Counterexample-Guided Quantifier Instantiation for Synthesis in SMT

Andrew Reynolds, Morgan Deters, Viktor Kuncak,

Cesare Tinelli, Clark Barrett

July 24, 2015

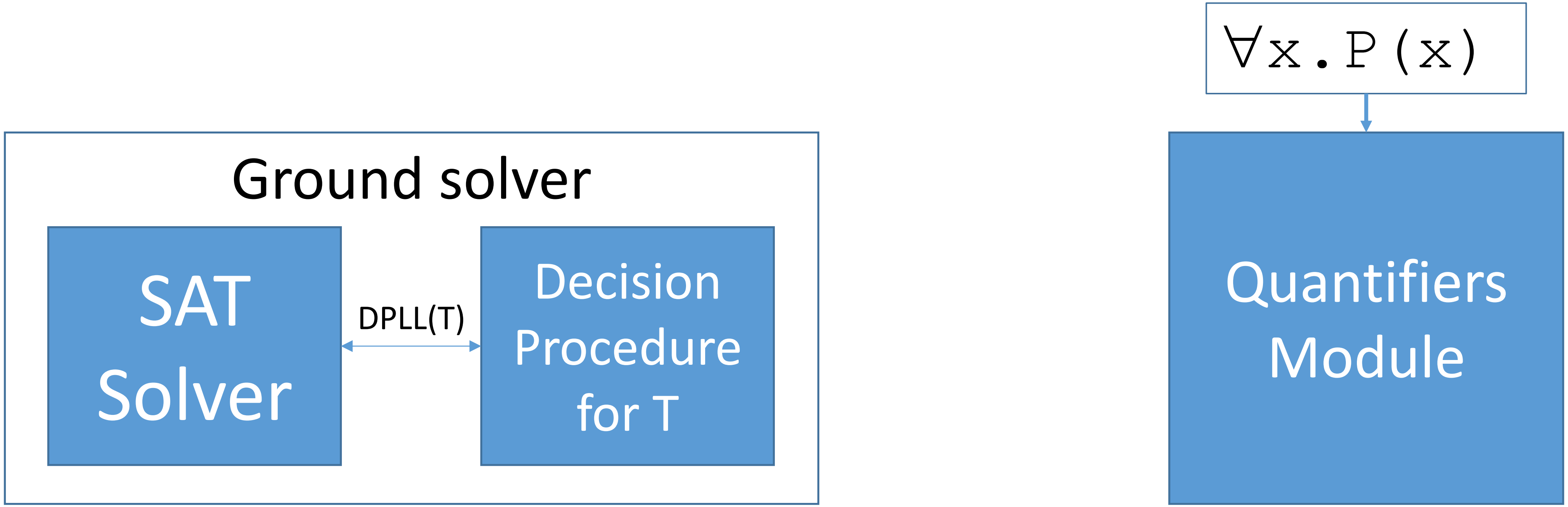
Overview

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

  There exists a function f such that for all x, $P(f, x)$

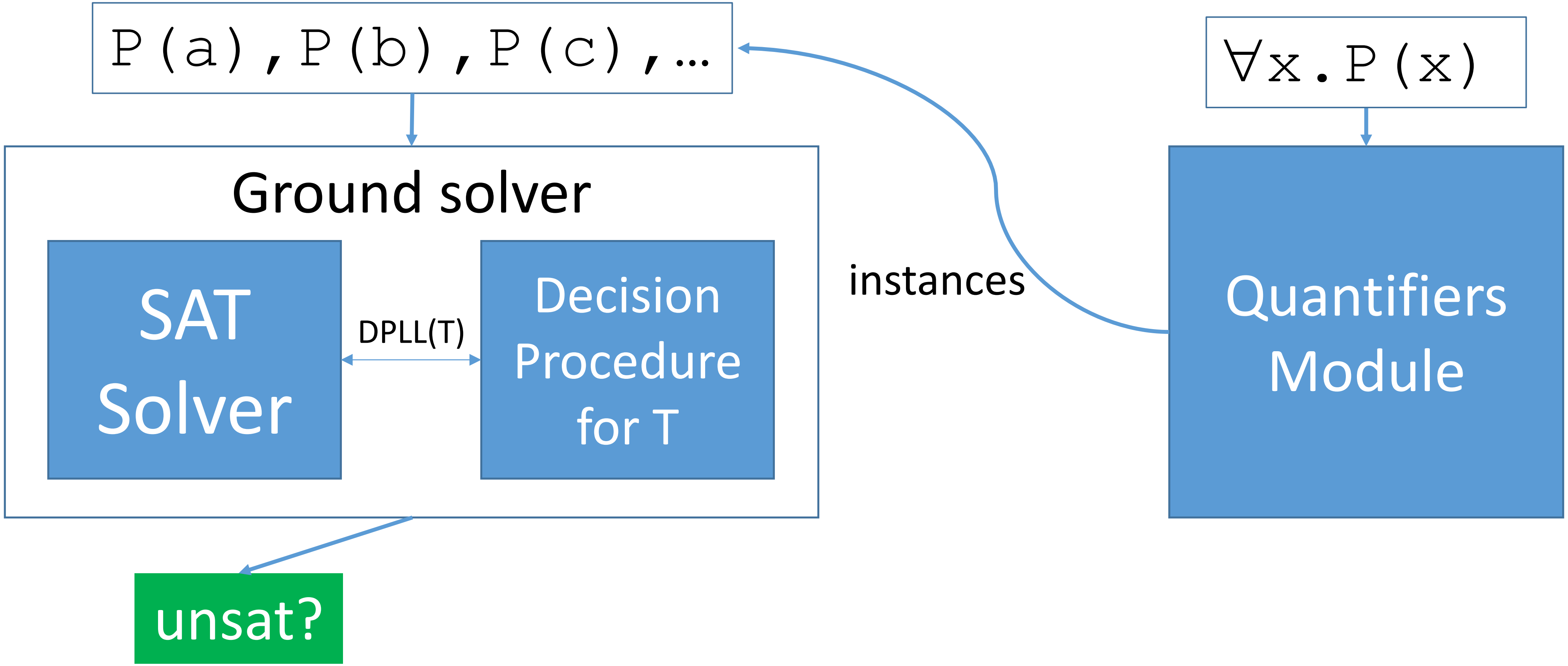
- Most existing approaches for synthesis
  - Rely on specialized solver that makes subcalls to an SMT Solver
- Approach for synthesis in this talk:
  - *Instrumented entirely inside SMT solver*

SMT Solver + Quantified Formulas



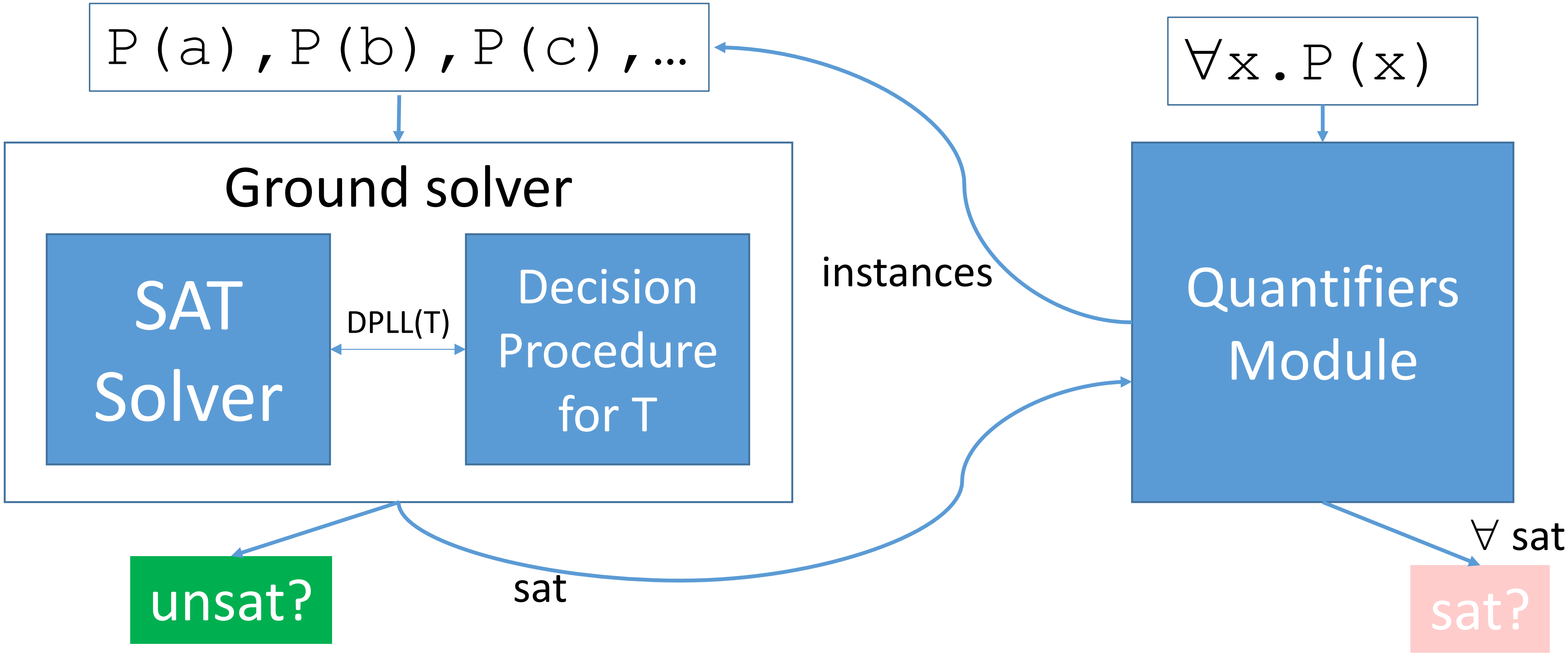
- SMT solver consists of:
  - Ground solver maintains a set of ground (variable-free) constraints
  - Quantifiers Module maintains a set of quantified formulas: $\forall x. P(x)$

SMT Solver + Quantified Formulas



- Goal : add instances of axioms until ground solver can answer “unsat”

SMT Solver + Quantified Formulas



- Generally, a sound but incomplete procedure
  - Difficult to answer sat (when have we added enough instances of ∀x.P(x)?)

Running Example : Max of Two Integers

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

- Specifies that f computes the maximum of integers x and y
- Solution:

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

How does an SMT solver handle Max example?

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

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

- Direct approach:
  - Treat `f` as an *uninterpreted function*
  - Succeed if SMT solver can find correct interpretation of `f`, answer sat
    - ⇒*This is challenging*
      - How does the solver know the right interpretation for `f` to pick?

Refutation-Based Synthesis

$$\exists \mathtt{f}.\forall \mathbf{x}.\mathtt{P}(\mathtt{f},\mathbf{x})$$

- Since it is challenging to answer "sat" when ∀ are present,

  ⇒ Can we instead use a *refutation-based* approach for synthesis?

Refutation-Based Synthesis

$$\neg \exists \mathtt{f}.\forall \mathbf{x}.\mathtt{P}(\mathtt{f},\mathbf{x})$$

- What if we negate the synthesis conjecture?
- If we are in a *satisfaction-complete* theory T (e.g. LIA, BV):
  - *F is T-satisfiable if and only if $\neg F$ is T-unsatisfiable*

$\Rightarrow$ Will suffice for us to show the above formula is unsat

Challenge: Second-Order Quantification

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

negate

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

- Challenge: negation introduces universal $\forall$ over function f
  - No SMT solvers directly support second-order quantification

Challenge: Second-Order Quantification

$$\neg \exists \texttt{f}. \forall \mathbf{x}. \texttt{P}(\texttt{f}, \mathbf{x})$$

negate

$$\forall \texttt{f}. \exists \mathbf{x}. \neg \texttt{P}(\texttt{f}, \mathbf{x})$$

- Challenge: negation introduces universal $\forall$ over function `f`
  - No SMT solvers directly support second-order quantification
- However, we can avoid this quantification using two approaches:
  1. When property `P` is single invocation for `f`
  2. When `f` is given syntactic restrictions

Single Invocation Properties

$$\forall f.\ \exists xy.\ (f(x,y)<x \lor f(x,y)<y \lor (f(x,y)\neq x \land f(x,y)\neq y))$$

Single Invocation Properties

$$\forall f. \exists xy. (f(x,y)<x \lor f(x,y)<y \lor (f(x,y)\neq x \land f(x,y)\neq y))$$

- *Single invocation* properties
  - Are properties such that:
    - All occurrences of `f` are of a particular form, e.g. `f(x,y)` above
  - Are a common class of properties useful for:
    - Software Synthesis (post-conditions describing the result of a function)

Single Invocation Properties

$$\forall f. \exists xy. (f(x,y) < x \lor f(x,y) < y \lor (f(x,y) \neq x \land f(x,y) \neq y))$$

Push quantification downwards

$$\exists xy. \forall g. (g < x \lor g < y \lor (g \neq x \land g \neq y))$$

- Occurrences of $f(x,y)$ are replaced with integer variable g
- Resulting formula is equisatisfiable, and first-order

Single Invocation Properties

$$\forall f.\exists xy.(f(x,y)<x \vee f(x,y)<y \vee (f(x,y)\neq x \wedge f(x,y)\neq y))$$

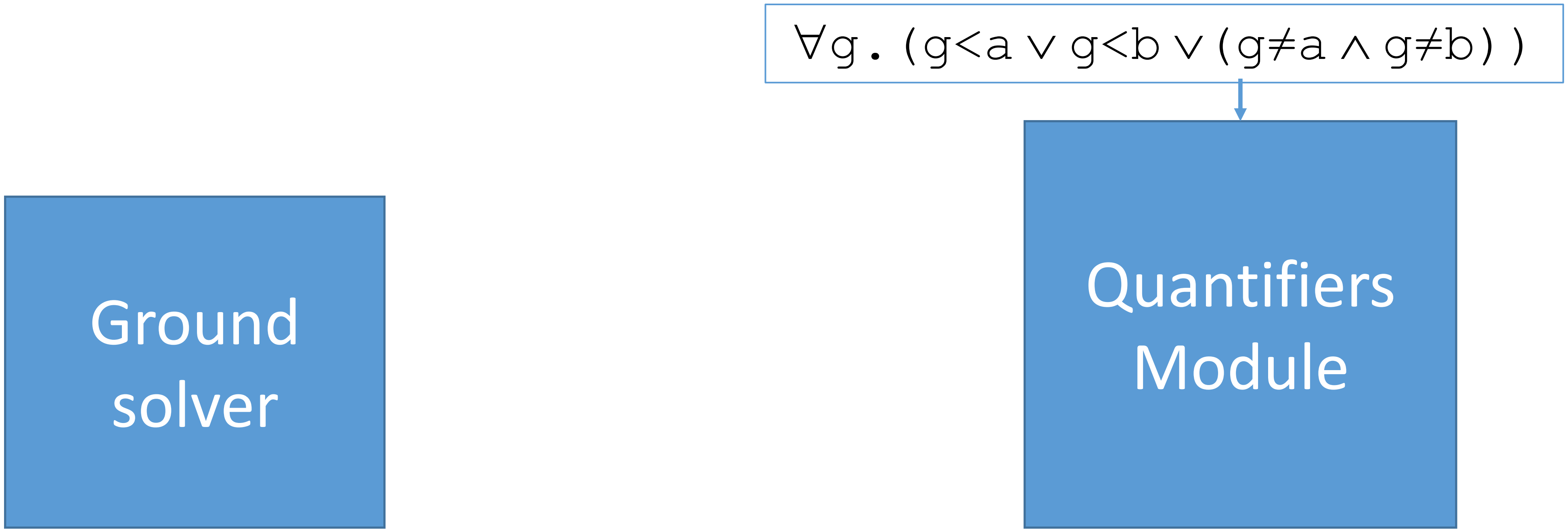
Push quantification downwards

$$\exists xy.\forall g.(g<x \vee g<y \vee (g\neq x \wedge g\neq y))$$

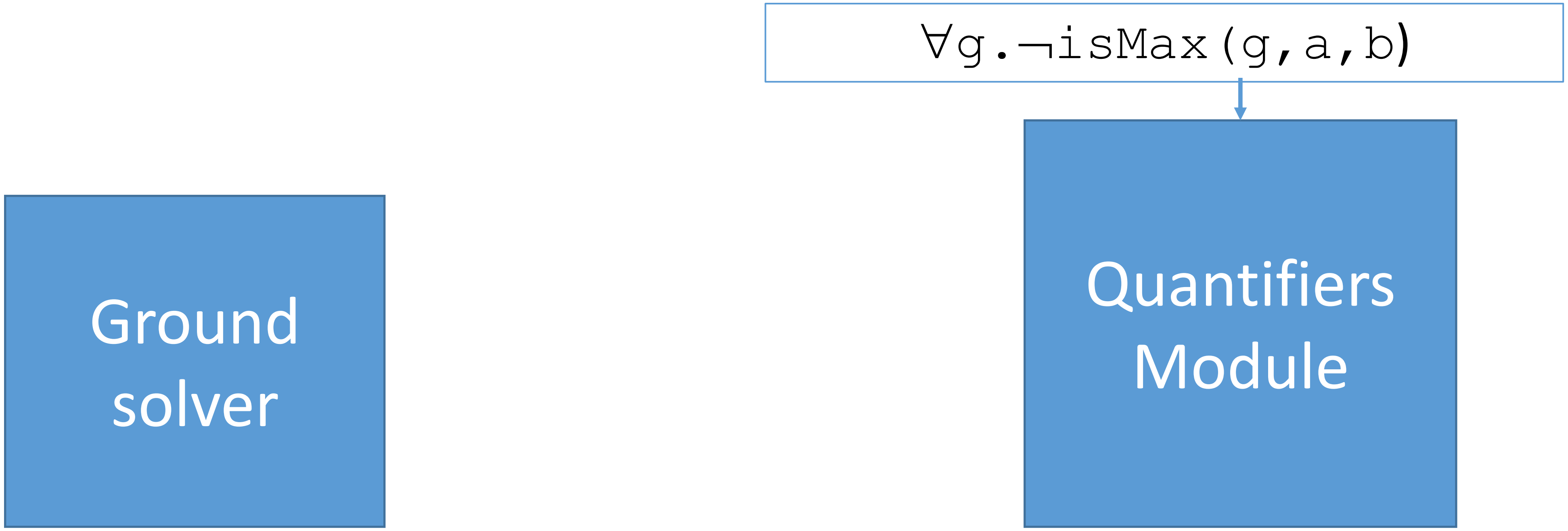
Skolemize, for fresh a and b

$$\forall g.(g<a \vee g<b \vee (g\neq a \wedge g\neq b))$$

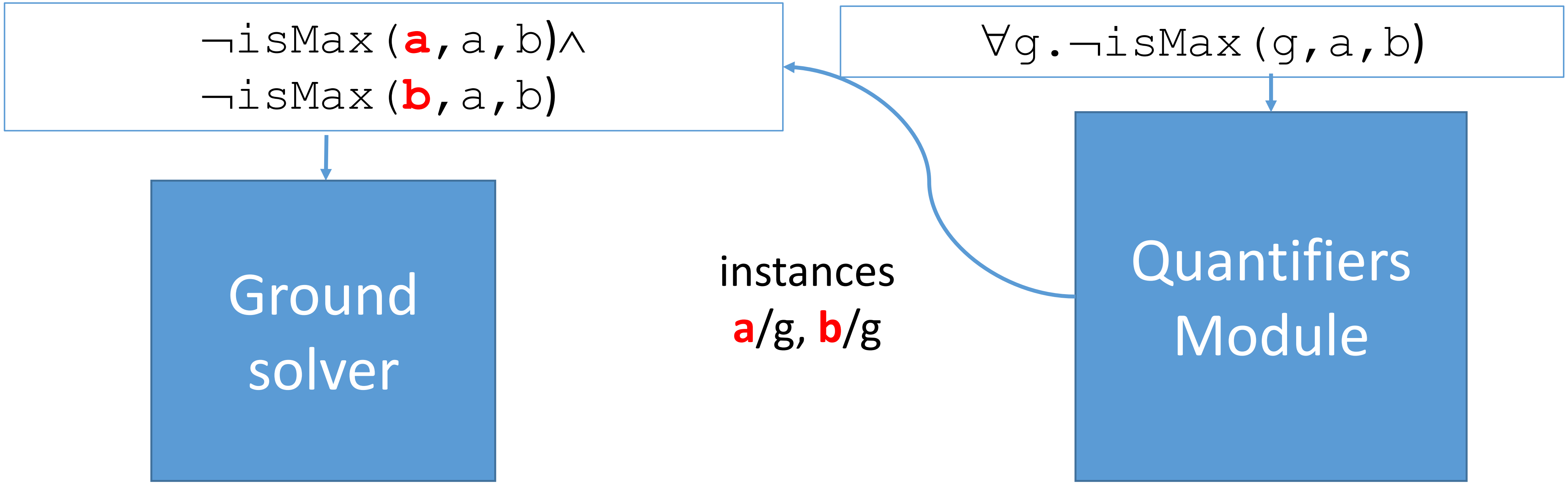
Solving Max Example



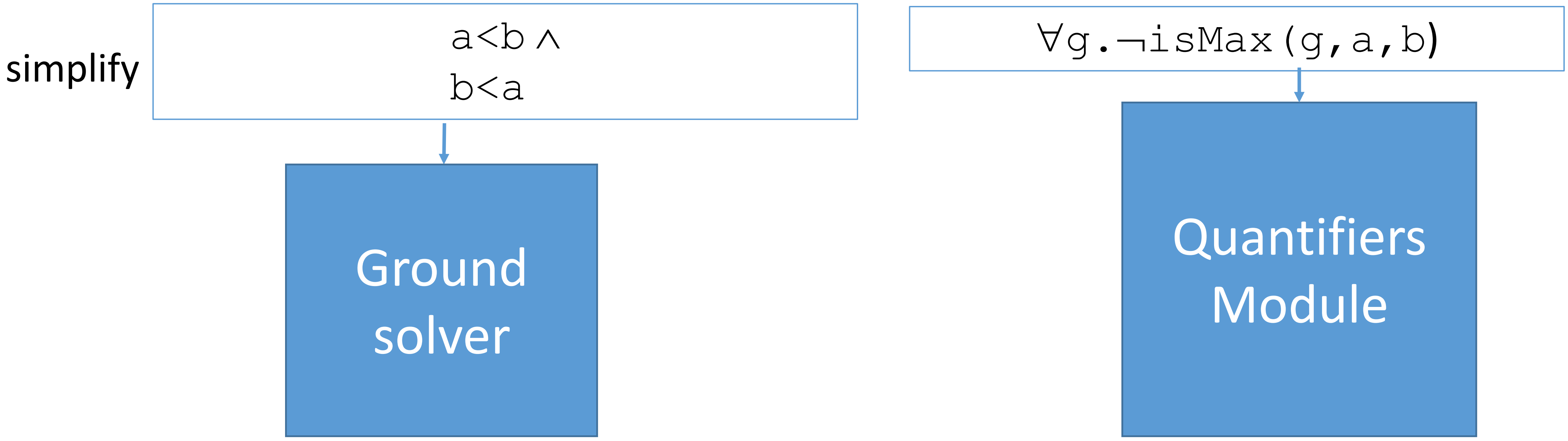
Solving Max Example



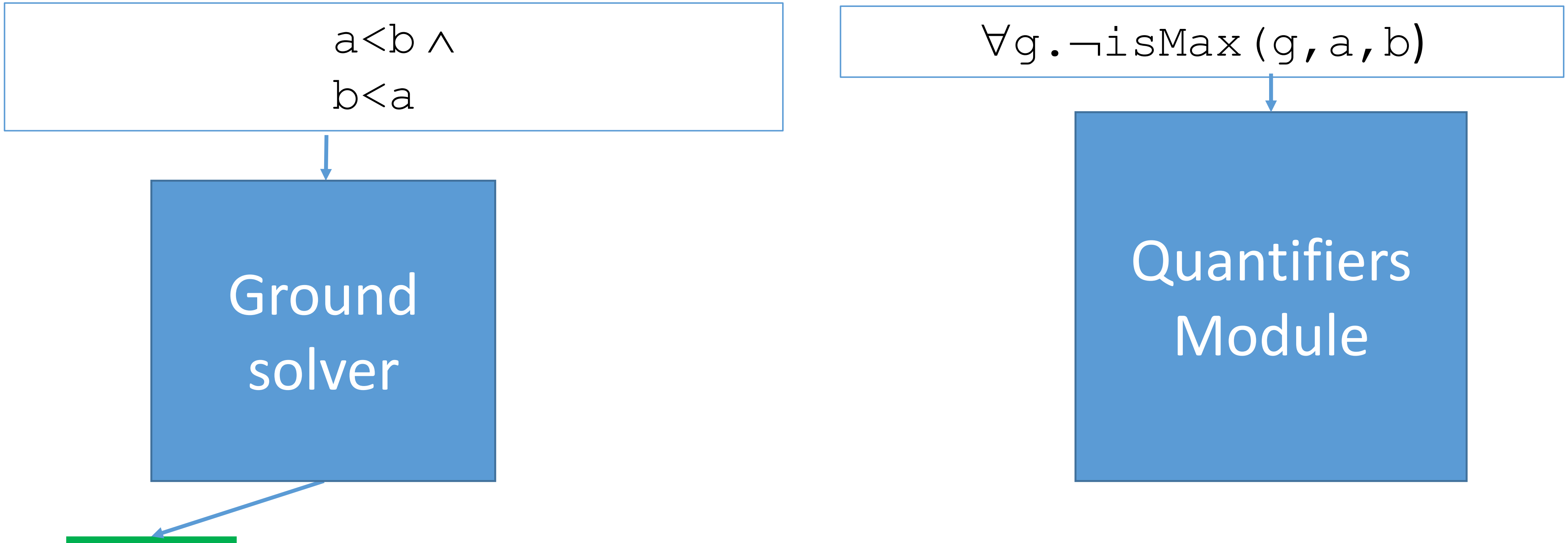
Solving Max Example



Solving Max Example



Solving Max Example

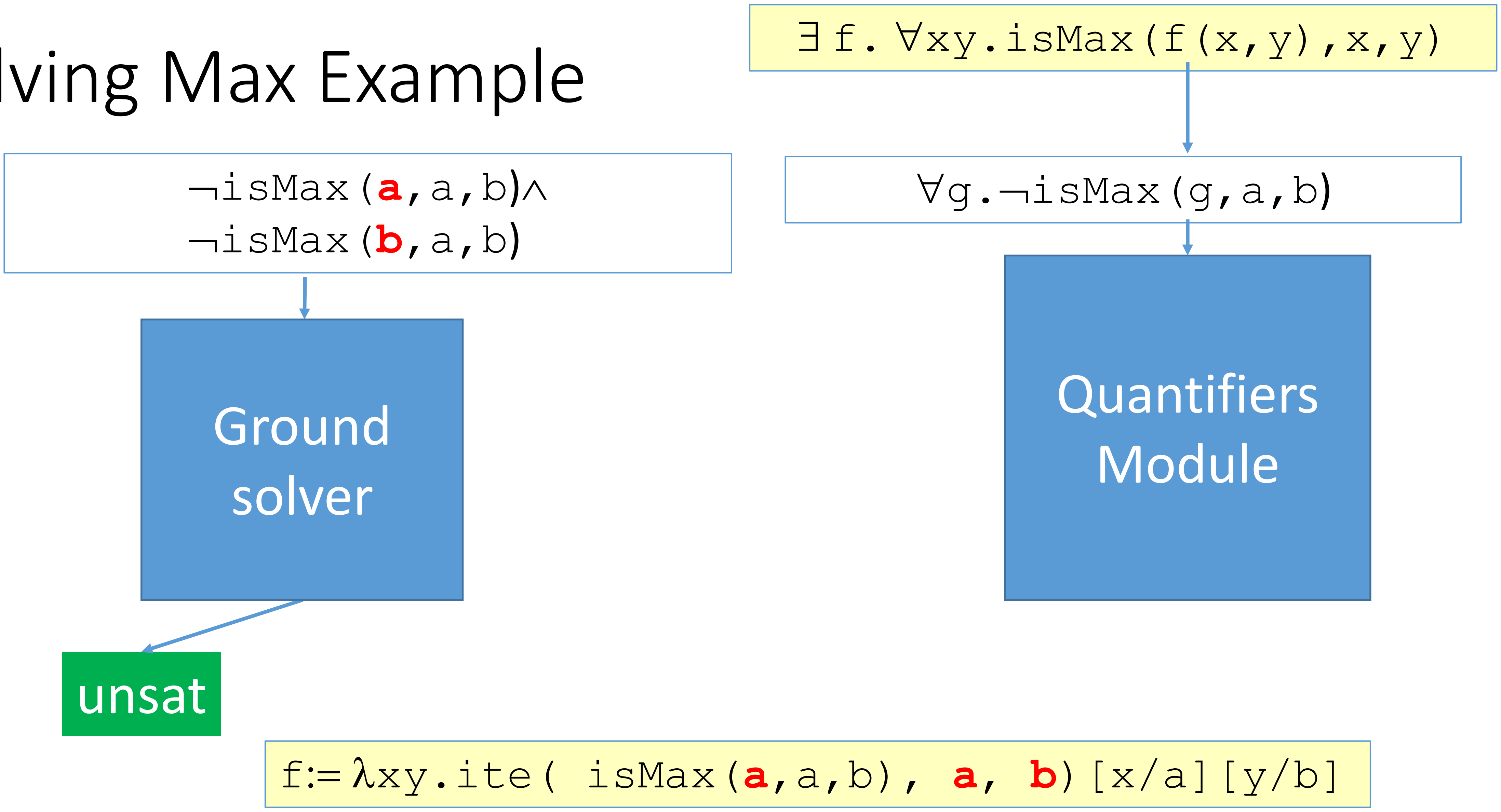


unsat ⇒ ∀g.¬isMax(g,a,b) is unsatisfable

by instances a/g, b/g,

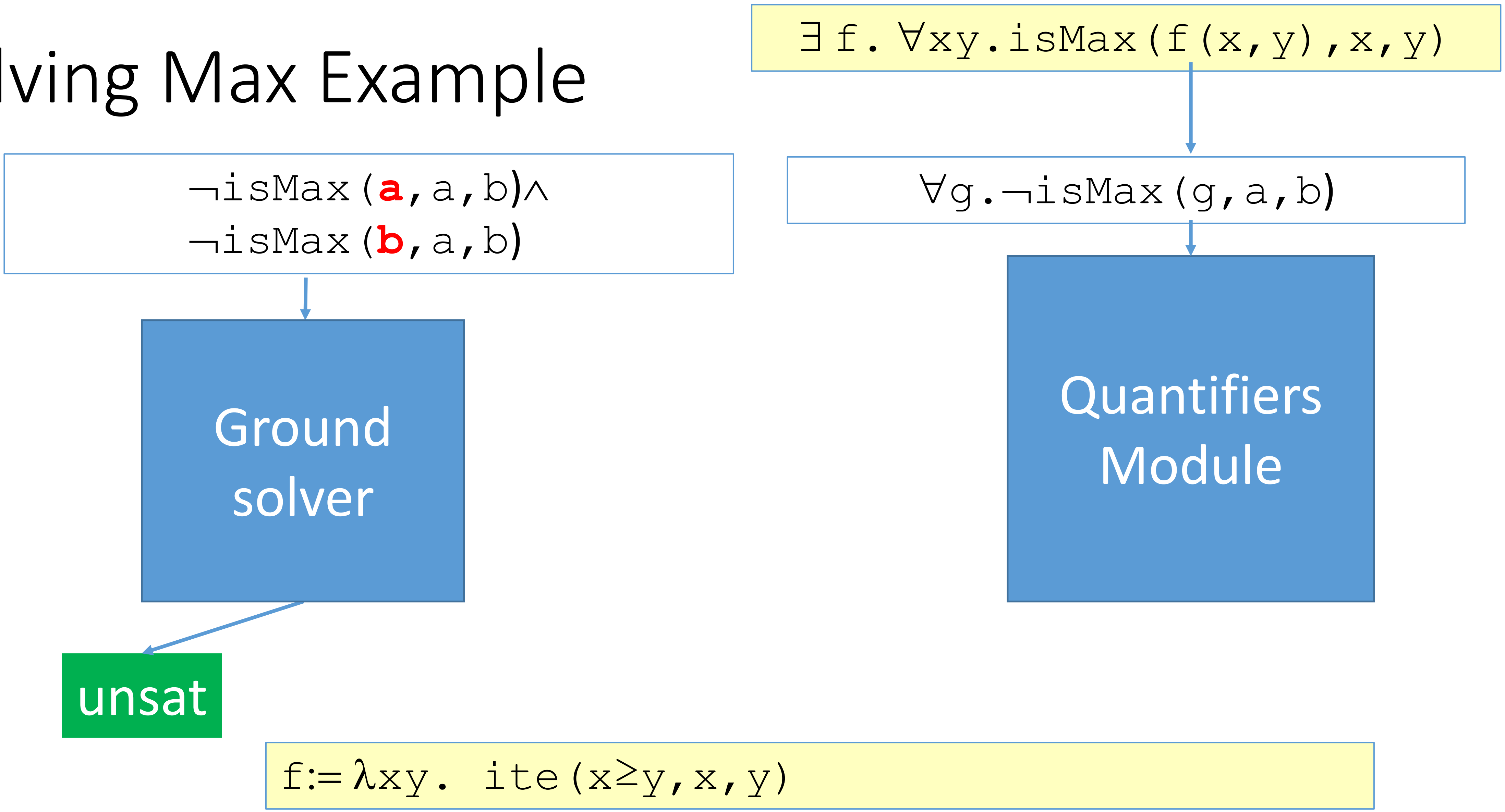
implies original synthesis conjecture has a solution

Solving Max Example



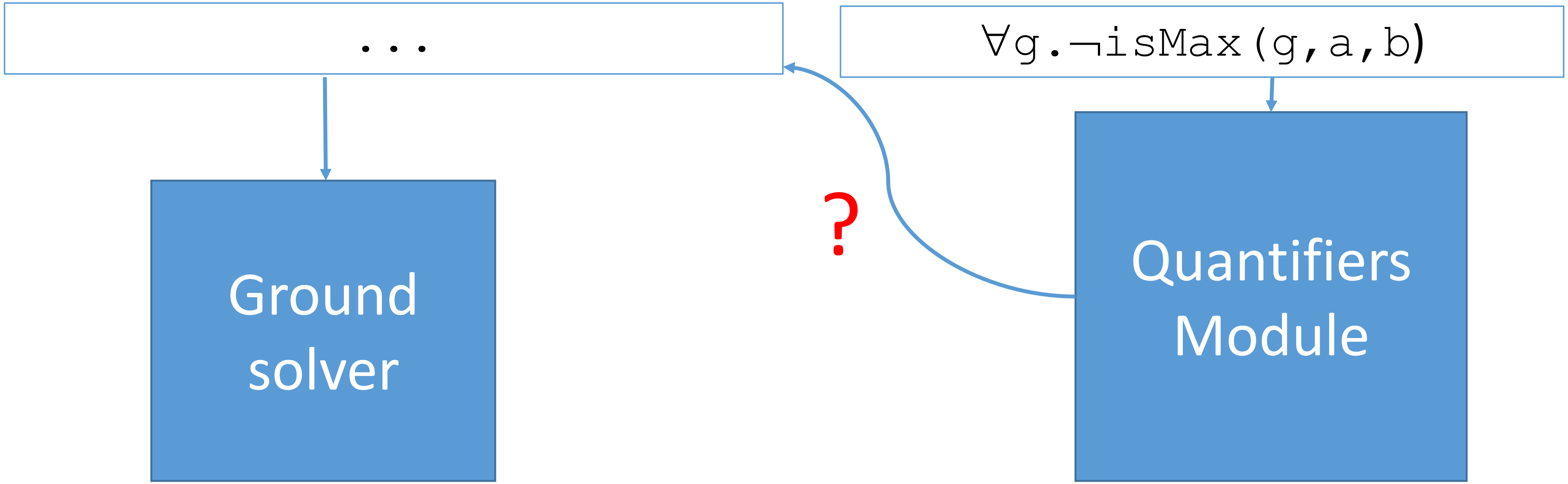
⇒Solution can be extracted from unsatisfiable core of instantiations a/g, b/g

Solving Max Example

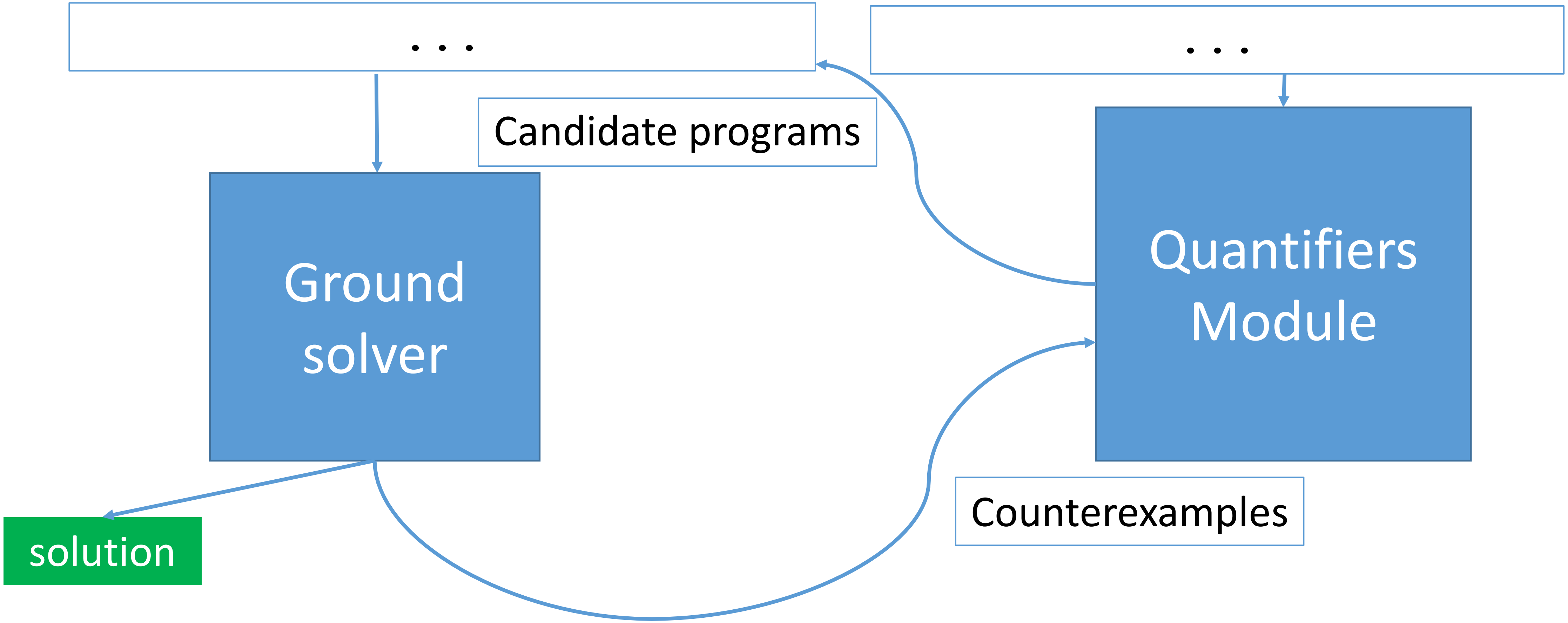


⇒ Desired function, after simplification

How do we Choose Relevant Instances?



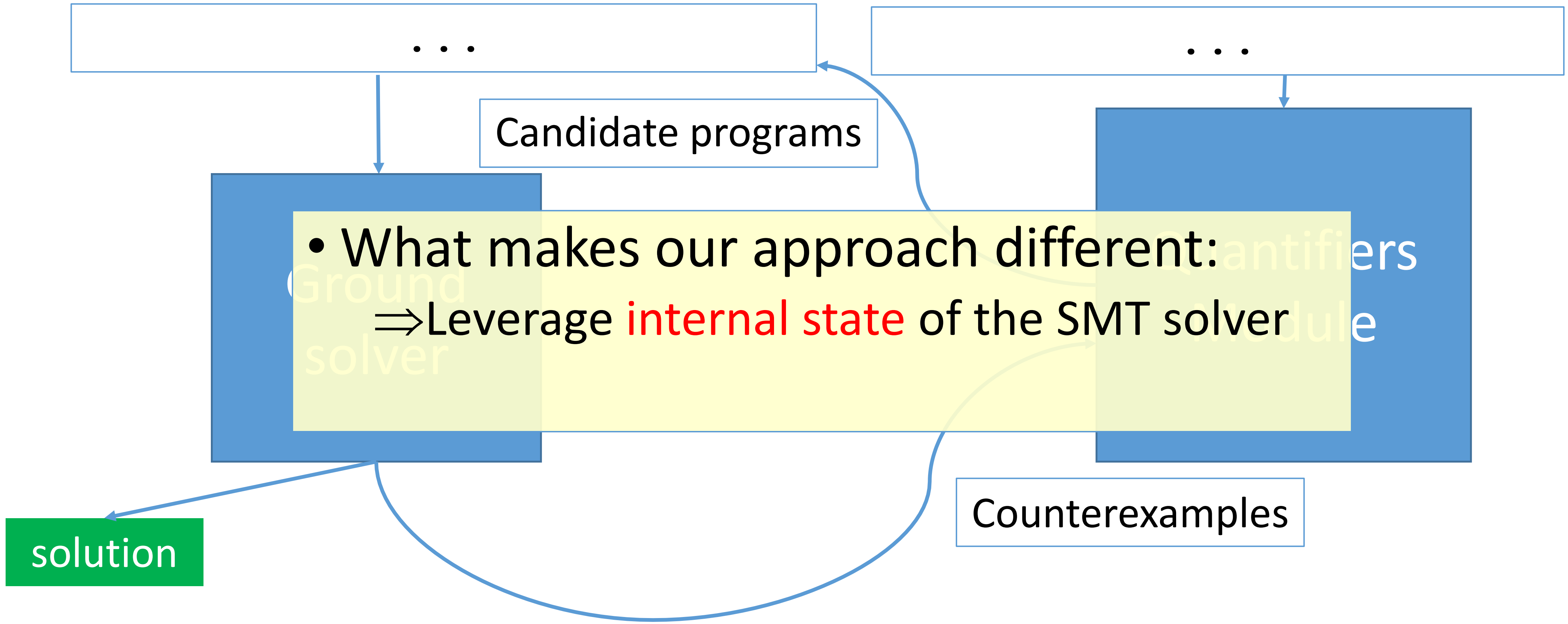
Counterexample-Guided Quantifier Instantiation



- Instances chosen *counterexample-guided quantifier instantiation*

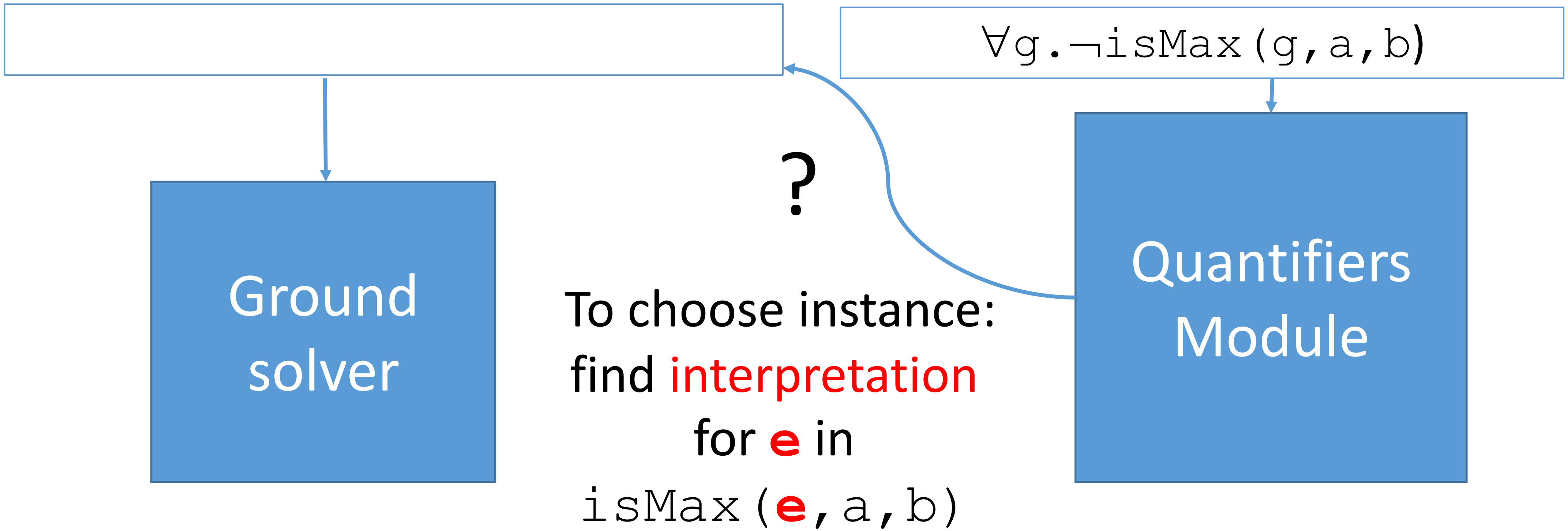
  ⇒ Follows counterexample-guided inductive synthesis (CEGIS) approach

Counterexample-Guided Quantifier Instantiation

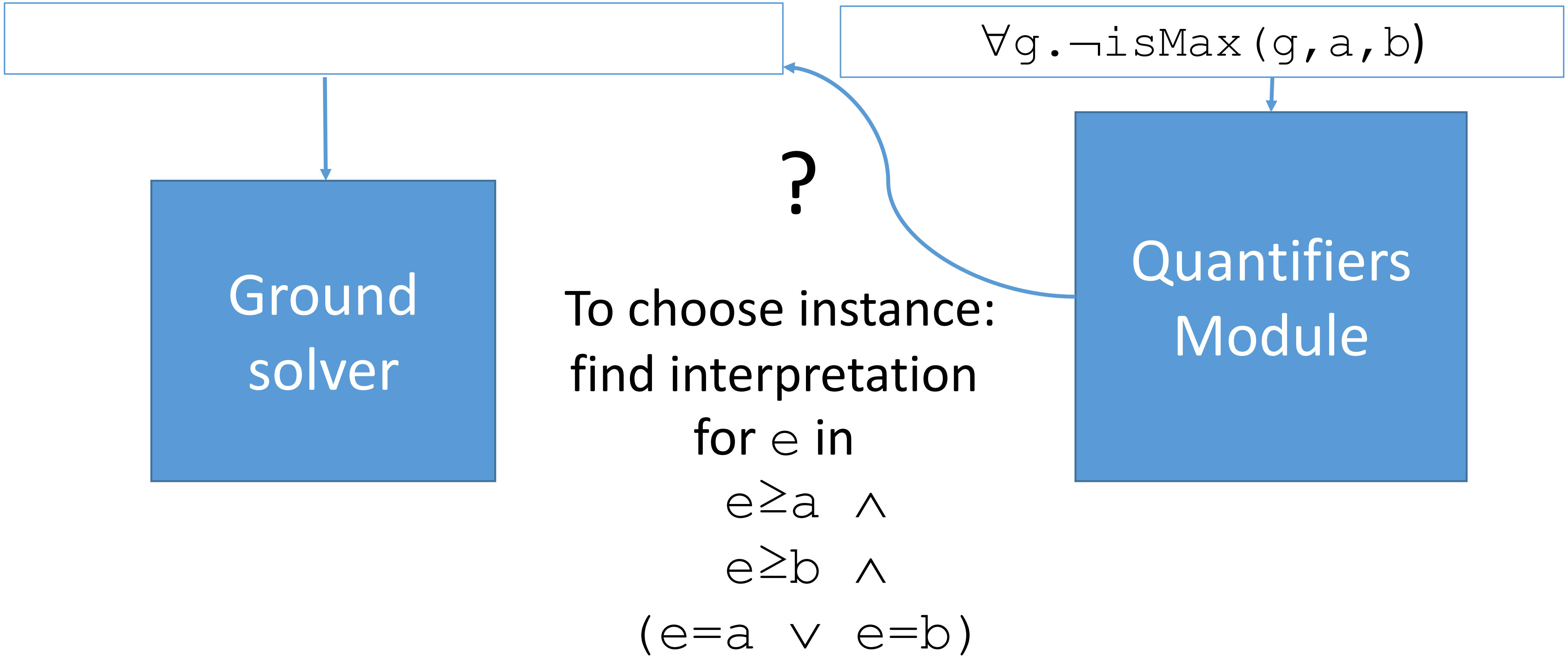


- What makes our approach different:
  - ⇒Leverage internal state of the SMT solver

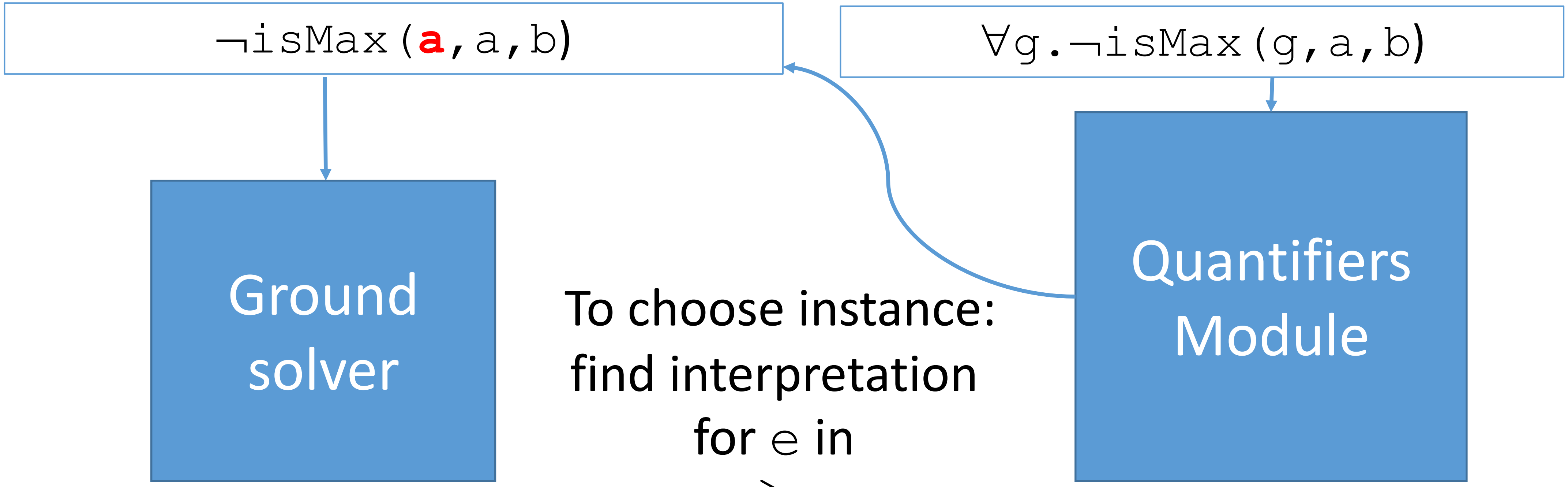
Counterexample-Guided Quantifier Instantiation



Counterexample-Guided Quantifier Instantiation



Counterexample-Guided Quantifier Instantiation

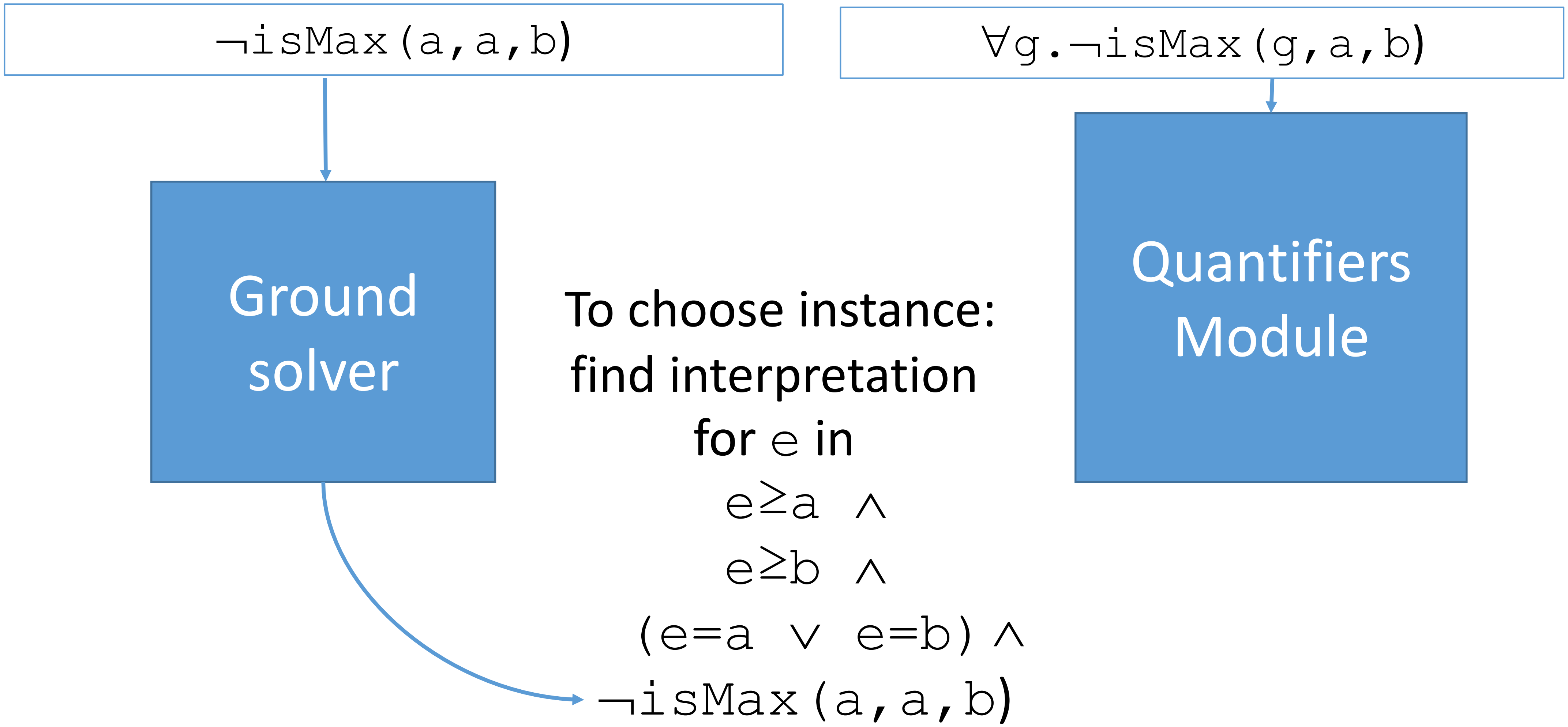


To choose instance:
find interpretation
for e in

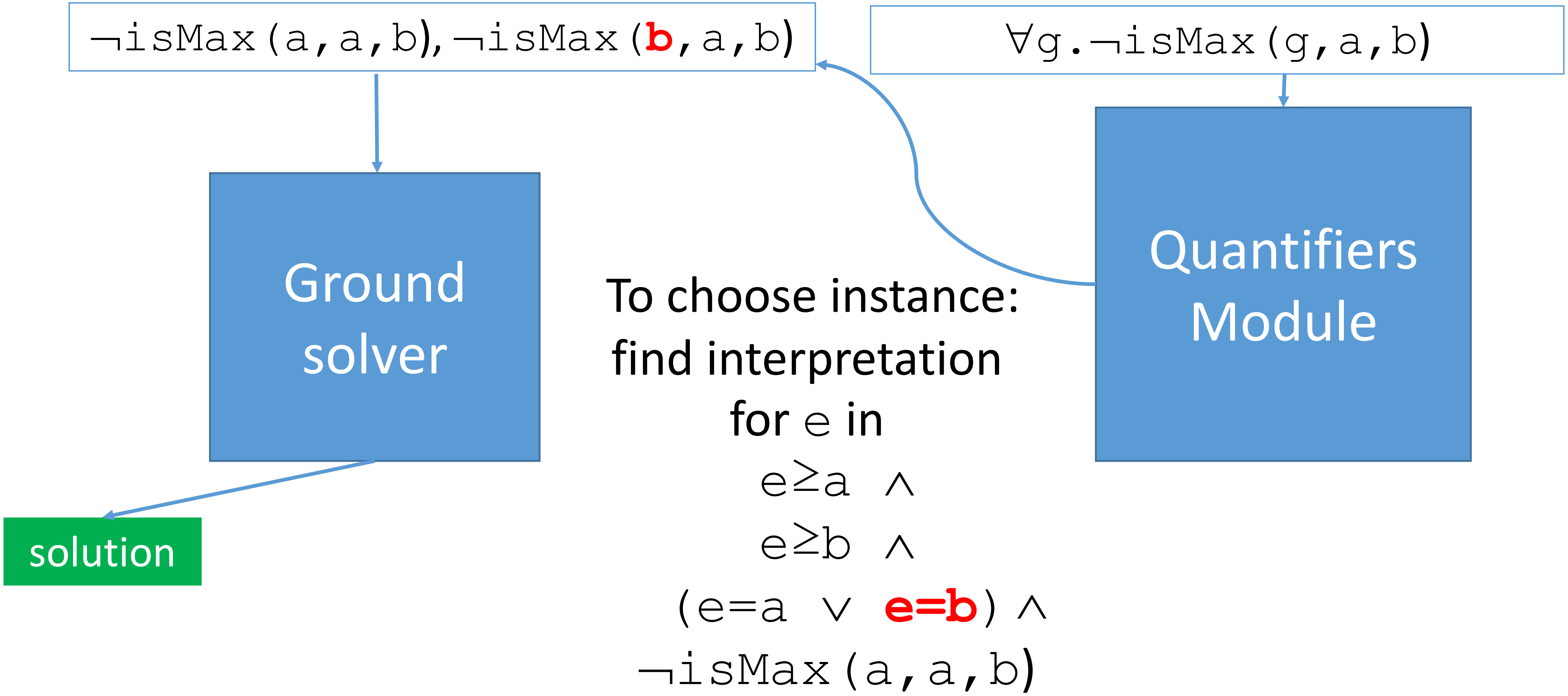
$$e \geq a \;\wedge$$
$$e \geq b \;\wedge$$
$$(\mathbf{e=a} \;\vee\; e=b)$$

⇒ e.g. based on the equivalence class of e

Counterexample-Guided Quantifier Instantiation



Counterexample-Guided Quantifier Instantiation



Non-Single Invocation Properties

- What if property is *not single invocation*?

$$\exists c. \forall xy. c(x,y)=c(y,x)$$

e.g. `c` is commutative

Non-Single Invocation Properties

- What if property is *not single invocation*?

$$\exists c.\ \forall xy.\, c(x,y)=c(y,x)$$

Negate

$$\forall c.\ \exists xy.\, c(x,y)\neq c(y,x)$$

Non-Single Invocation Properties

- What if property is *not single invocation*?

$$\exists c.\ \forall xy.c(x,y)=c(y,x)$$

Negate

$$\forall c.\ \exists xy.c(x,y)\neq c(y,x)$$

Model domain of c as algebraic datatype D

D := zero | one | x1 | x2 | plus(D1,D2)

$\forall$**d:D**$.\ \exists xy.\mathrm{eval}(d,x,y)\neq \mathrm{eval}(d,y,x)\wedge$

$\forall xy.\mathrm{eval}(\mathrm{zero},x,y)=0\ \wedge\ \forall xy.\mathrm{eval}(\mathrm{one},x,y)=1\ \wedge$

$\forall xy.\mathrm{eval}(x1,x,y)=x\ \wedge\ \forall xy.\mathrm{eval}(x2,x,y)=y\ \wedge$

$\forall d_1 d_2 xy.\mathrm{eval}(\mathrm{plus}(d_1,d_2),x,y)=\mathrm{eval}(d_1,x,y)+\mathrm{eval}(d_2,x,y)$

Single Invocation + Syntactic Restrictions

- What if property is single invocation, but has *syntactic restrictions*?

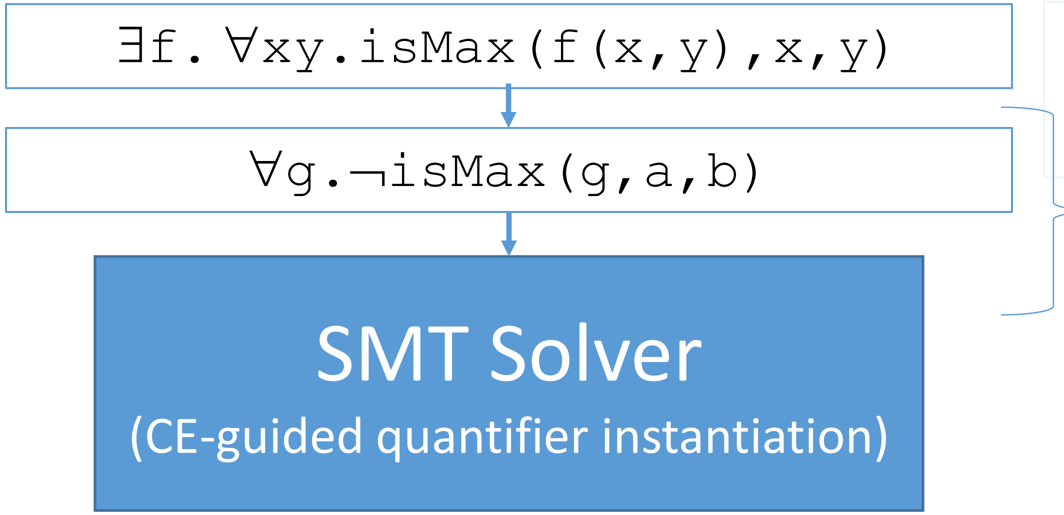
```
∃f. ∀xy.isMax(f(x,y),x,y)
```

Max example
(single invocation)

```
D := 0 | 1 | x1 | x2 | ite(B1,D1,D2)
B := ≤(D1,D2) | =(D1,D2) | ∧(B1,B2)
```

Syntactic restrictions for `f`

Single Invocation + Syntactic Restrictions




```
D := 0 | 1 | x1 | x2 | ite(B1,D1,D2)
B := ≤(D1,D2) | =(D1,D2) | ∧(B1,B2)
```

Convert to first order based on transformation for single invocation properties

Single Invocation + Syntactic Restrictions

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

↓

$\forall g.\neg\text{isMax}(g,a,b)$

↓

SMT Solver

(CE-guided quantifier instantiation)

↓

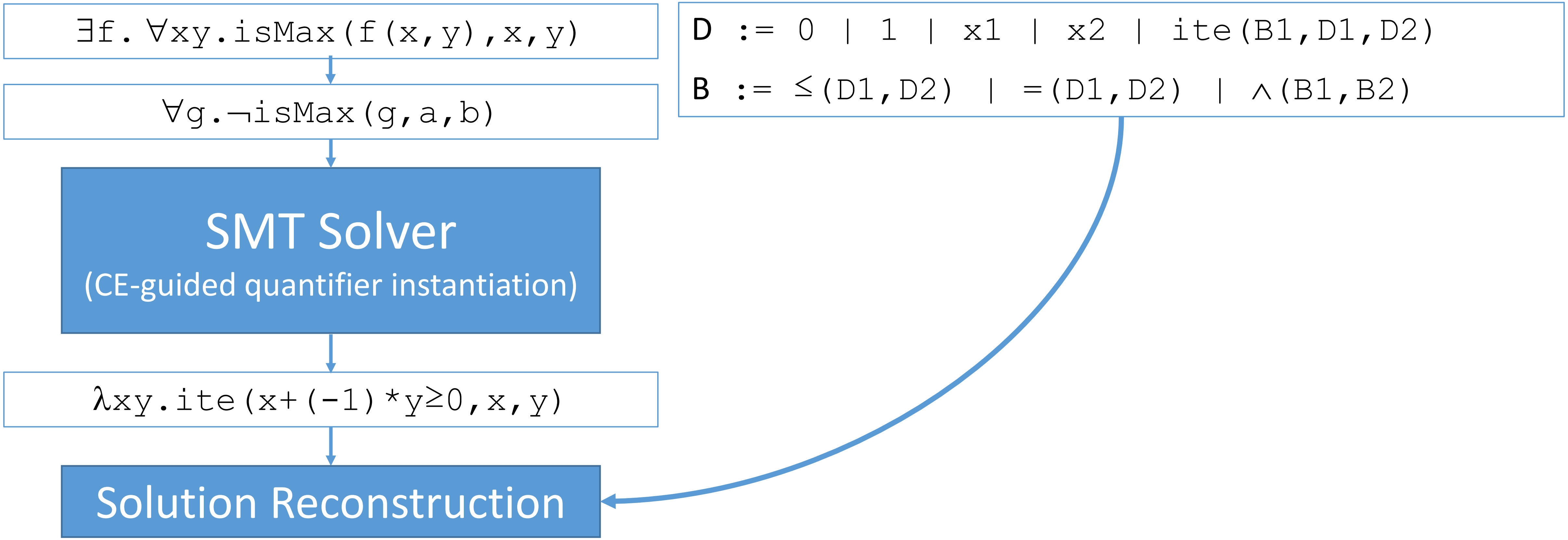
$\lambda xy.\text{ite}(x+(-1)*y\geq 0,x,y)$

Solve, while ignoring syntactic restrictions

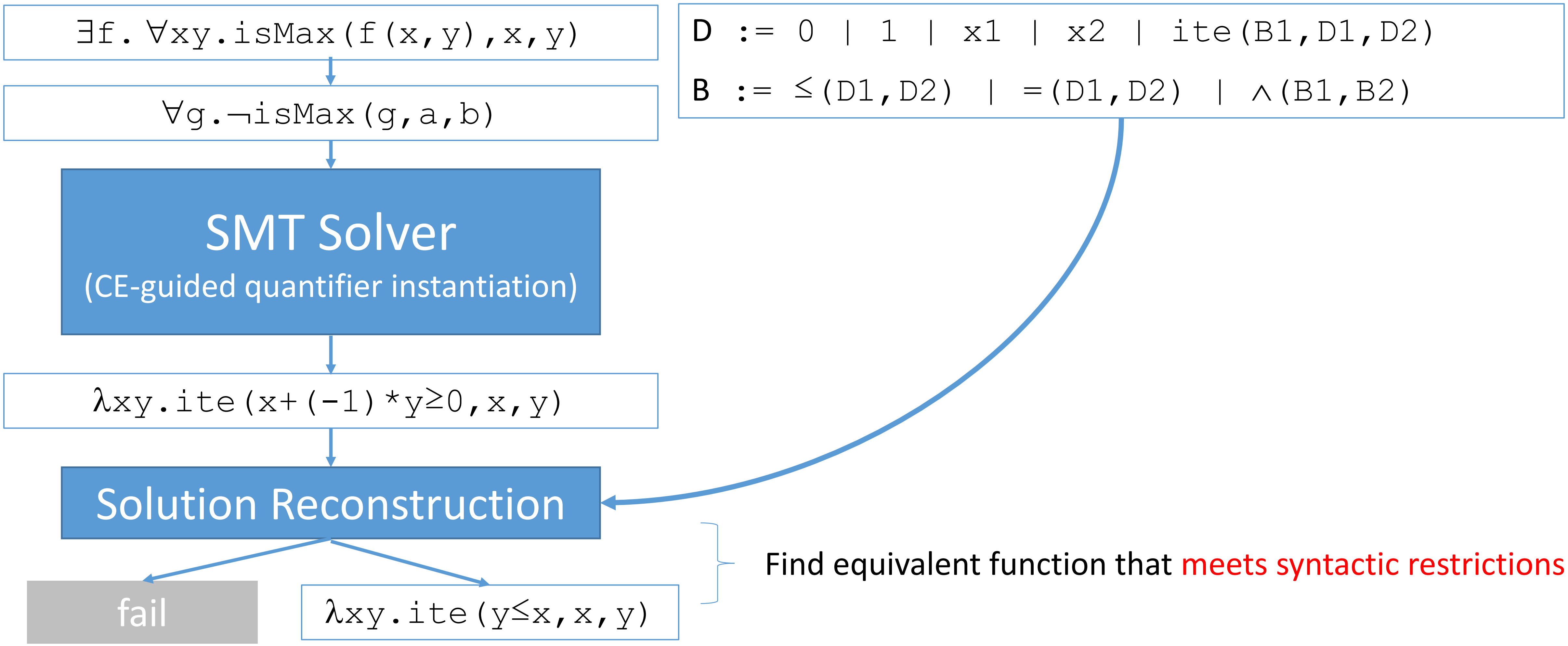
D := 0 | 1 | x1 | x2 | ite(B1,D1,D2)

B := ≤(D1,D2) | =(D1,D2) | ∧(B1,B2)

Single Invocation + Syntactic Restrictions



Single Invocation + Syntactic Restrictions



Evaluation

- Implemented techniques in SMT solver CVC4
- Compared CVC4 against tools taken from 2014 SyGuS competition
  - In particular: enumerative CEGIS solver **esolver** (Upenn)
- Of 243 benchmarks from this competition:
  - 176 were single invocation

Results: Single-Invocation Properties

	array (32)		**bv** (7)		**hd** (56)		**icfp** (50)		**int** (15)		**let** (8)		**multf** (8)		**Total** (176)	
	#	time	#	time	#	time	#	time	#	time	#	time	#	time	#	time
esolver	4	2250.7	2	71.2	50	878.5	0	0	5	1416.7	2	0.0	7	0.6	70	4617.7
cvc4+si-r	(32)	1.2	(6)	4.7	(56)	2.1	(43)	3403.5	(15)	0.6	(8)	1.0	(8)	0.2	(168)	3413.3
cvc4+si	30	1449.5	5	0.1	52	2322.9	0	0	6	0.1	2	0.5	7	0.1	102	3773.2

- Considered CVC4:
  - With solution reconstruction **cvc4+si**
  - Without solution reconstruction **cvc4+si-r**
- **cvc4+si** solves 35 that **esolver** does not
- **esolver** solves 3 that **cvc4+si** does not
- **cvc4+si** solves 25 benchmarks unsolved by any other known solver
  - Many of these in fraction of a second

Non-single invocation Properties

	int (3)		**invgu** (28)		**invg** (28)		**vctrl** (8)		**Total** (67)	
	#	time	#	time	#	time	#	time	#	time
esolver	3	1.6	25	86.3	25	85.6	5	29.5	58	203.0
cvc4+sg	3	1476.0	23	811.6	22	2283.2	5	2933.1	53	7503.9

- **cvc4+sg** fairly competitive with **esolver**
  - **cvc4+sg** solves 2 that **esolver** does not
  - **esolver** solves 7 that **cvc4+sg** does not

CVC4 in Sygus Comp 2015

- Won General and LIA tracks

LIA Track		73	
Solver	#solved	total-expr-size	average-expr-size
CVC4-1.5-syguscomp2015-v4	70	43726	624.66
AlchemistCSDT	47	6658	141.66
Alchemist CS	33	866	26.24

  - In LIA track, solved 70/73 benchmarks, 60 of these in <1 second
    - Nearest competitor **AlchemistCSDT** solved 47/73 in a timeout of 1 hour
- Did not win INV track (won by **IceDT**)
  - Due to form of benchmarks, for transition relations T:

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

$\Rightarrow$Resorts to syntax-guided approach

Max example : Sygus Comp 2015

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cvc4+si	0.0	0.0	0.0	0.0	0.1	0.1	0.2	0.3	0.6	1.0	1.9	3.2	5.3	6.5
AlchemistCSDT	0.2	0.6	1.5	6.4	20.8	132.8	877.9	–	–	–	–	–	–	–
AlchemistCS	0.0	3.7	–	–	–	–	–	–	–	–	–	–	–	–

- Outperforms existing approaches by an order of magnitude or more

$\Rightarrow$ Our approach is highly efficient for synthesizing non-recursive functions that are defined by cases

Summary

- Refutation-based approach for synthesis
  - Highly competitive for single invocation properties
- Uses Counterexample-Guided Quantifier Instantiation
  - *Applicable to theorem proving, not just synthesis*
    - Also used in **SMT Comp** 2014 and 2015, **CASC** J7 and 25
- Solutions constructed from unsat core of instantiations
- Implemented in CVC4

Thanks!

- CVC4 publicly available at:

  http://cvc4.cs.nyu.edu/web/

- Handles inputs in the sygus language format *.sl