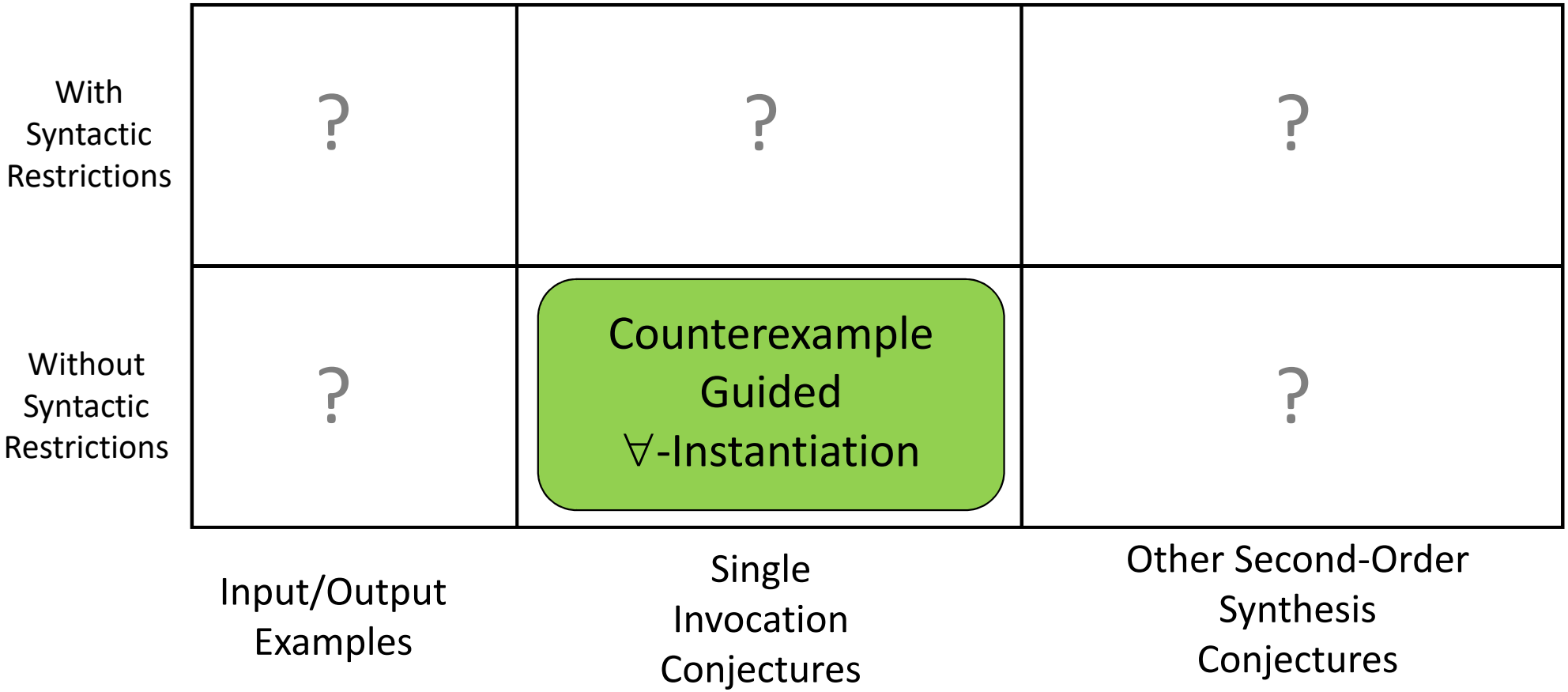
Ĝ *Each gives a **sound** and **complete** procedure for single invocation conjectures*

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 $\exists fg.\forall x.P(f(x),g(x),x)$
- **Sound** and **complete** for single invocation conjectures over LIA, LRA

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

Overview

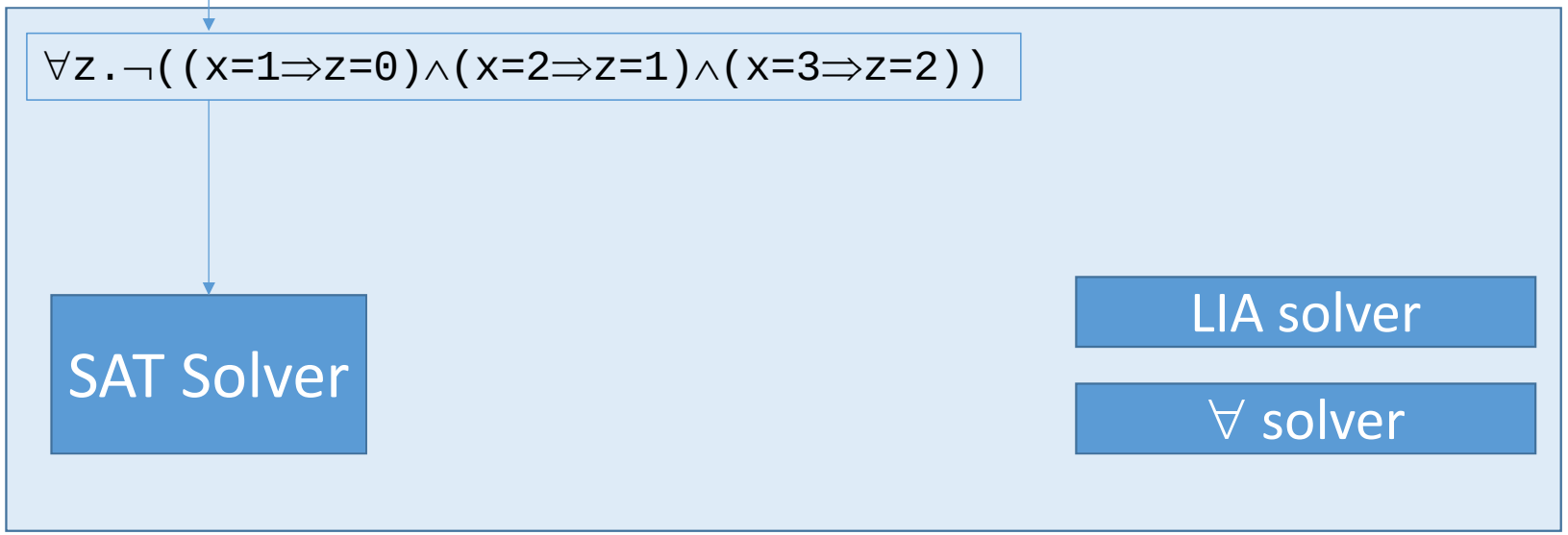

With Syntactic Restrictions
Without Syntactic Restrictions
?
?
?
?
Counterexample Guided ∀-Instantiation
?
Input/Output Examples
Single Invocation Conjectures
Other Second-Order Synthesis Conjectures

What if we apply CEGQI to I/O Examples?

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

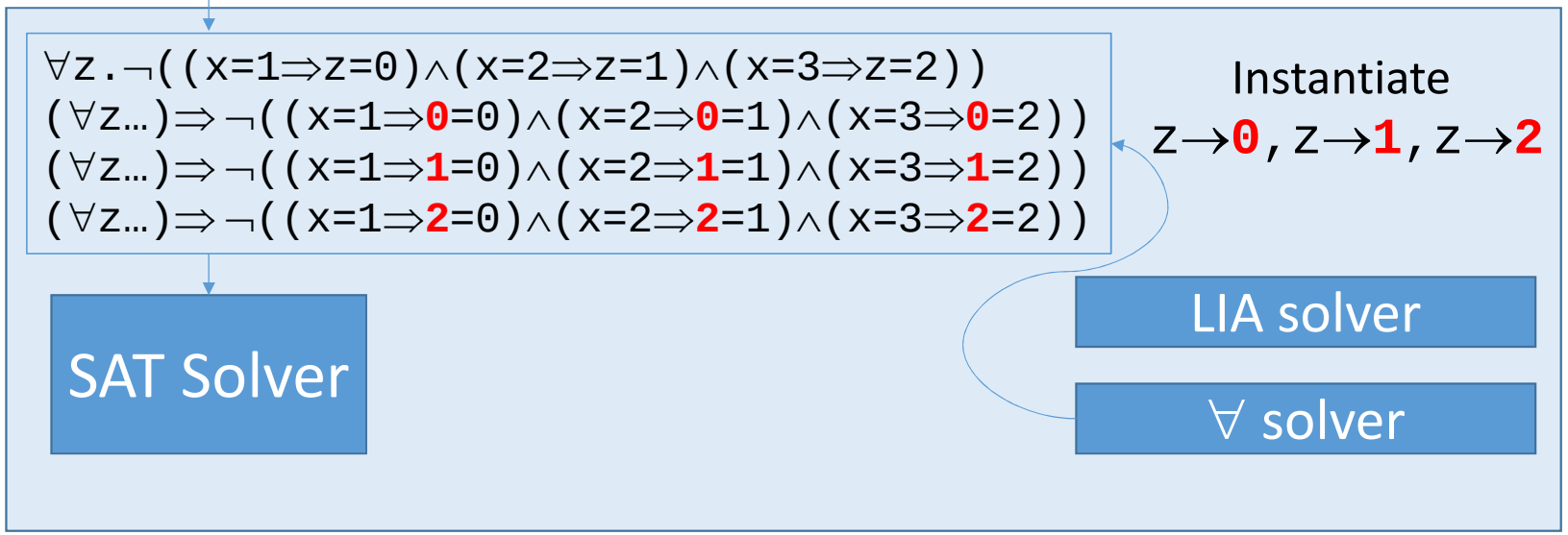
What if we apply CEGQI to I/O Examples?

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



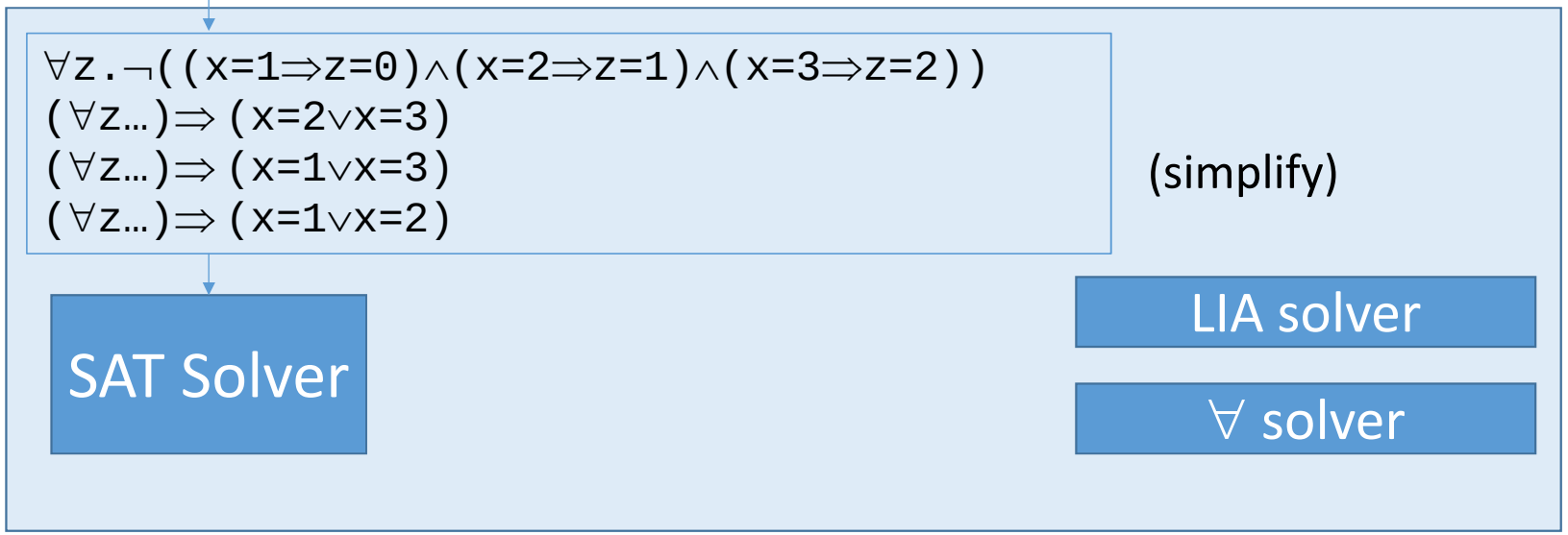
What if we apply CEGQI to I/O Examples?

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

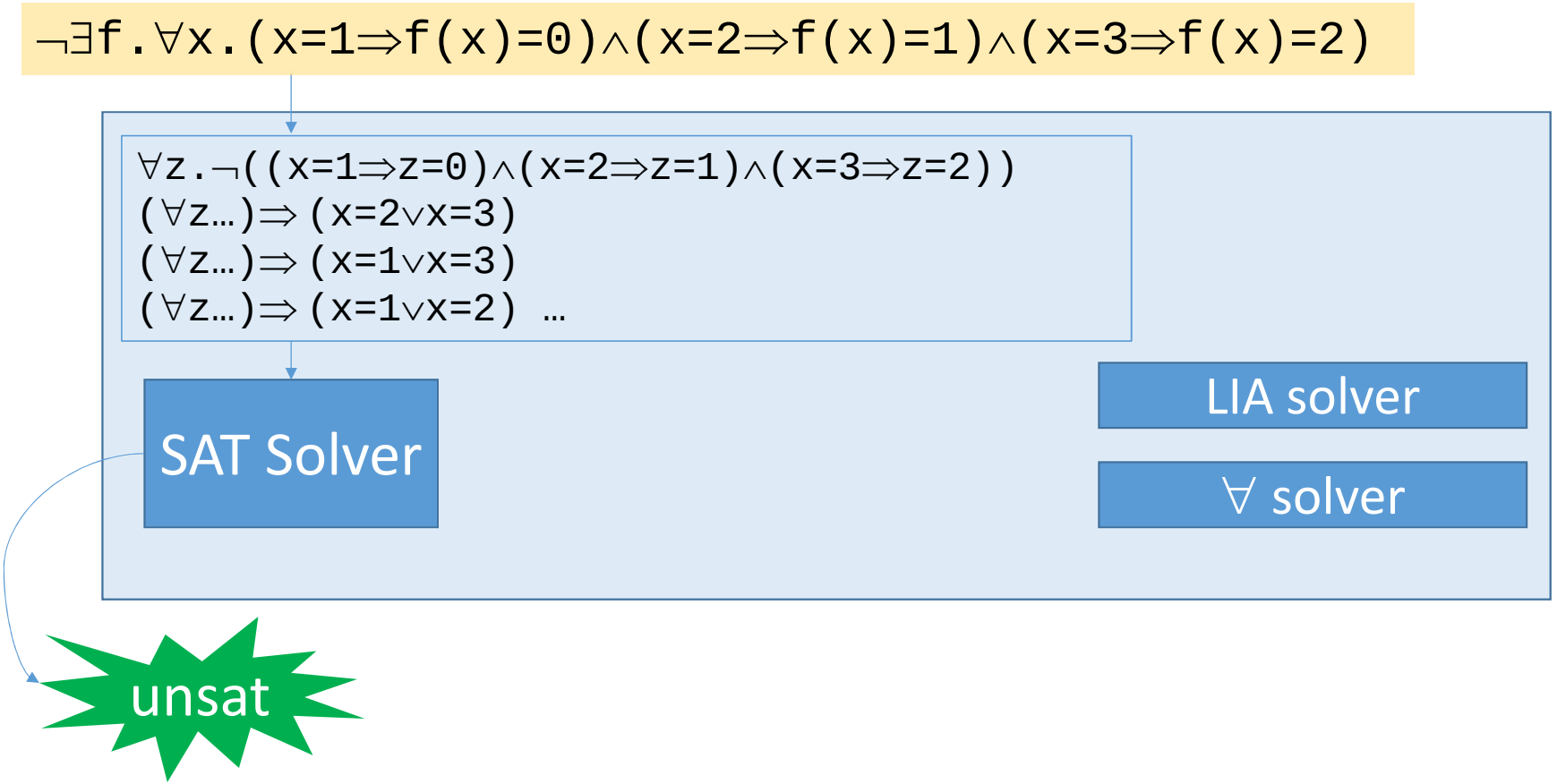


What if we apply CEGQI to I/O Examples?

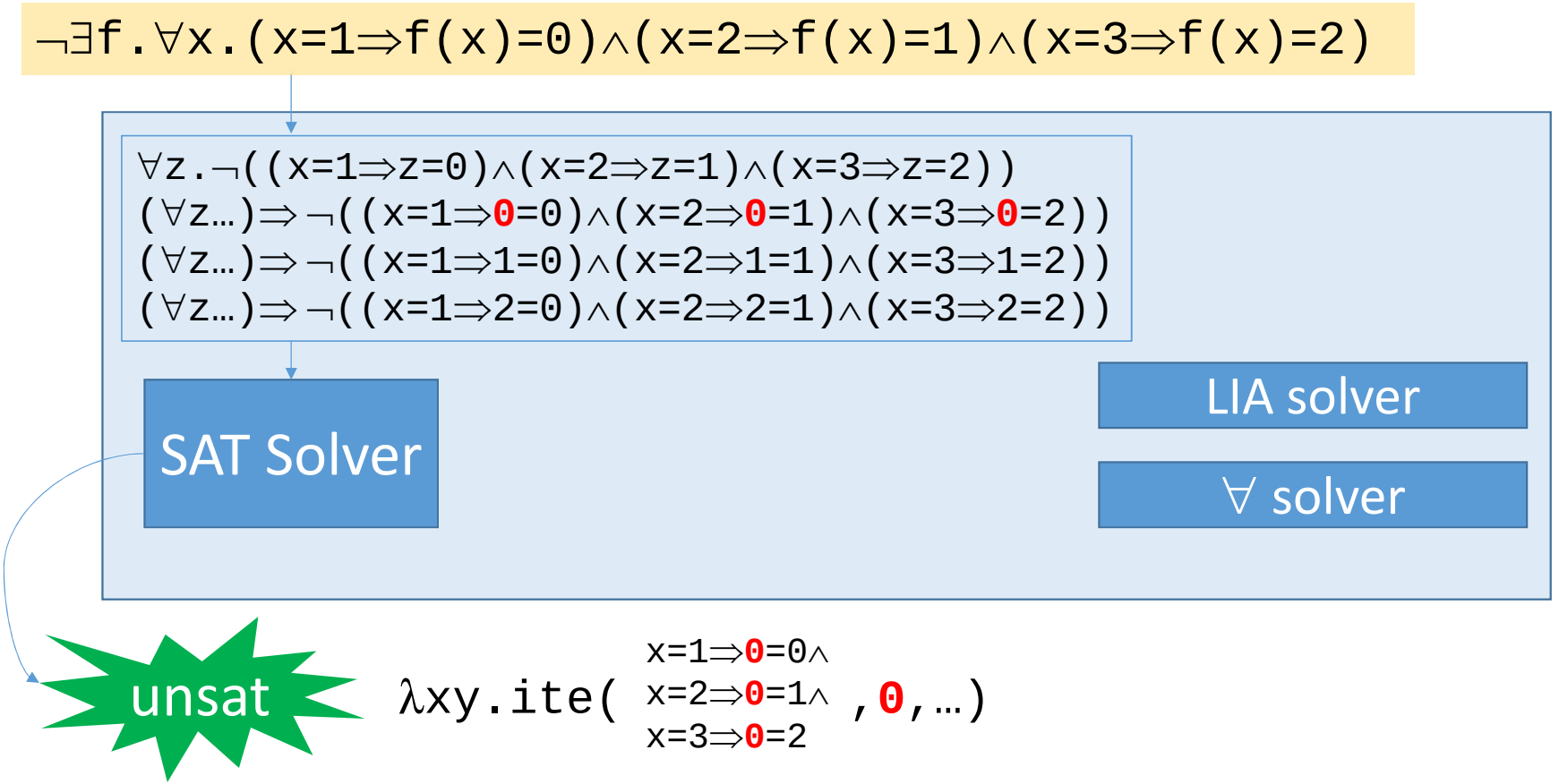
$\neg\exists f.\forall x.(x=1\Rightarrow f(x)=0)\wedge(x=2\Rightarrow f(x)=1)\wedge(x=3\Rightarrow f(x)=2)$



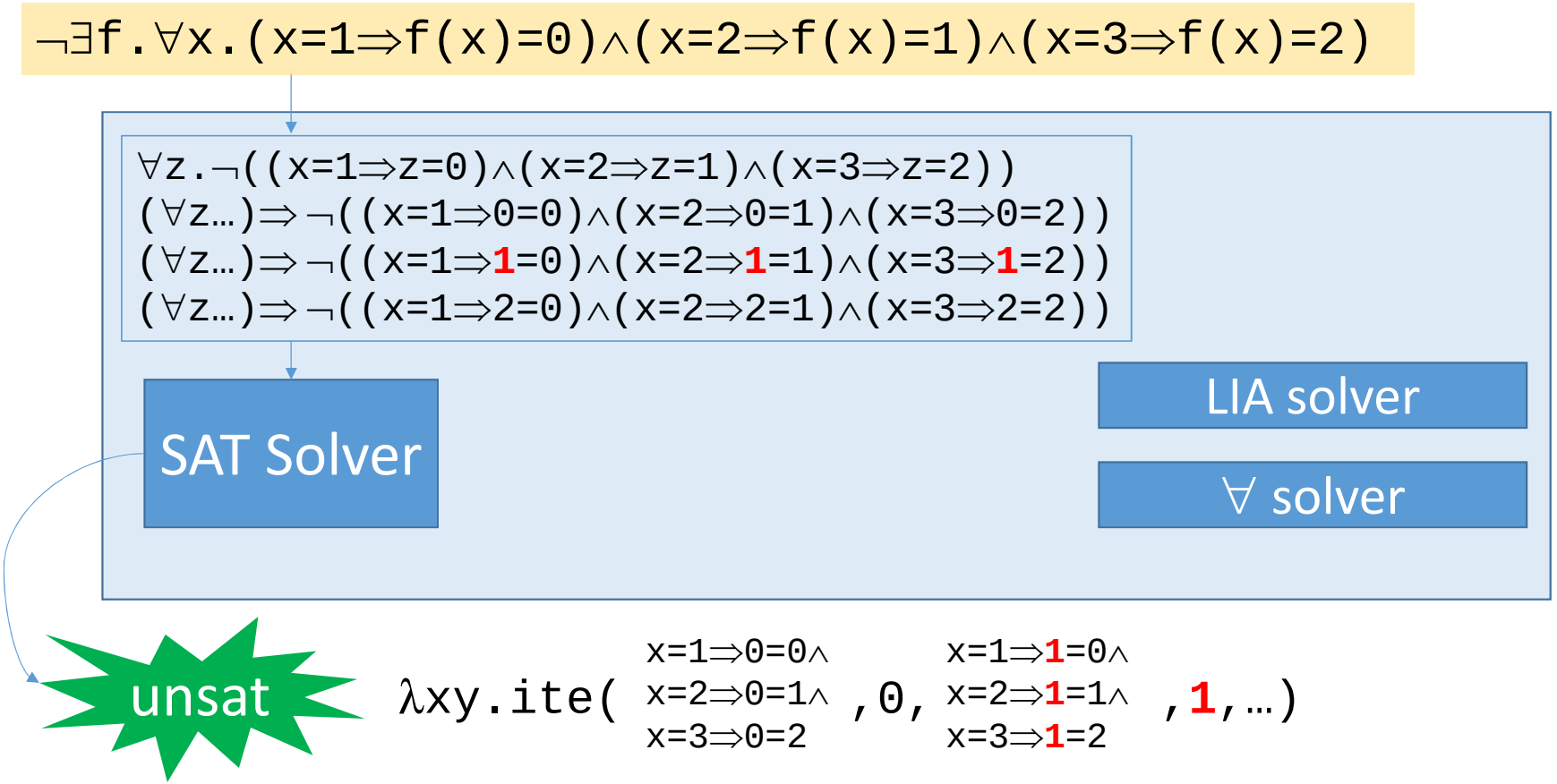
What if we apply CEGQI to I/O Examples?



What if we apply CEGQI to I/O Examples?

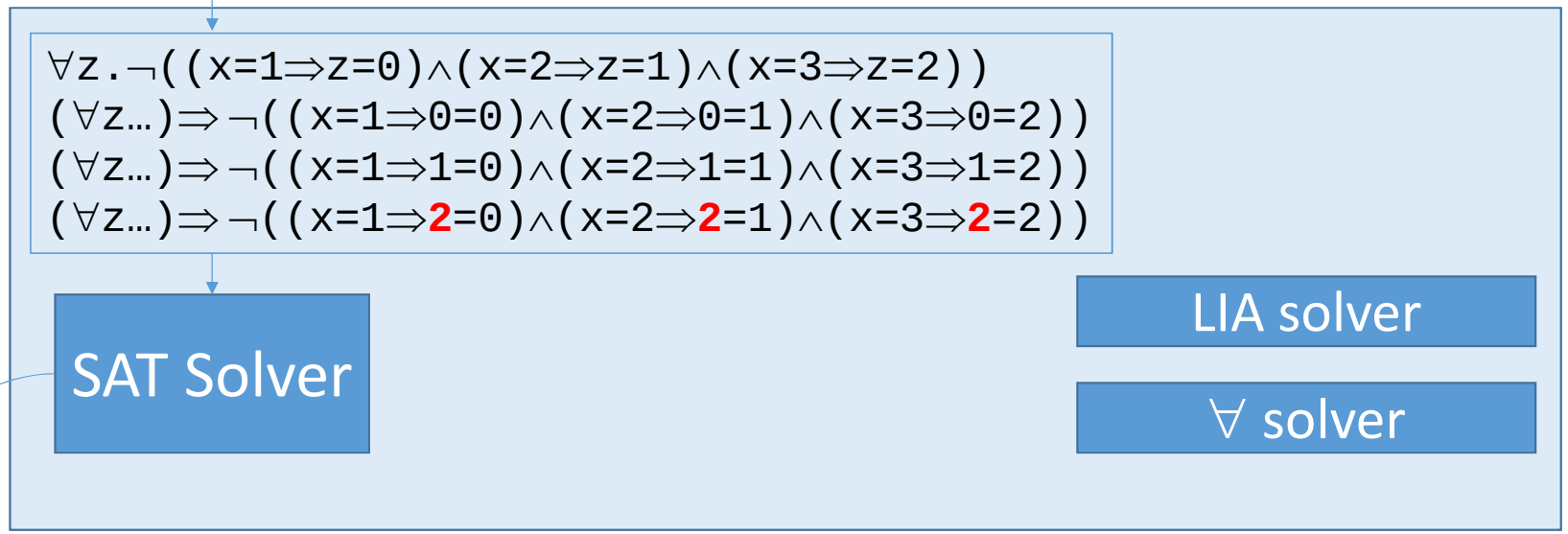


What if we apply CEGQI to I/O Examples?



What if we apply CEGQI to I/O Examples?

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

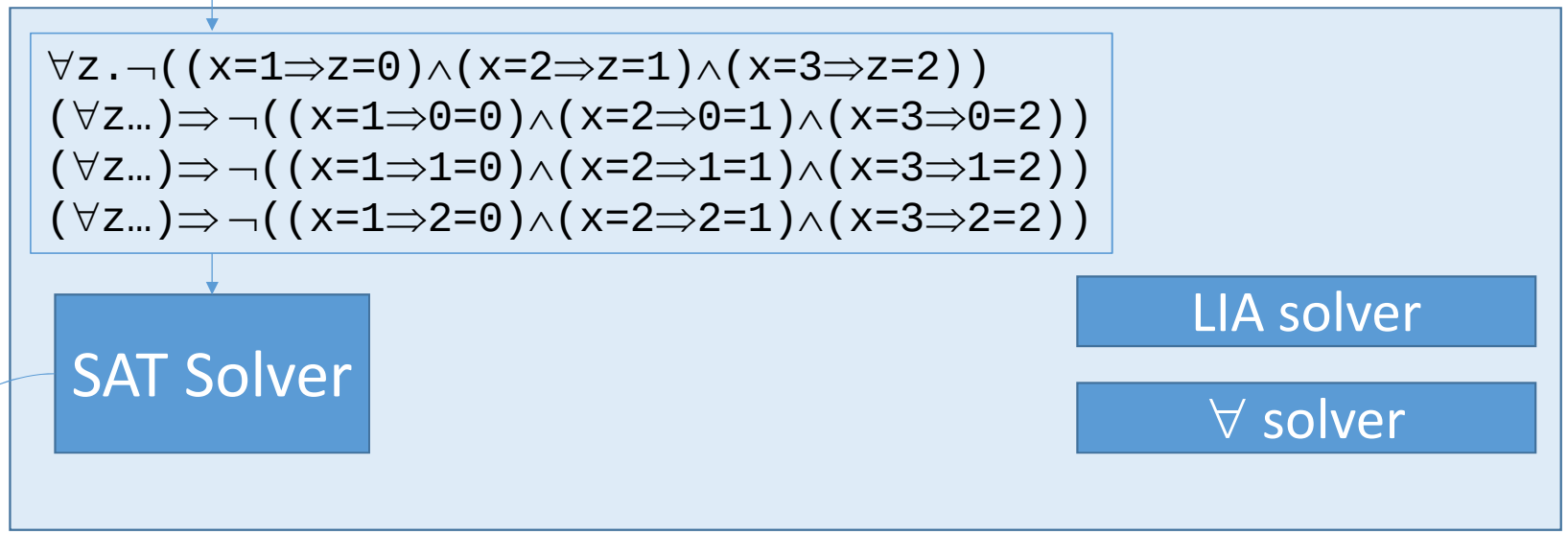


unsat

$\lambda xy.ite(\begin{array}{l}x=1\Rightarrow 0=0\wedge\\ x=2\Rightarrow 0=1\wedge\\ x=3\Rightarrow 0=2\end{array},0,\begin{array}{l}x=1\Rightarrow 1=0\wedge\\ x=2\Rightarrow 1=1\wedge\\ x=3\Rightarrow 1=2\end{array},1,2)$

What if we apply CEGQI to I/O Examples?

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



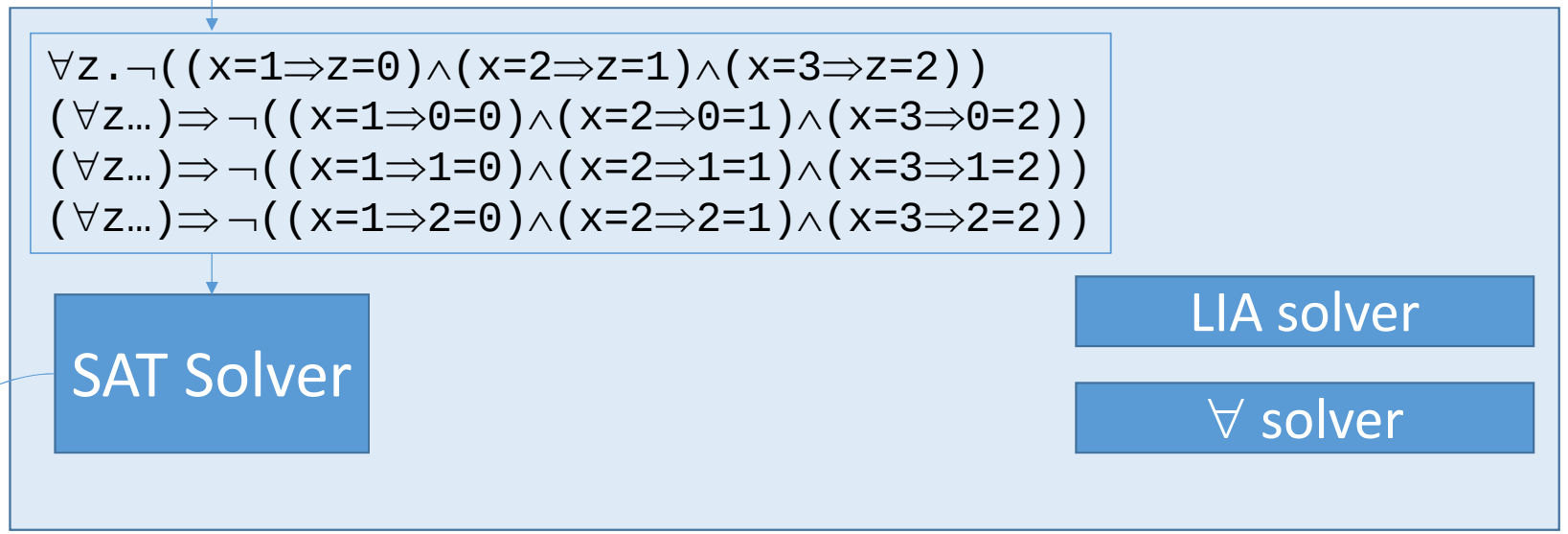
unsat

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

$\Rightarrow$ simplify

What if we apply CEGQI to I/O Examples?

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

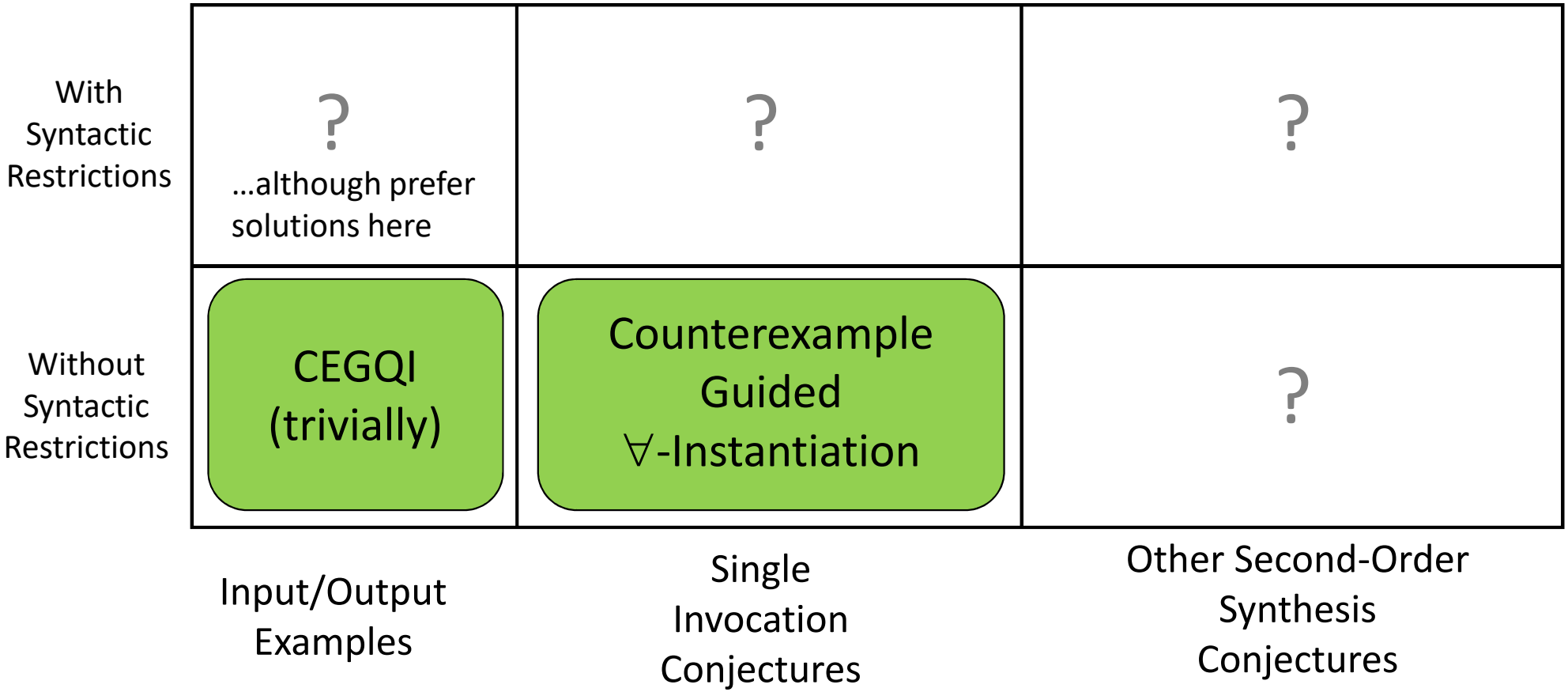


unsat

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

⇒ *Produces **trivial solution** (input/output table)*

Overview


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

What if there are syntactic restrictions?

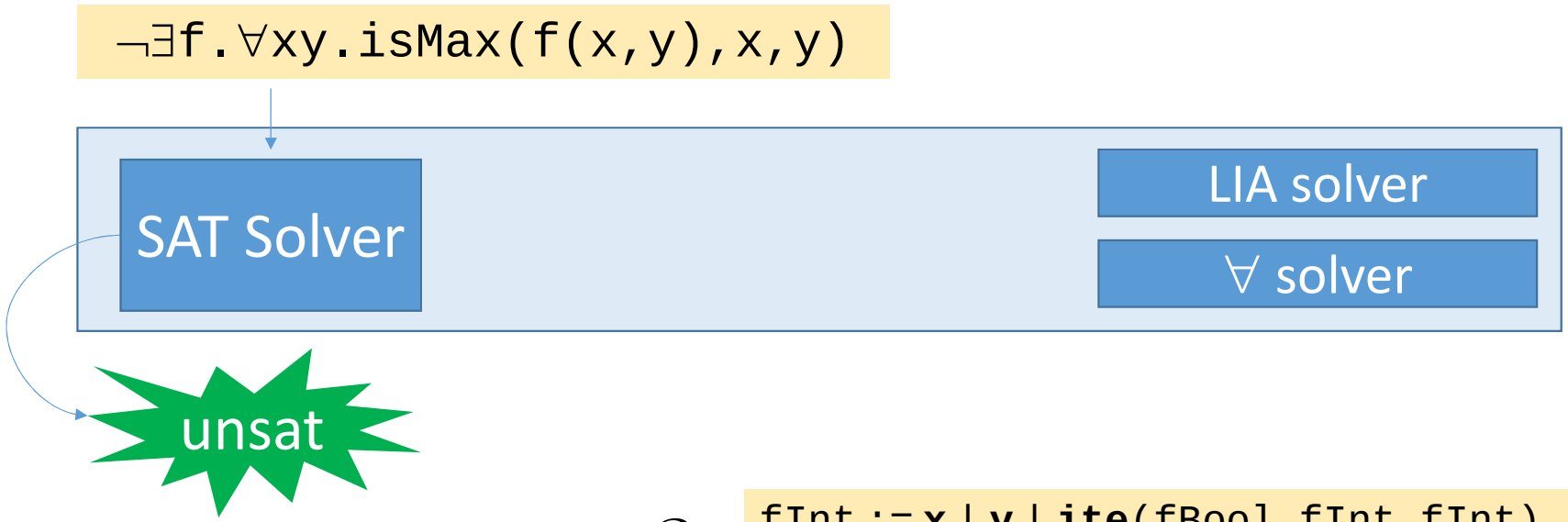
$\neg\exists f.\forall xy.isMax(f(x,y),x,y)$

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

$\mathcal{R}$:

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

What if there are syntactic restrictions?

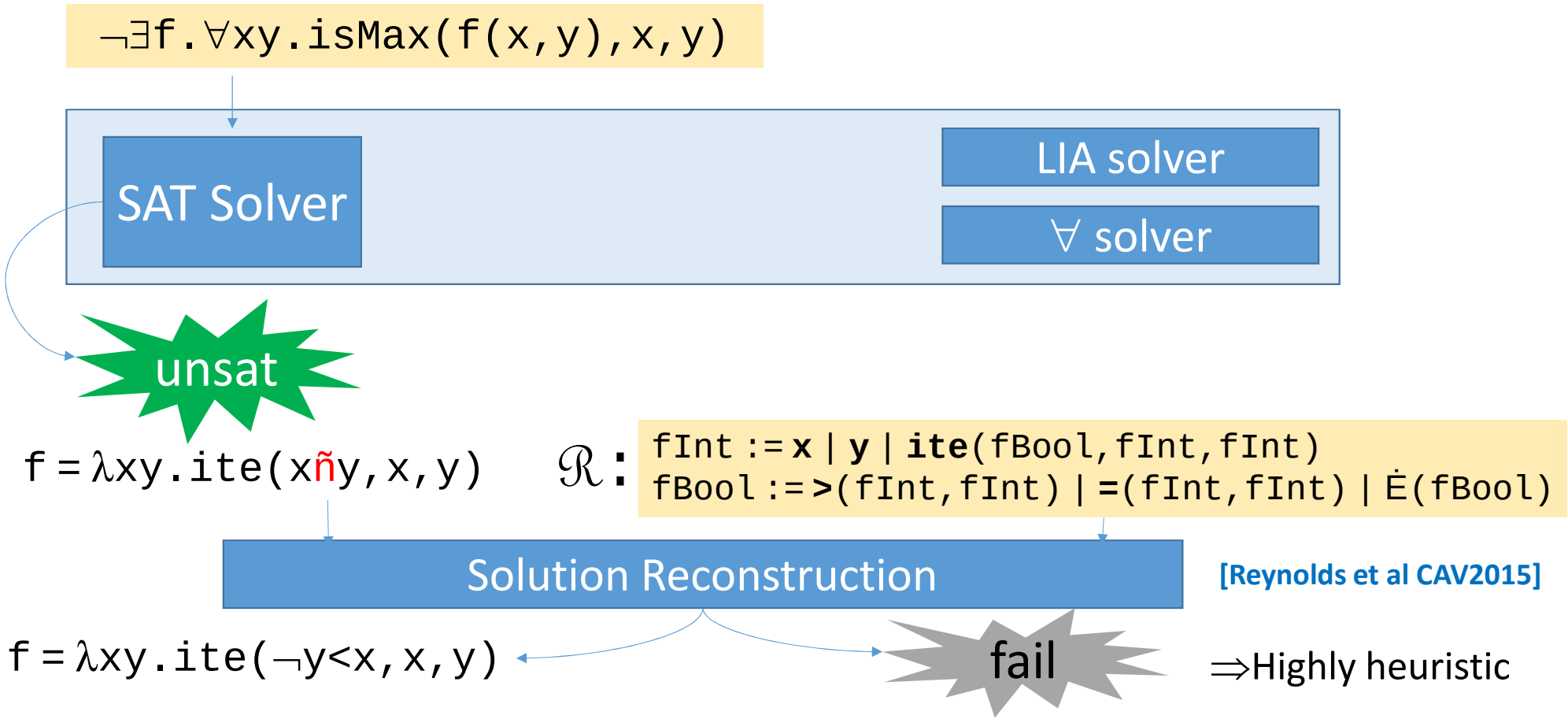


$f = \lambda xy.ite(x \tilde{n} y, x, y)$

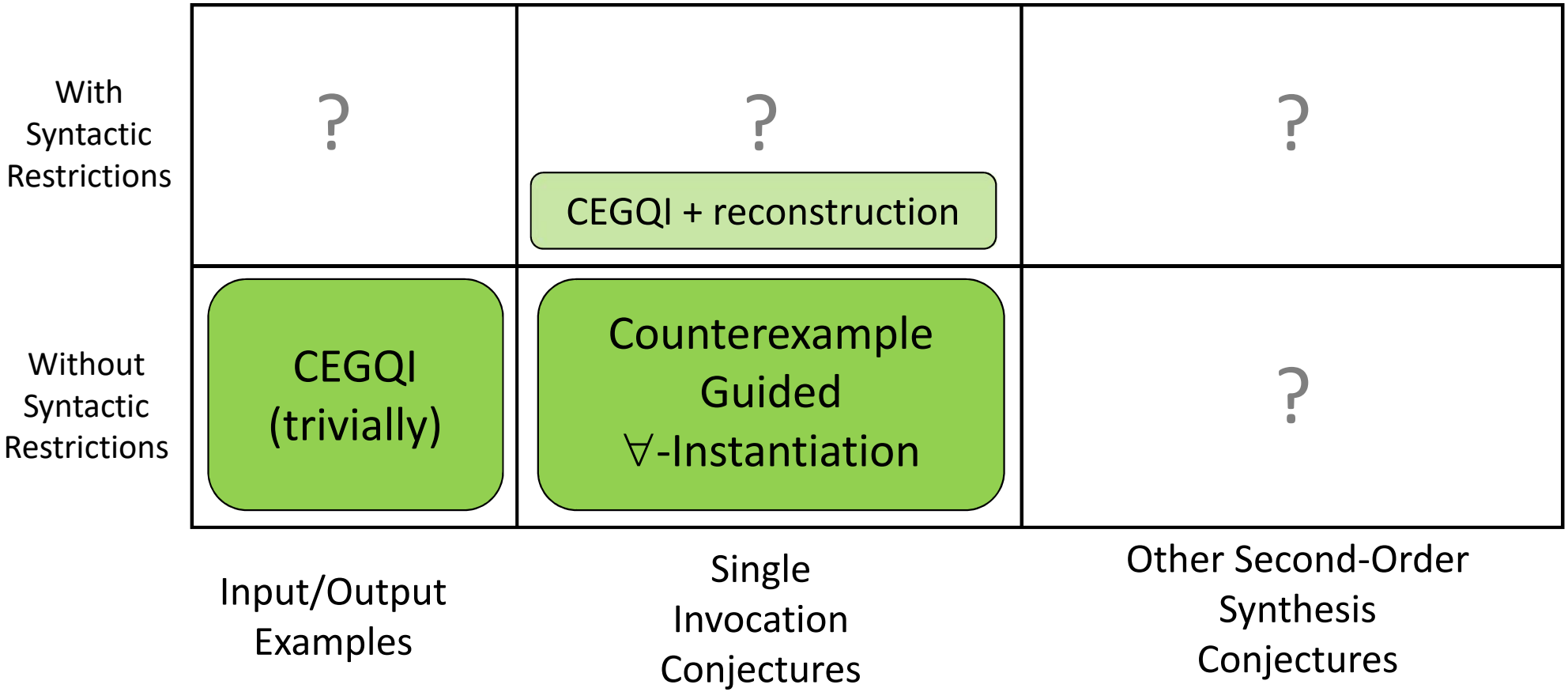
$\mathcal{R}$:

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

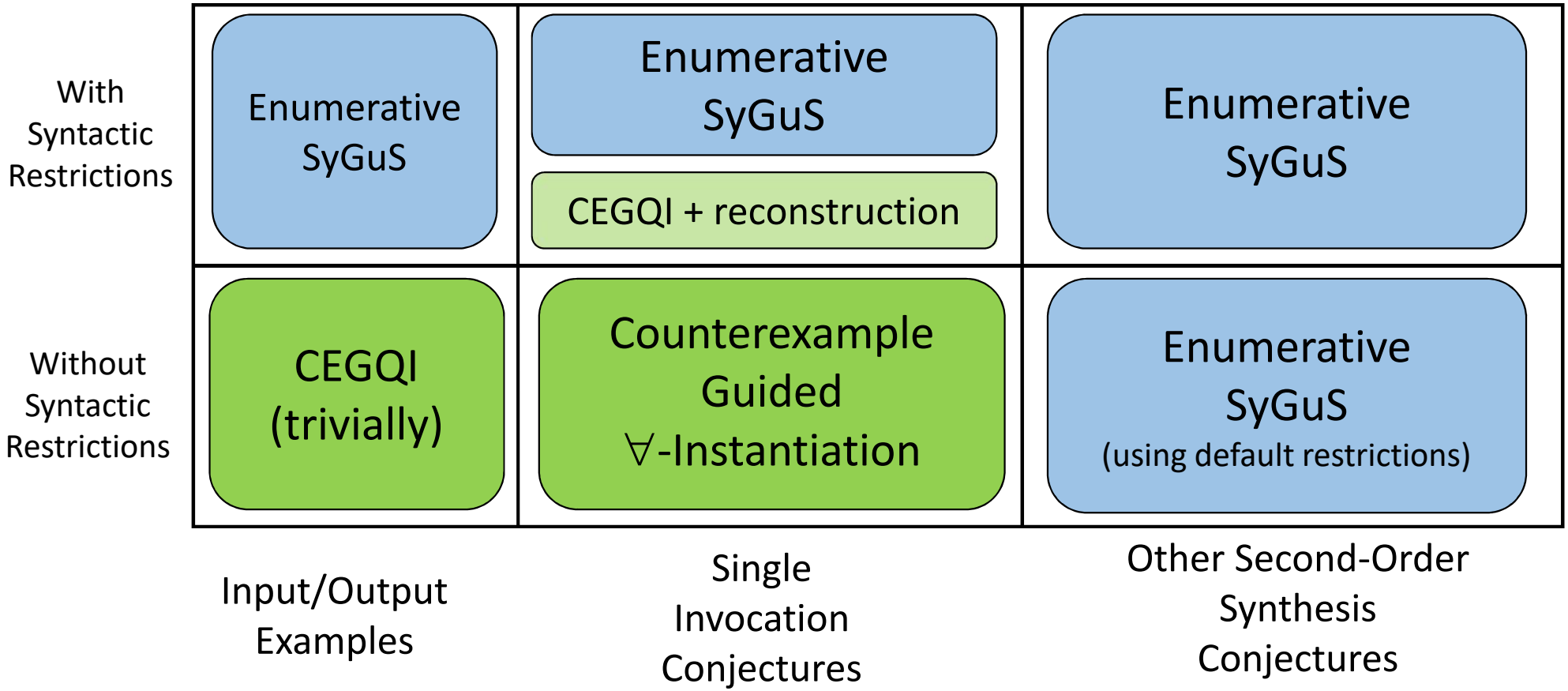
What if there are syntactic restrictions?



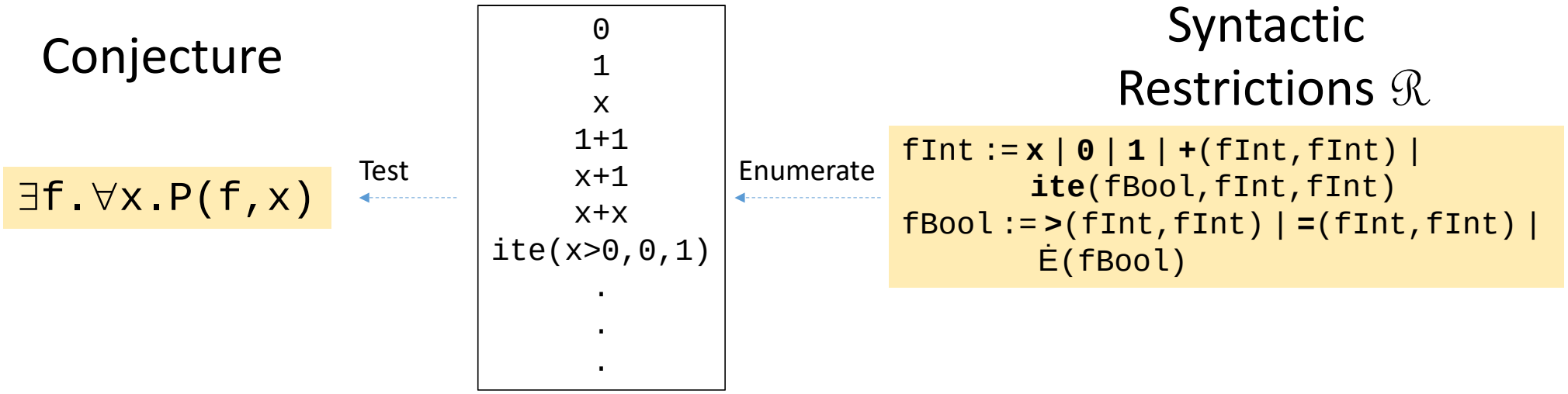
Overview


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

For Everything Else…

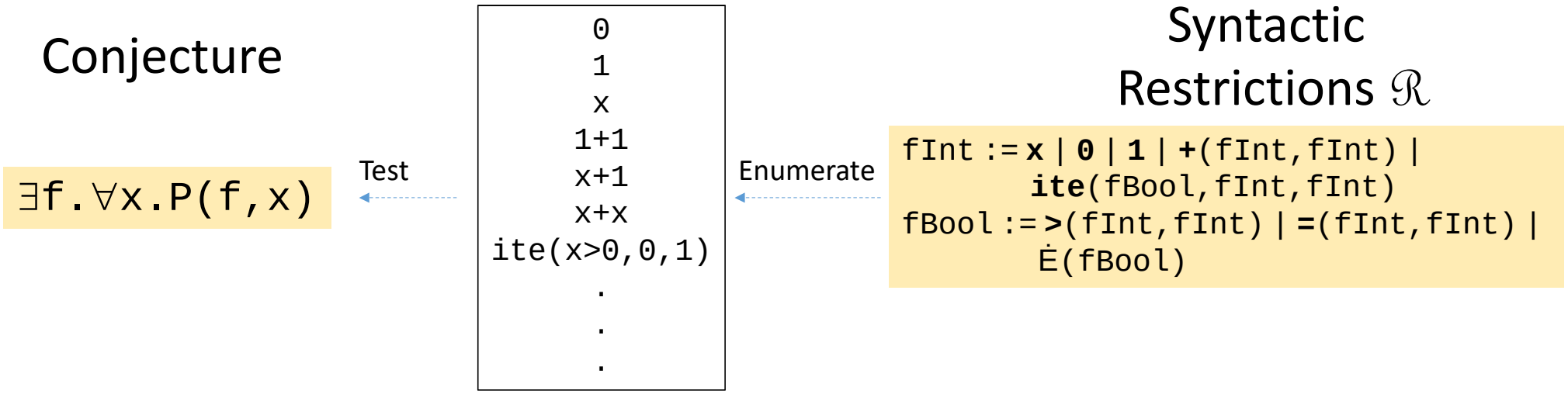


Enumerative Syntax-Guided Synthesis



- 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



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

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

Enumerative Syntax-Guided Synthesis in SMT

Conjecture

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

$\mathcal{R}$:

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

Enumerative Syntax-Guided Synthesis in SMT

Conjecture

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

$\mathcal{R}$:

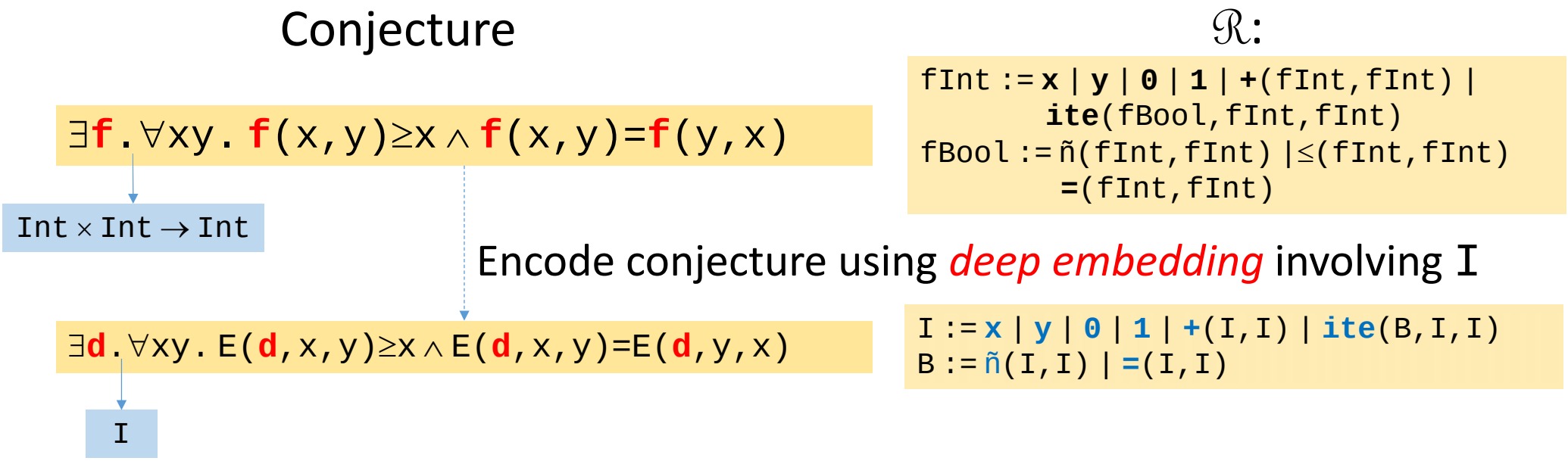
```
fInt := x | y | 0 | 1 | +(fInt,fInt) |
        ite(fBool,fInt,fInt)
fBool := ñ(fInt,fInt) |≤(fInt,fInt)
         =(fInt,fInt)
```

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

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

$\Rightarrow$ Involves flattening, minimization

Enumerative Syntax-Guided Synthesis in SMT



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

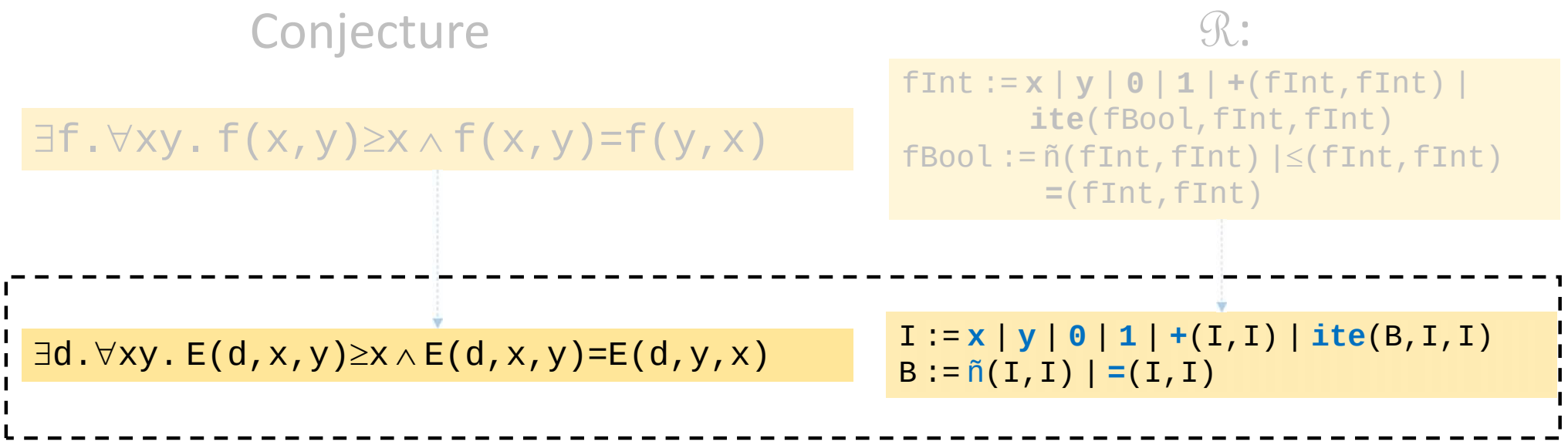
$\mathcal{R}$:

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

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

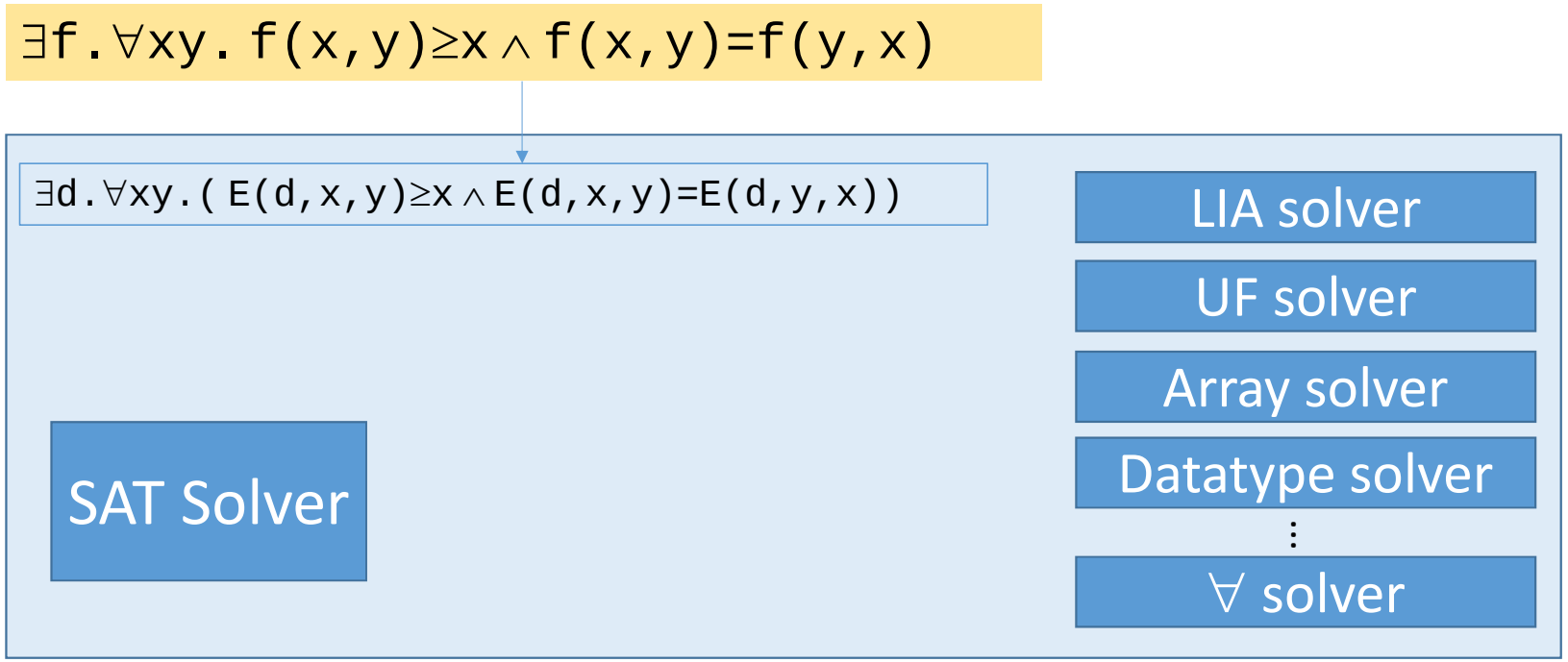


Ĝ *Solvable by combination of datatypes, LIA, UF, and ∀-instantiation*

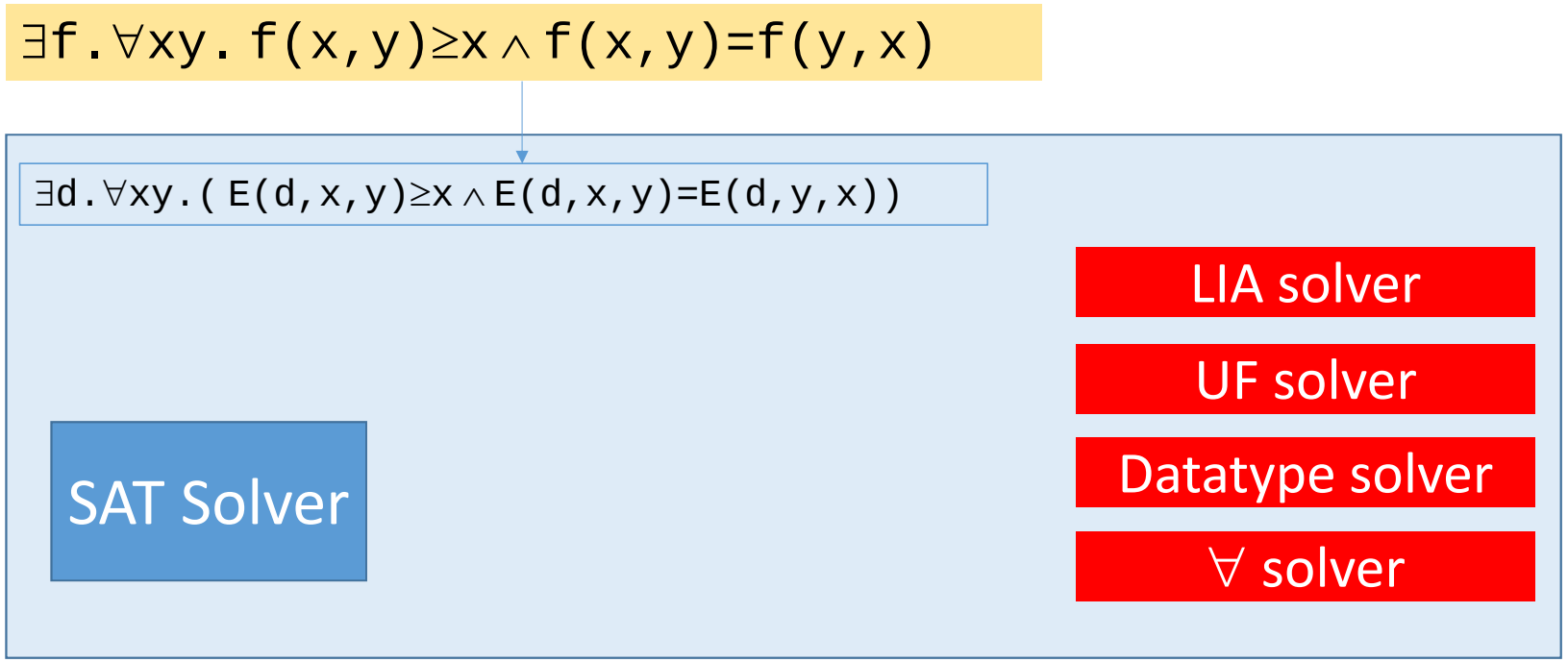
Enumerative Syntax-Guided Synthesis in SMT

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

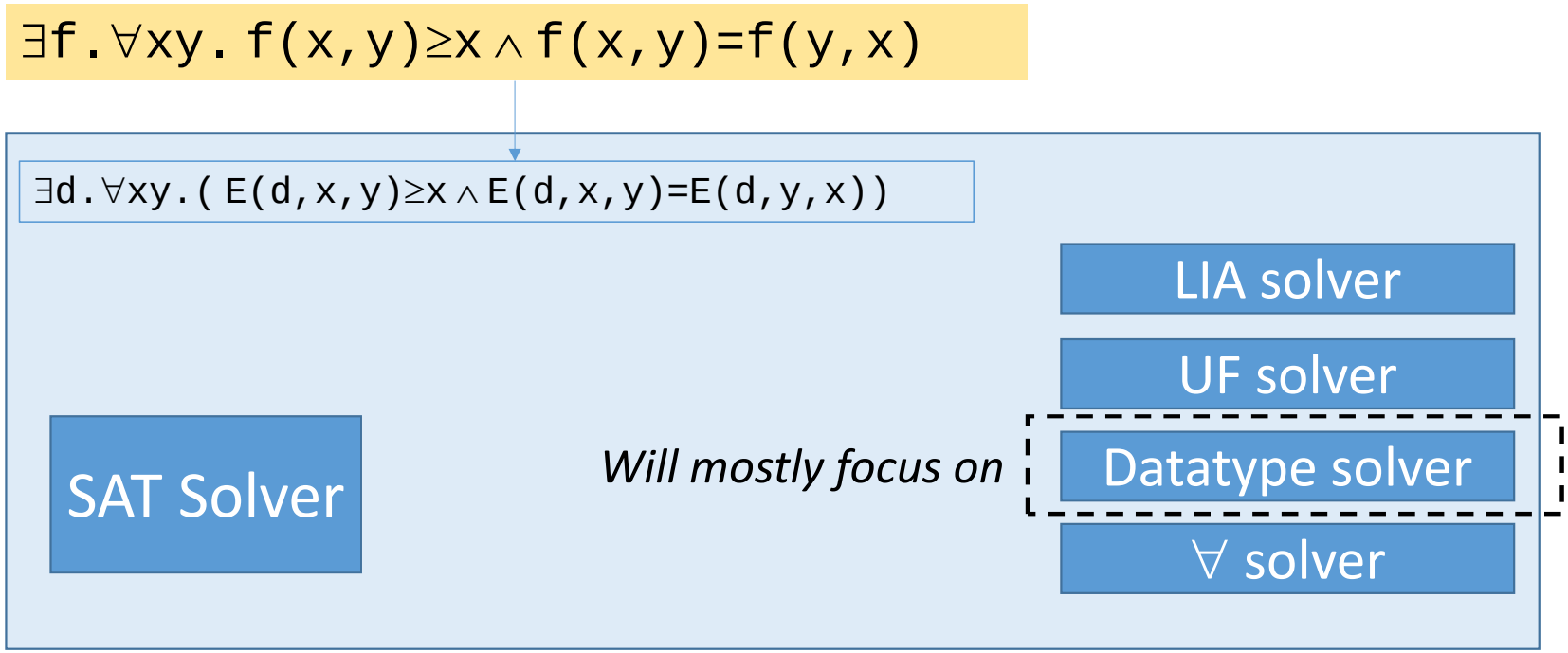
Enumerative Syntax-Guided Synthesis in SMT



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 := ñ(I,I) | =(I,I)| ¬(B)
```

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

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

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

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

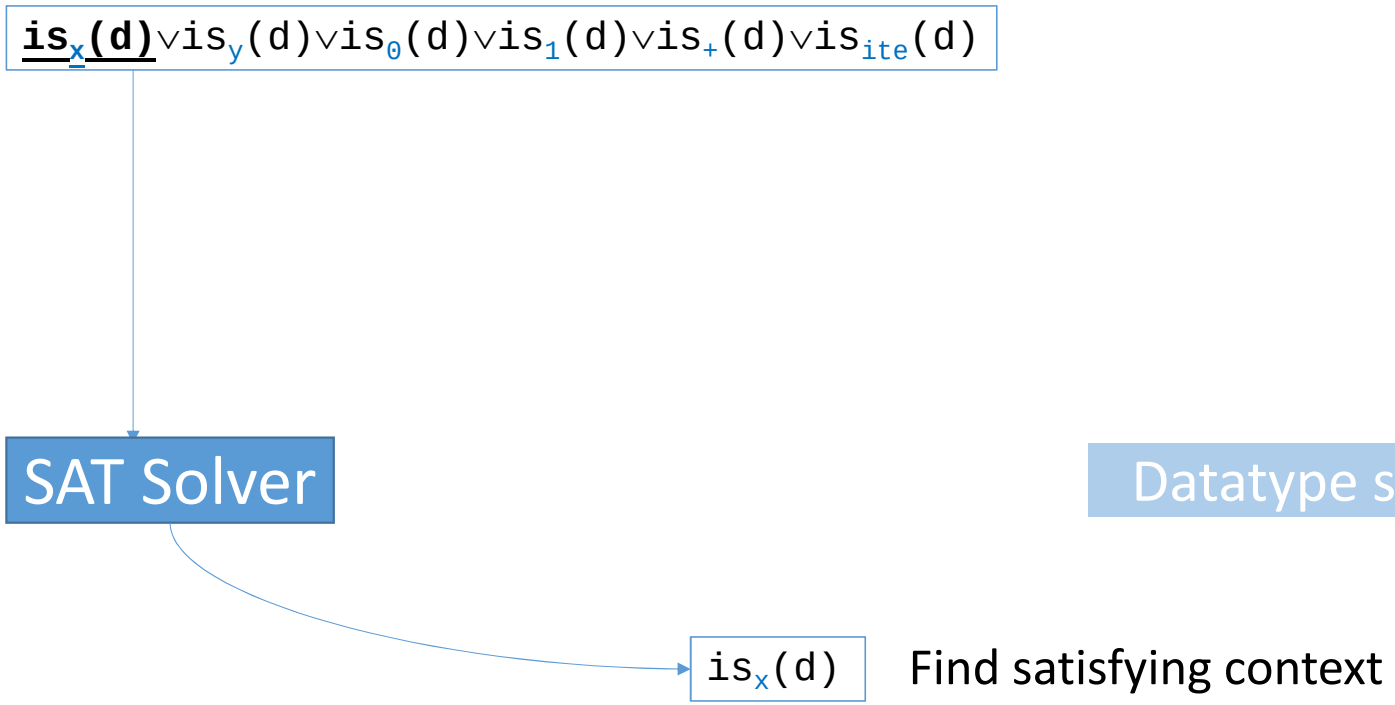
Split on top symbol of d

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

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



Enumerative Syntax-Guided Synthesis in SMT

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

$is_x(d)$

$d^{\mathcal{M}}=x$

Find model $\mathcal{M}$

Enumerative Syntax-Guided Synthesis in SMT

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

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

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

SAT Solver

Datatype solver

$\forall$ solver

$d^{\mathcal{M}} = \mathbf{x}$

Enumerative Syntax-Guided Synthesis in SMT

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

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

Simplify

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

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

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

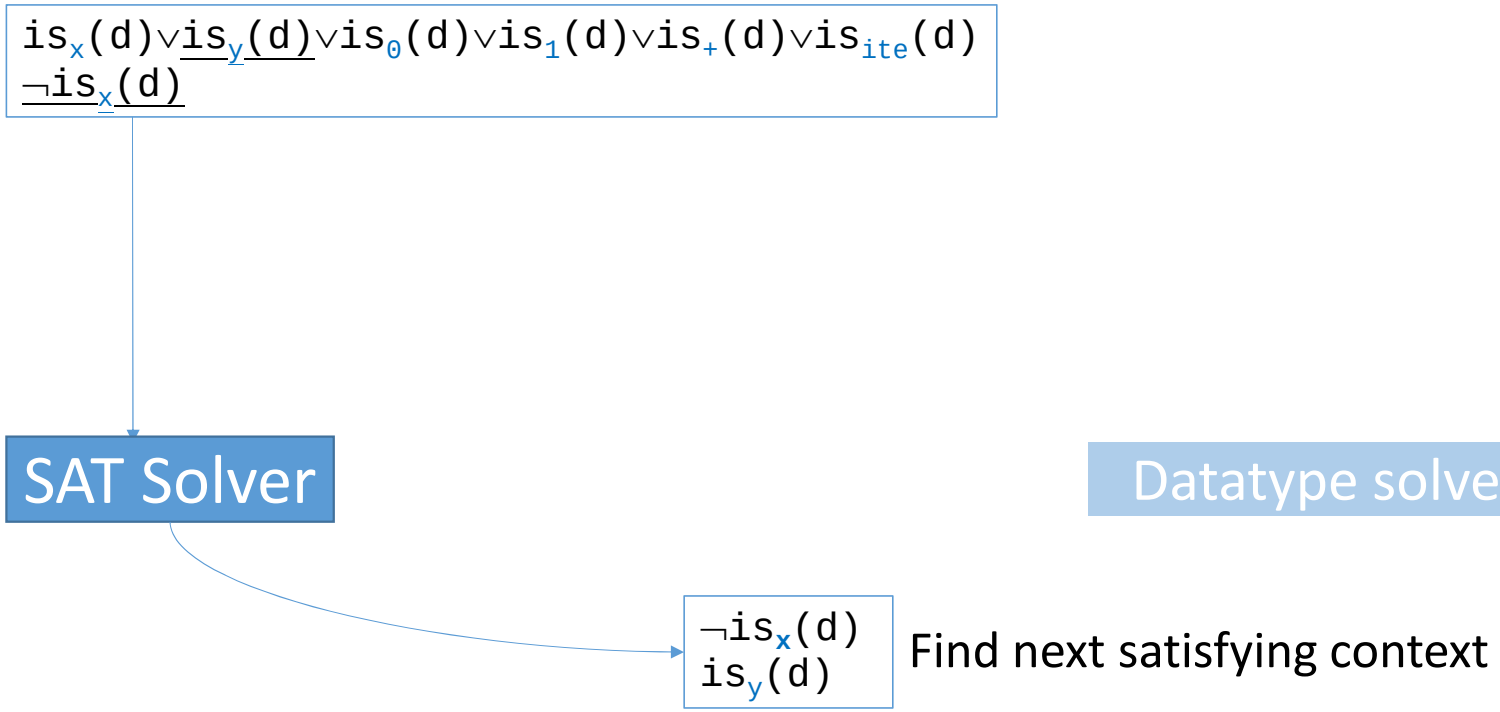
Simplify

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

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



Enumerative Syntax-Guided Synthesis in SMT

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

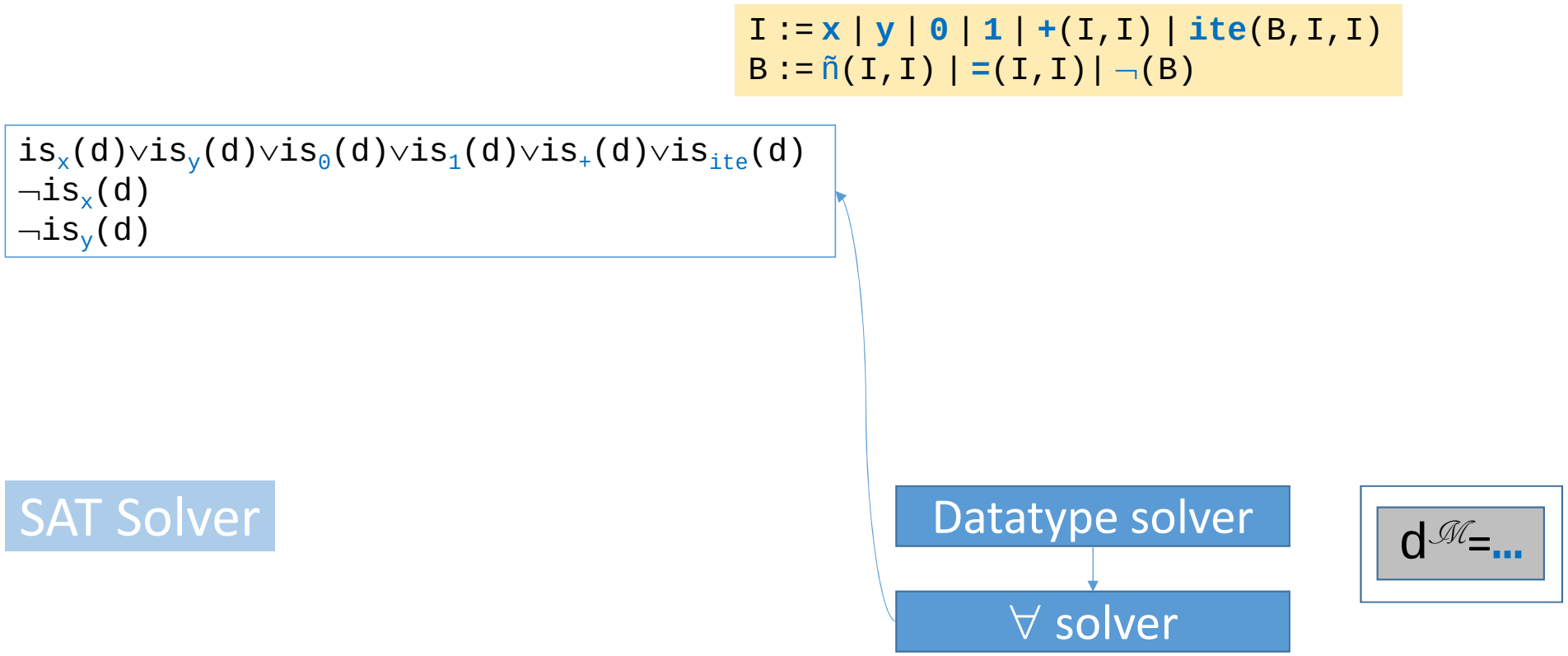
Datatype solver

$d^{\mathscr{M}}=y$

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

Find next model $\mathscr{M}$

Enumerative Syntax-Guided Synthesis in SMT



Enumerative Syntax-Guided Synthesis in SMT

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

$d^{\mathcal{M}}$=…

Enumerative Syntax-Guided Synthesis in SMT

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

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

$\neg is_x($

…

Optimization:

Only consider terms $\mathbf{d}^{\mathscr{M}}$ *whose* ***analog***

is unique up to theory-specific simplification ↓

SAT Solver

Datatype solver

∀ solver

$d^{\mathscr{M}}$=…

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ñ(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 $\mathbf{d}^{\mathcal{M}}$ *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

$\mathcal{M}_= \ldots$

Enumerative Syntax-Guided Synthesis in SMT

```
I := x | y | 0 | 1 | +(I,I) | ite(B,I,I)
B := ñ(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 $\mathbf{d}^{\mathcal{M}}$ *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

$\mathcal{M}$=…

Enumerative Syntax-Guided Synthesis in SMT

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

…

Maintain a database
of candidates whose simplified
analogs are pairwise unique

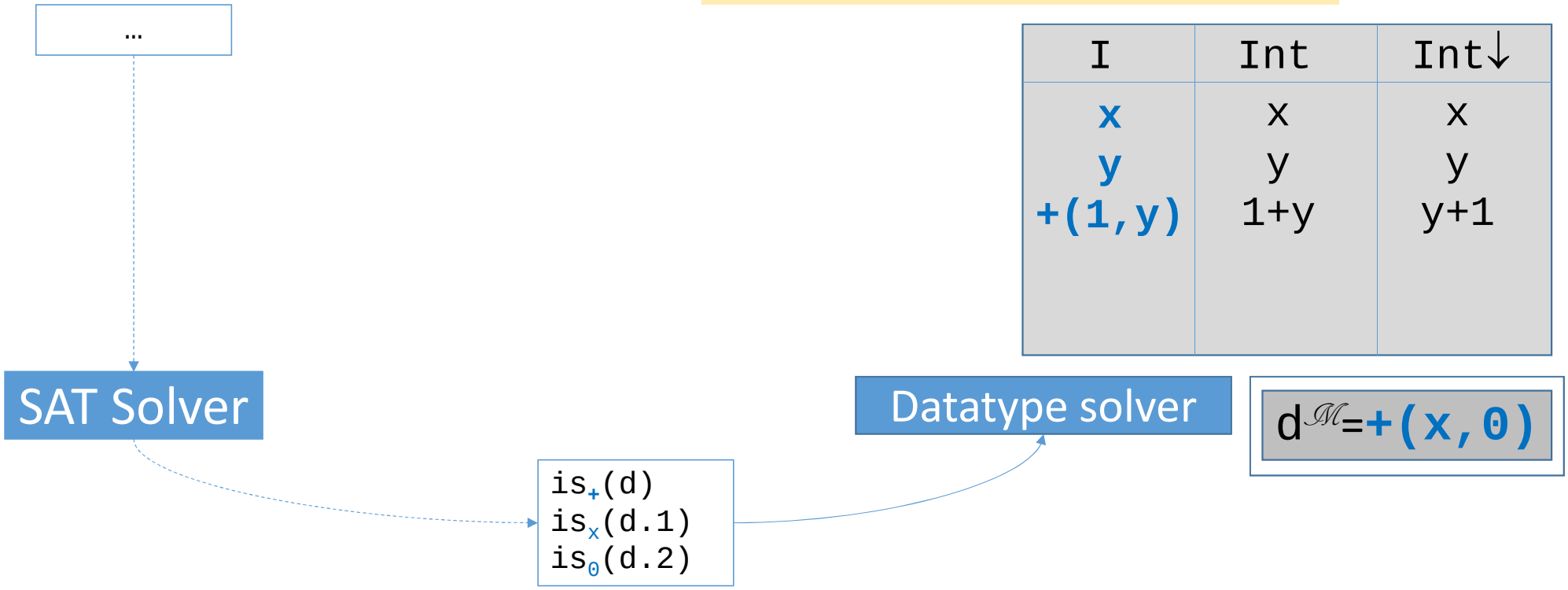
I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

SAT Solver

Datatype solver

Enumerative Syntax-Guided Synthesis in SMT

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



I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

Enumerative Syntax-Guided Synthesis in SMT

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

…

I	Int	Int↓
x	x	x
y	y	y
+(1,y)	1+y	y+1

SAT Solver

Datatype solver

$d^{\mathcal{M}}$=**+(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 := ñ(I,I) | =(I,I) | ¬(B)
```

…

I	Int	Int↓
x	x	**x**
y	y	y
+(1,y)	1+y	y+1

SAT Solver

Datatype solver

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

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

…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 := ñ(I,I) | =(I,I)| ¬(B)
```

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

- 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

$d^{\mathcal{M}}$=**+(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

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

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

- Can be applied for any subterm of d:

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

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

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)**"

- Can be applied for any subterm of d:

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

"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 := ñ(I,I) | =(I,I)| ¬(B)
```

I	Int	Int↓
		x
		y
		y+1

Optimization for I/O examples:

Only consider terms $\mathbf{d}^{\mathscr{M}}$ *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 := ñ(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 := ñ(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 := ñ(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

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

SAT Solver

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

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

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

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

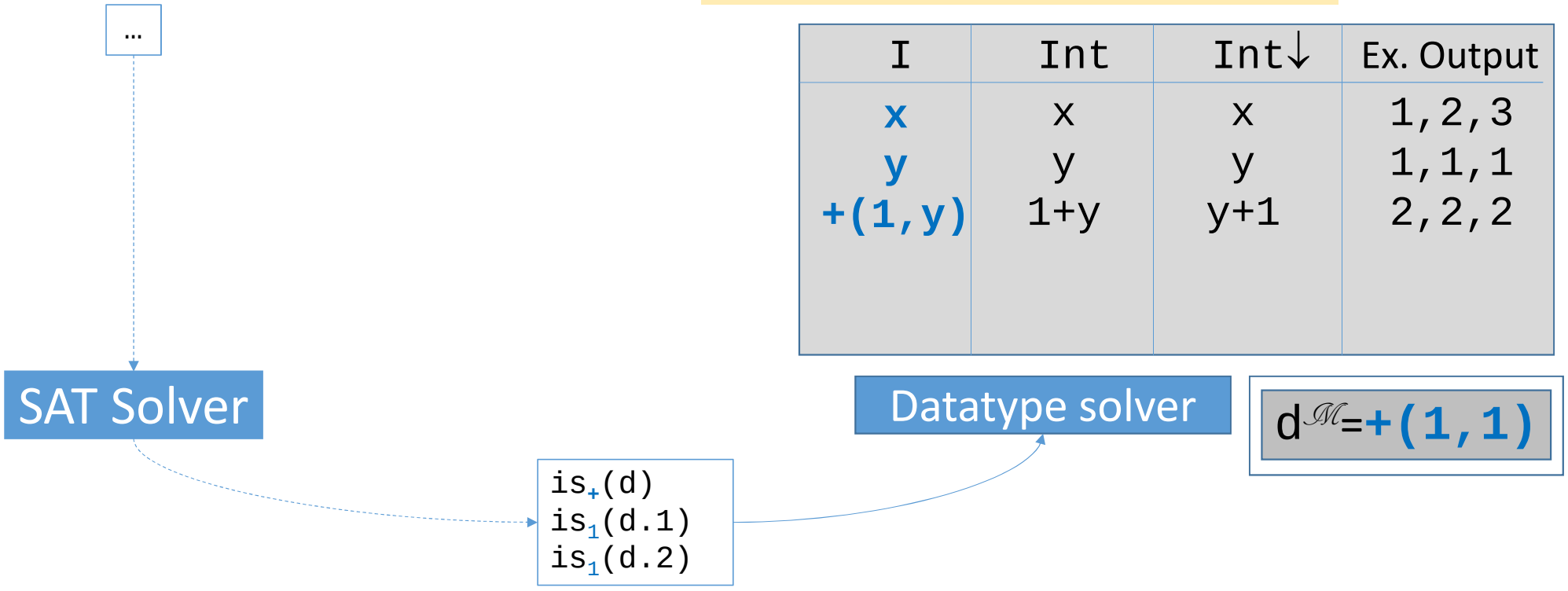
I	Int	Int↓	Ex. Output
x	x	x	1,2,3
y	y	y	1,1,1
+(1,y)	1+y	y+1	2,2,2

SAT Solver

Datatype solver

Enumerative SyGuS in SMT for I/O Examples

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

Enumerative SyGuS in SMT for I/O Examples

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

$d^{\mathcal{M}}$=**+(1,1)**

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

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 := ñ(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

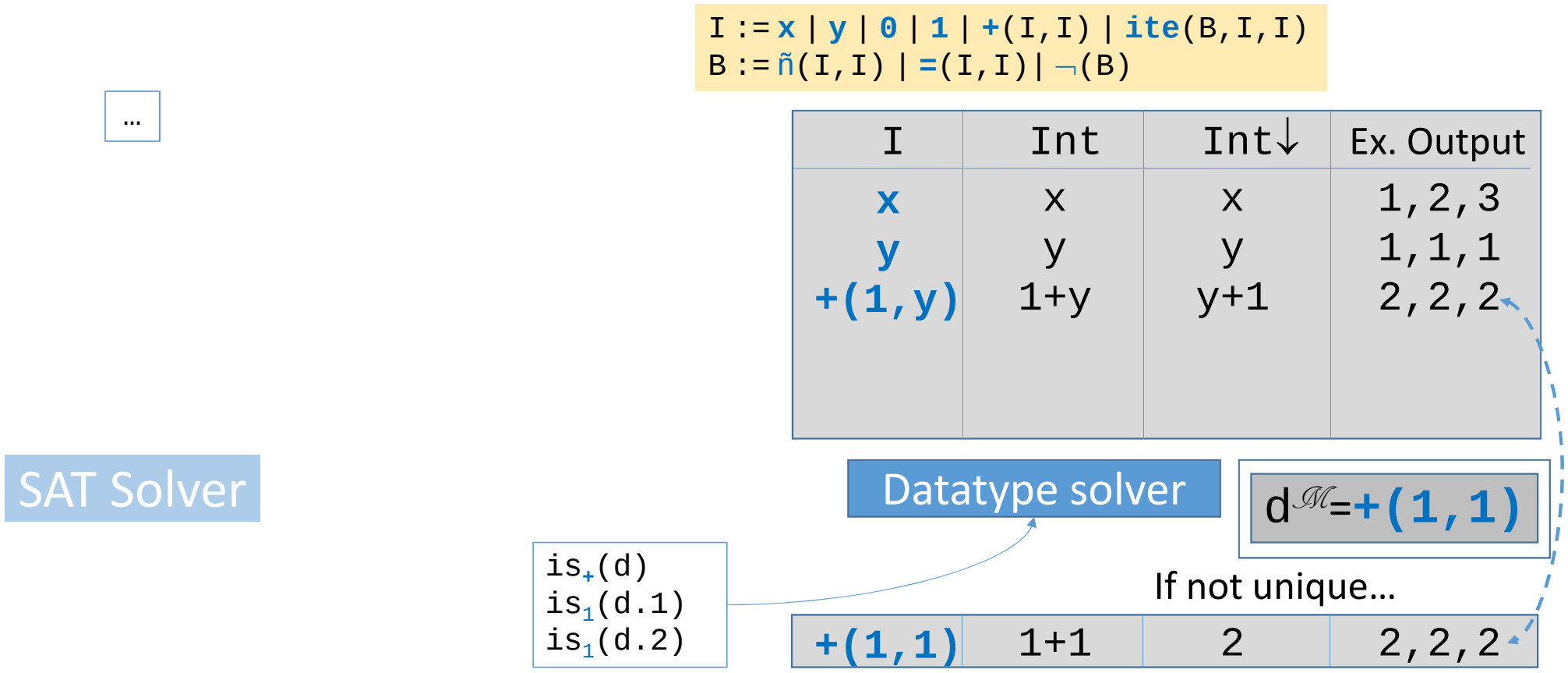
$d^{\mathcal{M}}$=**+(1,1)**

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

…and example outputs

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

Enumerative SyGuS in SMT for I/O Examples



Enumerative SyGuS in SMT for I/O Examples

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

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

- Return symmetry breaking clause
  ⇒ *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

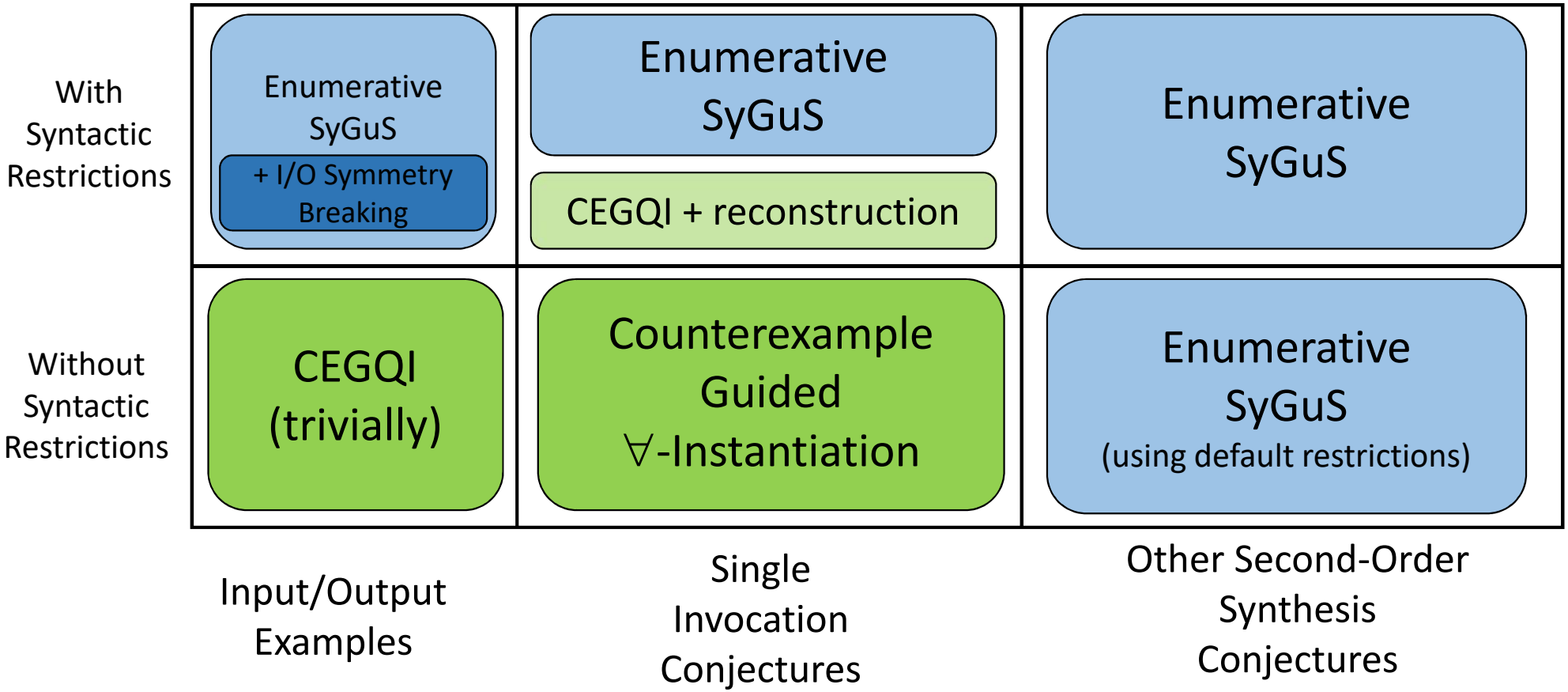
Datatype solver

$d^{\mathcal{M}}$=**+(1,1)**

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

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

Overview


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

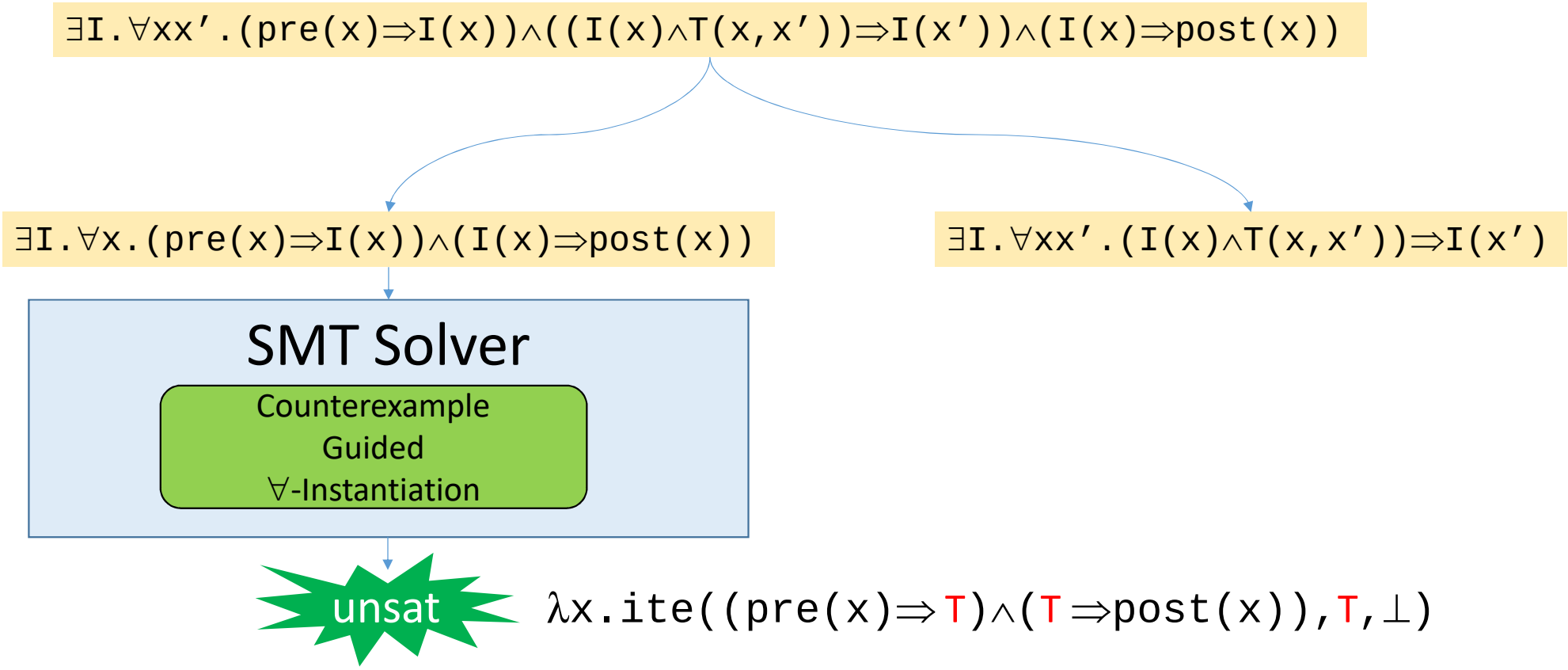
$\exists I.\forall x.(pre(x)\Rightarrow \mathbf{I(x)})\wedge(\mathbf{I(x)}\Rightarrow post(x))$

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

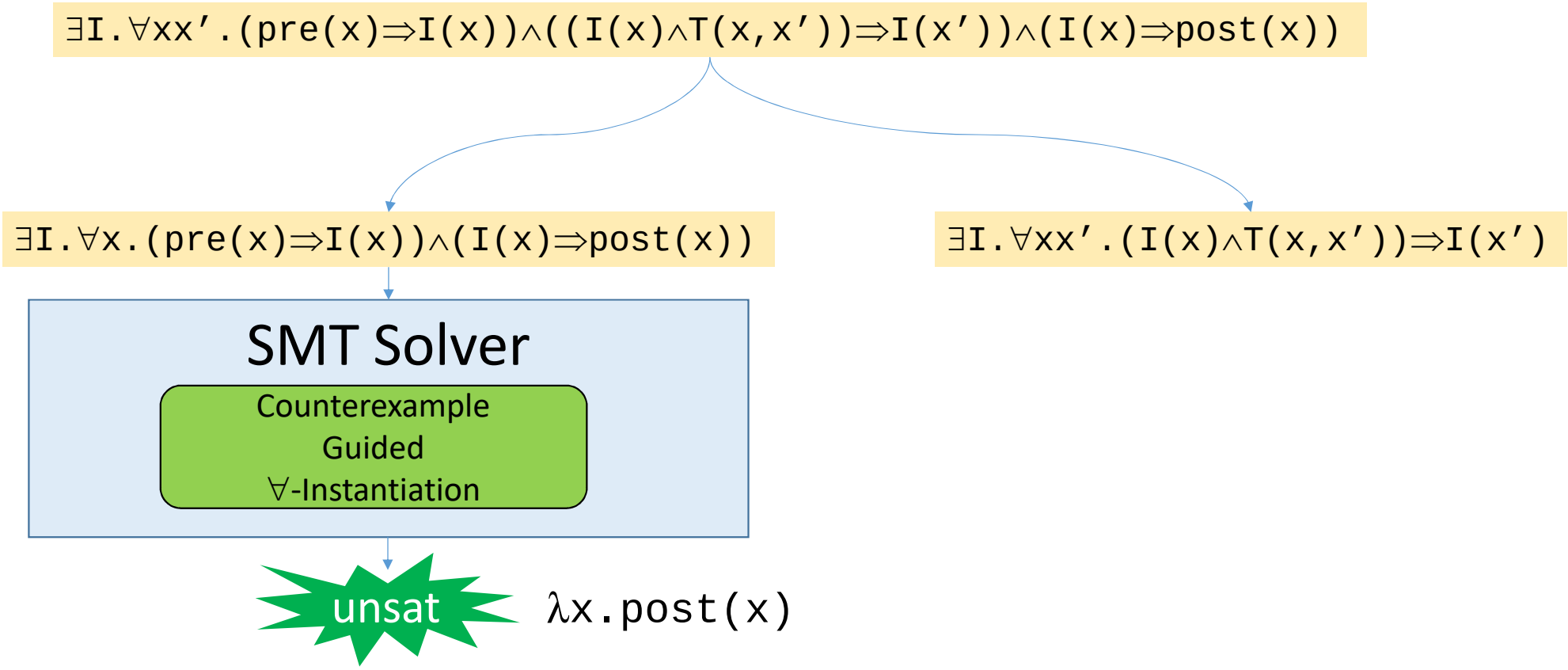
Single-invocation portion

Non-single-invocation portion

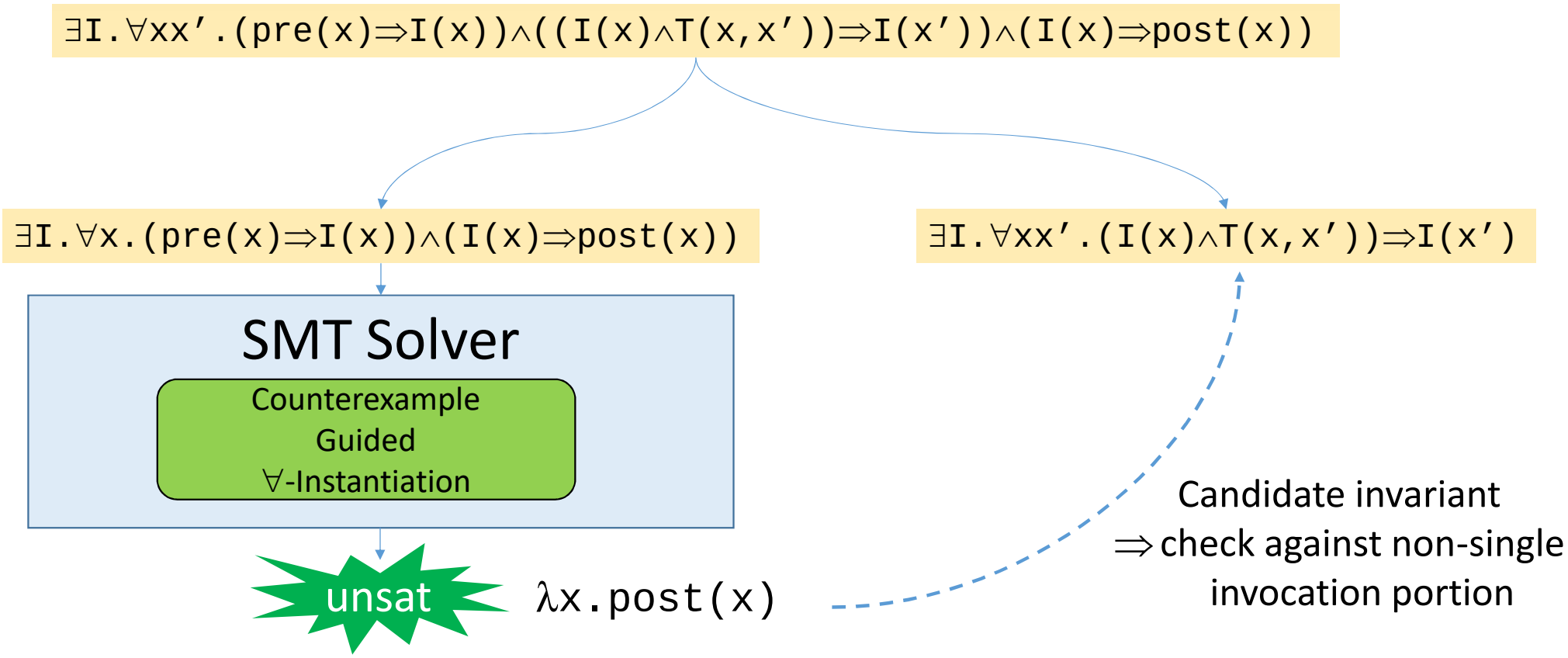
What if conjecture is *Partially Single Invocation?*



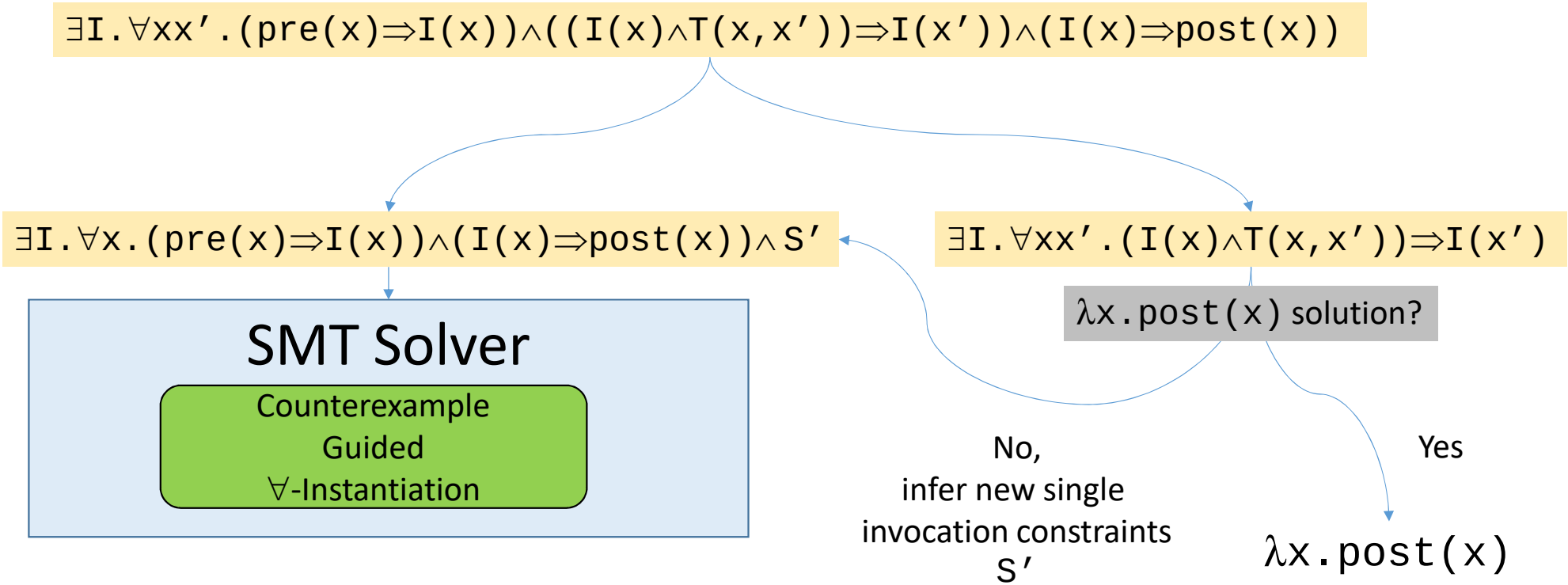
What if conjecture is *Partially Single Invocation?*



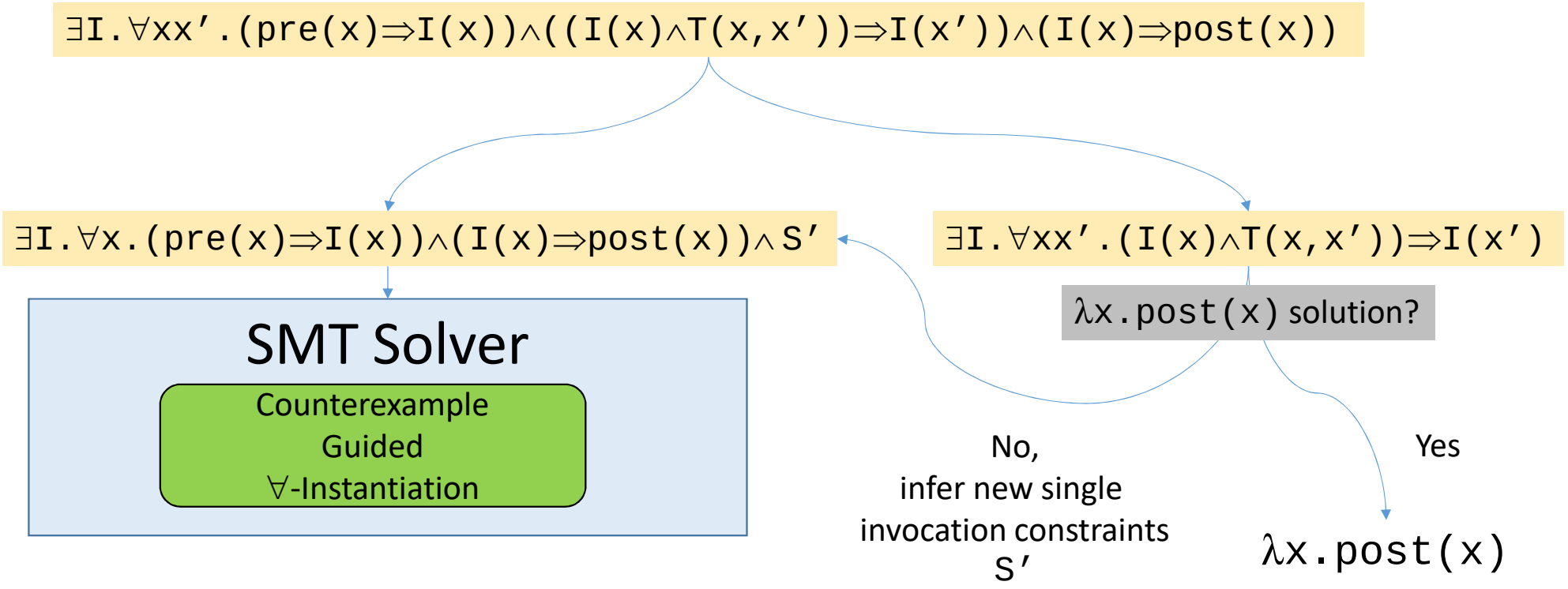
What if conjecture is *Partially Single Invocation?*



What if conjecture is *Partially Single Invocation?*

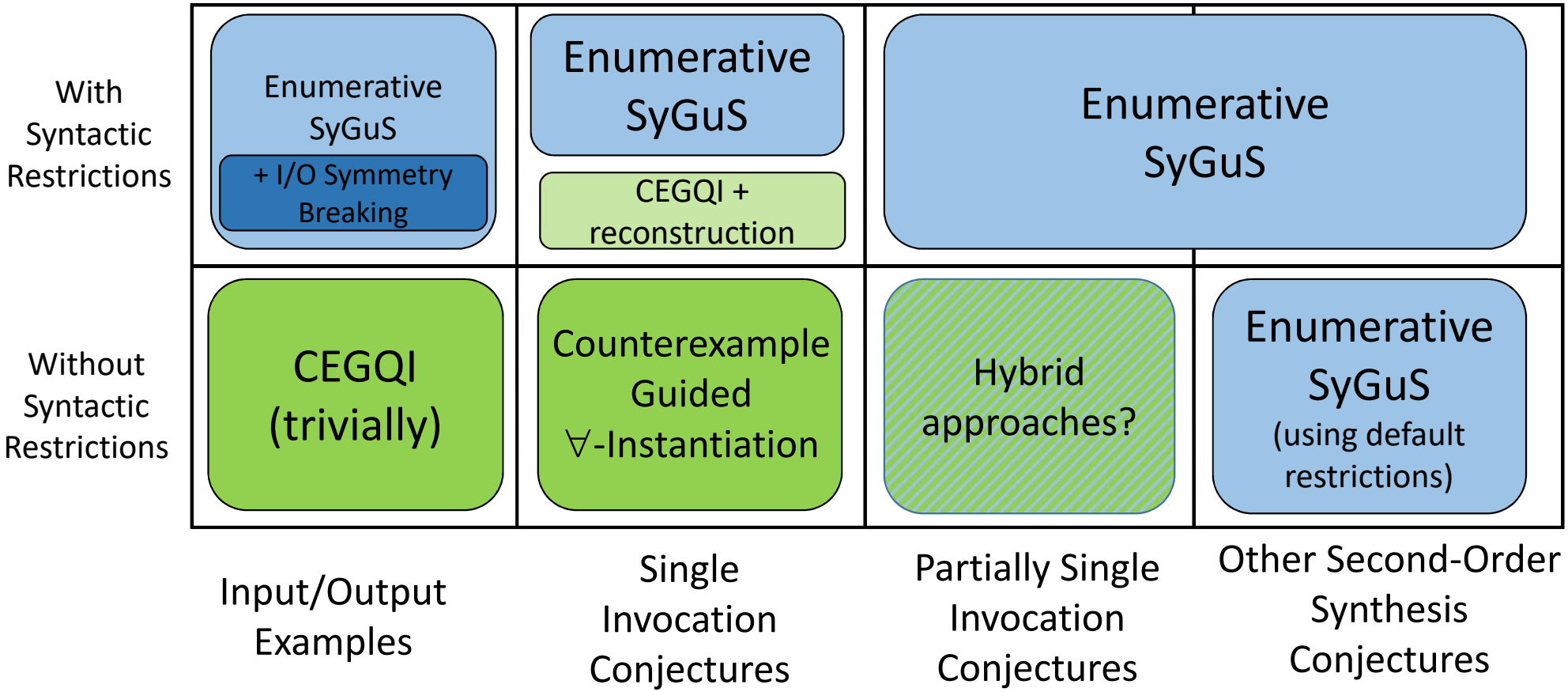


What if conjecture is *Partially Single Invocation?*

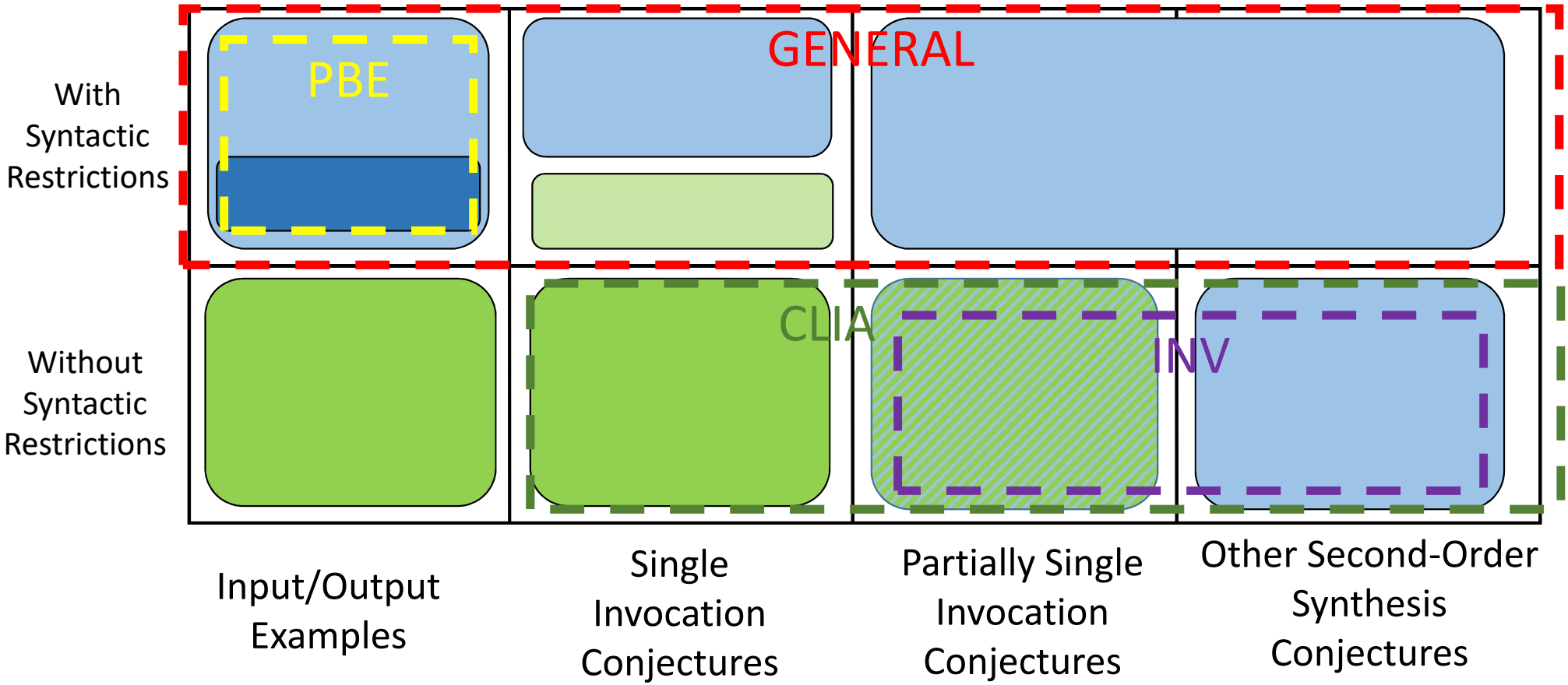


⇒Related to property-directed reachability (PDR) **[Bradley 2011]**

Overview


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

CVC4 for SyGuS Comp 2017:



Topics Not Covered:

- Automatically inferring when a conjecture $\Leftrightarrow$ a single-invocation one
- CEGQI solution minimization by proof analysis
  - Simpler proofs $\Rightarrow$ 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 $\forall$+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