Using Instantiation-Based Methods for Quantifier Elimination in SMT

Andrew Reynolds

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Overview

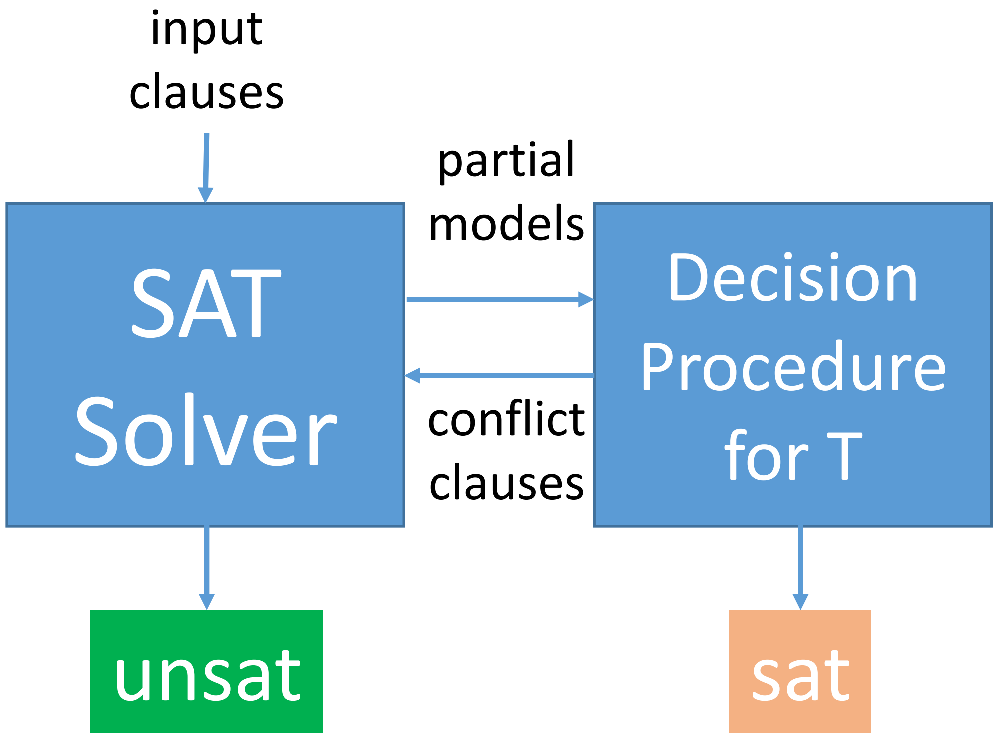
- Basics of quantified formulas in SMT
- New: approach for quantifier elimination that is
  - Tightly integrated in the SMT solver
  - Incremental
  - Model-Driven

  $\Rightarrow$ will focus only on linear arithmetic

CVC4

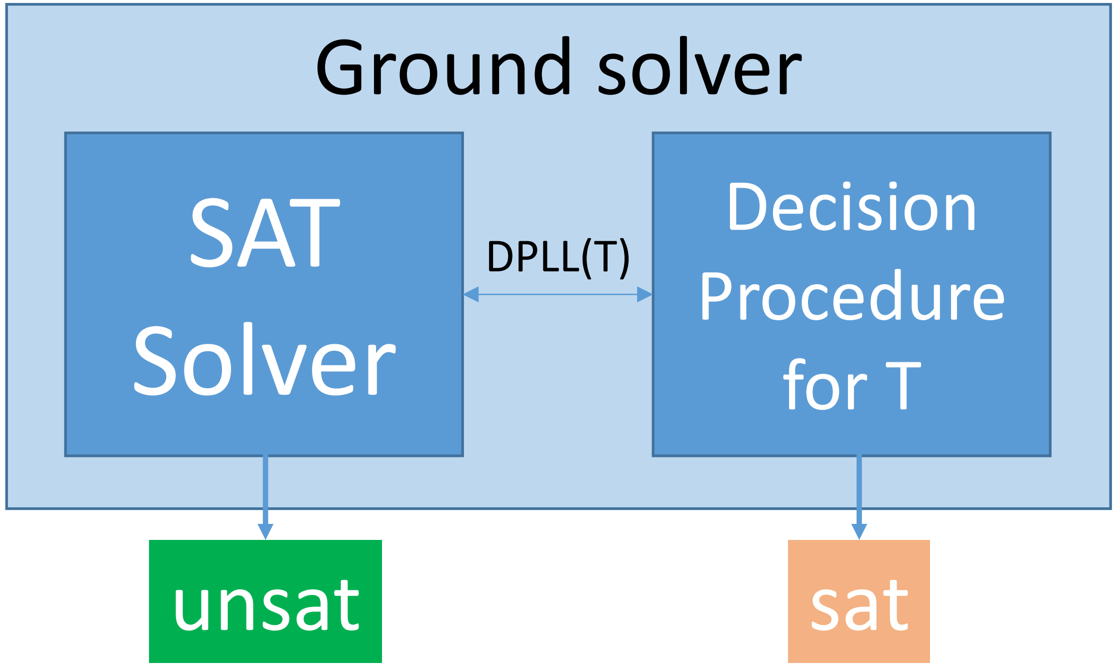
- Jointly developed at NYU and U of Iowa
  - Now at EPFL, Oxford, Google
- State-of-the-art performance
  - Won SMT COMP 2015
  - Won TFA division of CASC J7, TFN division of CASC 25
- Supports many theories
  - UF, Bitvectors, Arrays
  - (Linear) arithmetic
  - Datatypes, Strings, Sets
- Has techniques for arbitrary first-order quantified formulas
  - Complete for certain fragments
  - Heuristic for the general case (problem is undecidable)

DPLL(T)-based SMT Solver



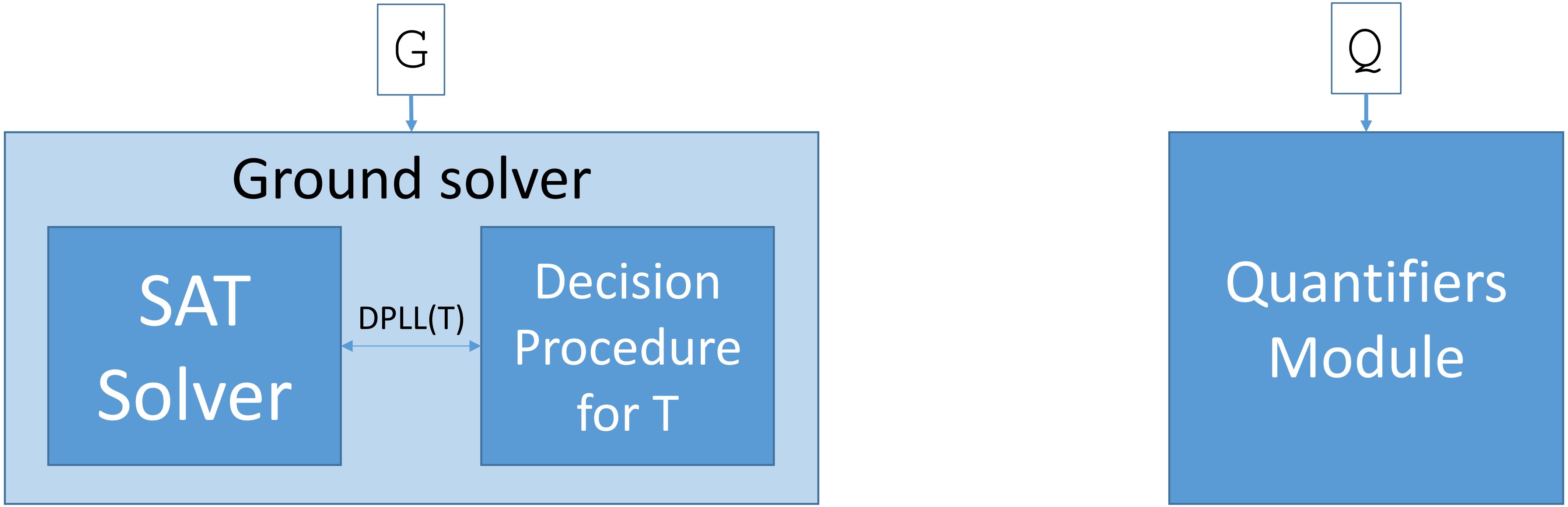
- DPLL(T)-based SMT solver
  - SAT solver maintains a set of propositional clauses
  - Decision Procedure for T determines satisfiability of conjunctions of T-literals

DPLL(T)-based SMT Solver



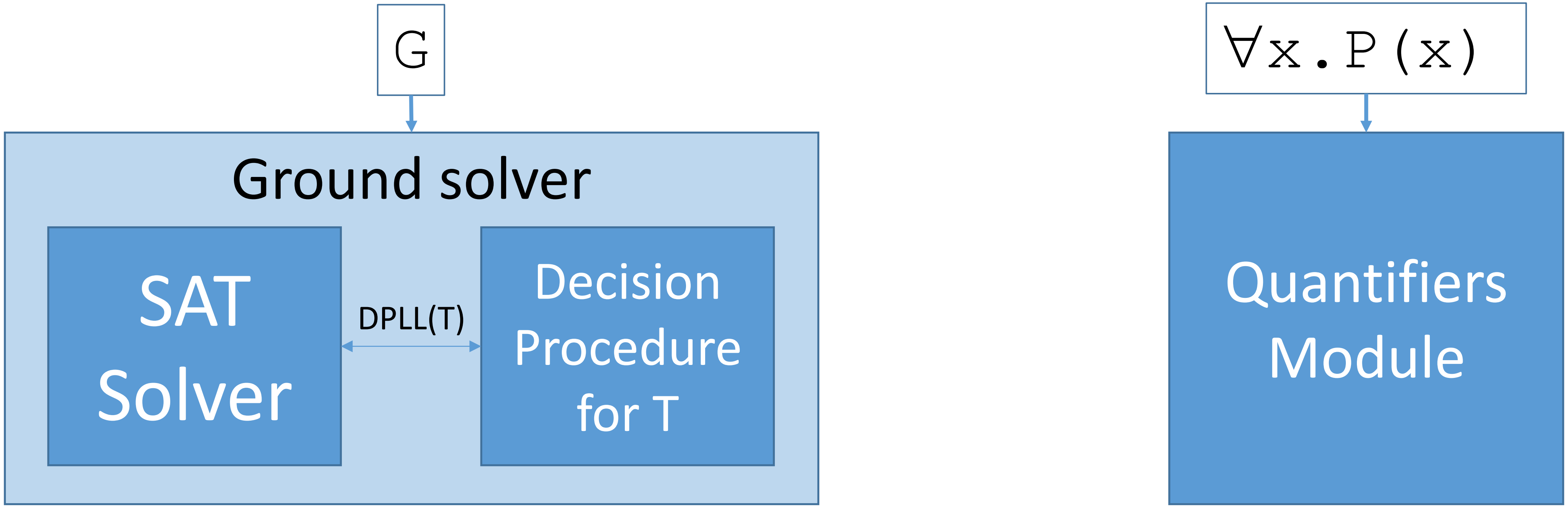
- Ground solver = SAT solver + Decision Procedure for T

DPLL(T) + Quantifiers



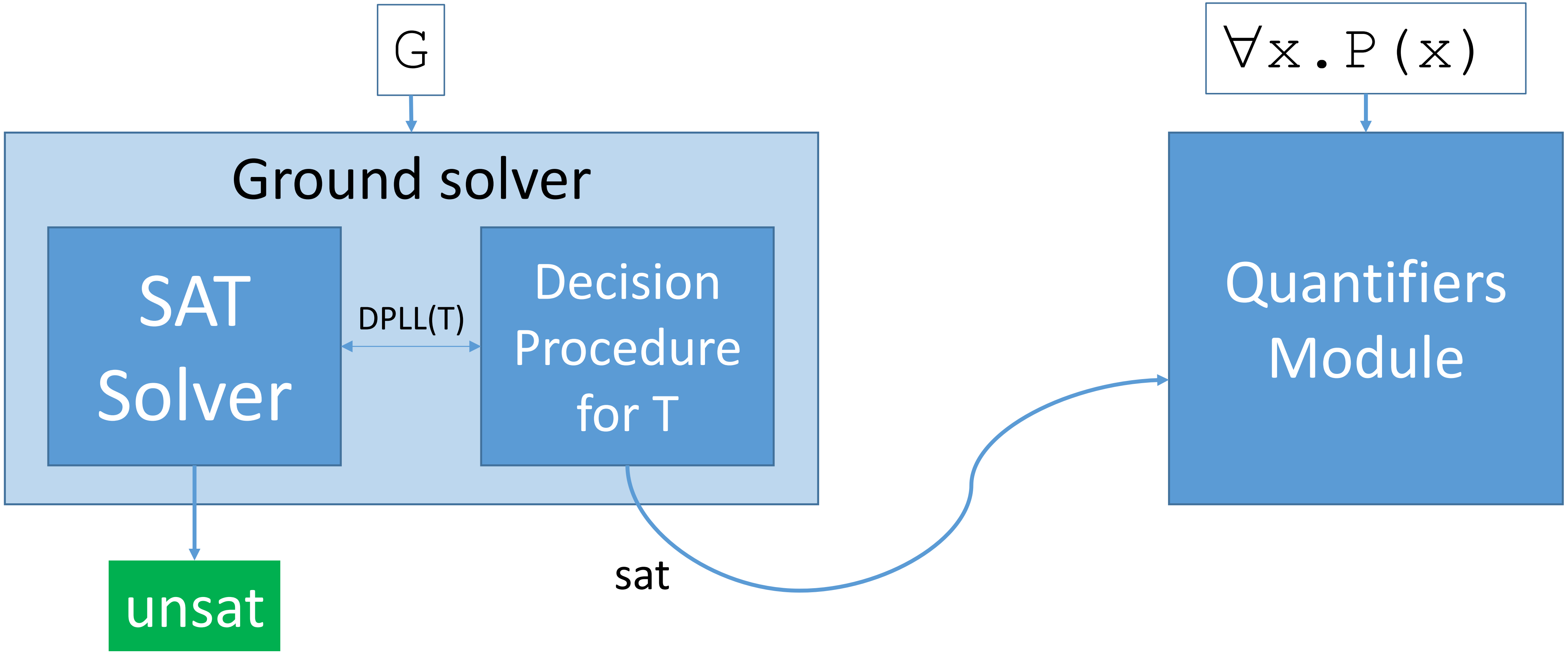
- SMT solver consists of:
  - Ground solver maintains a set of ground (quantifier-free) constraints G
  - Quantifiers Module maintains a set of universally quantified formulas Q

DPLL(T) + Quantifier Instantiation



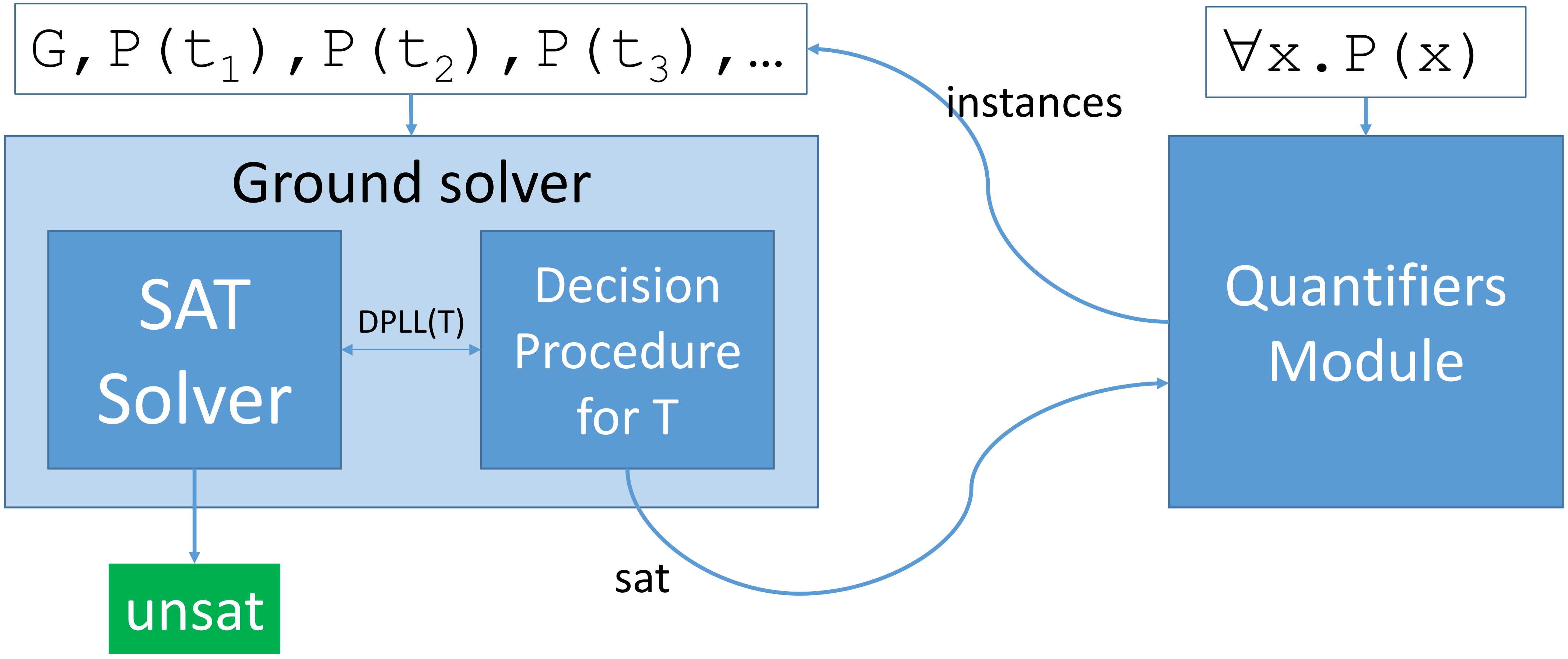
- Primary technique for quantifiers in this talk: **Quantifier Instantiation**

DPLL(T) + Quantifier Instantiation



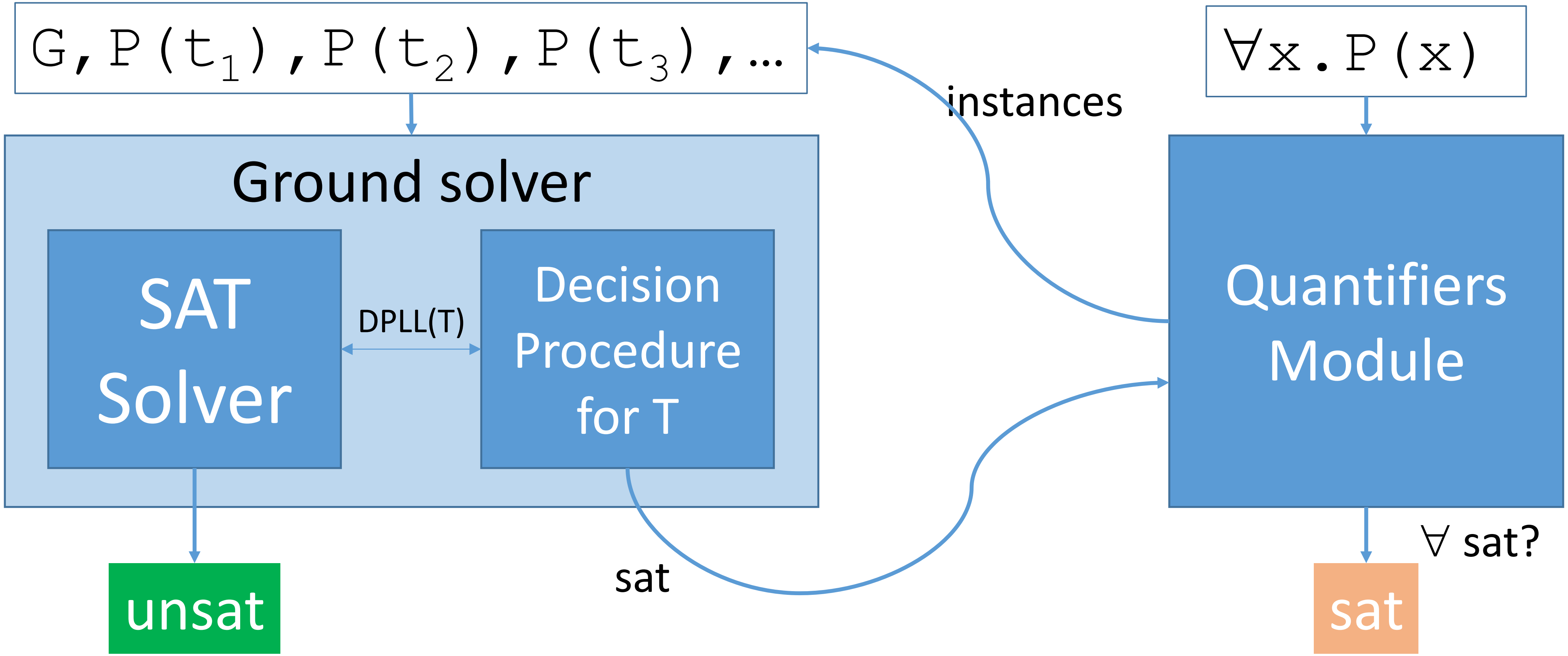
- If G is T-satisfiable, invoke quantifiers module

DPLL(T) + Quantifier Instantiation



- Add instances of axioms to G

DPLL(T) + Quantifier Instantiation



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

Instantiation-Based Approaches for ∀

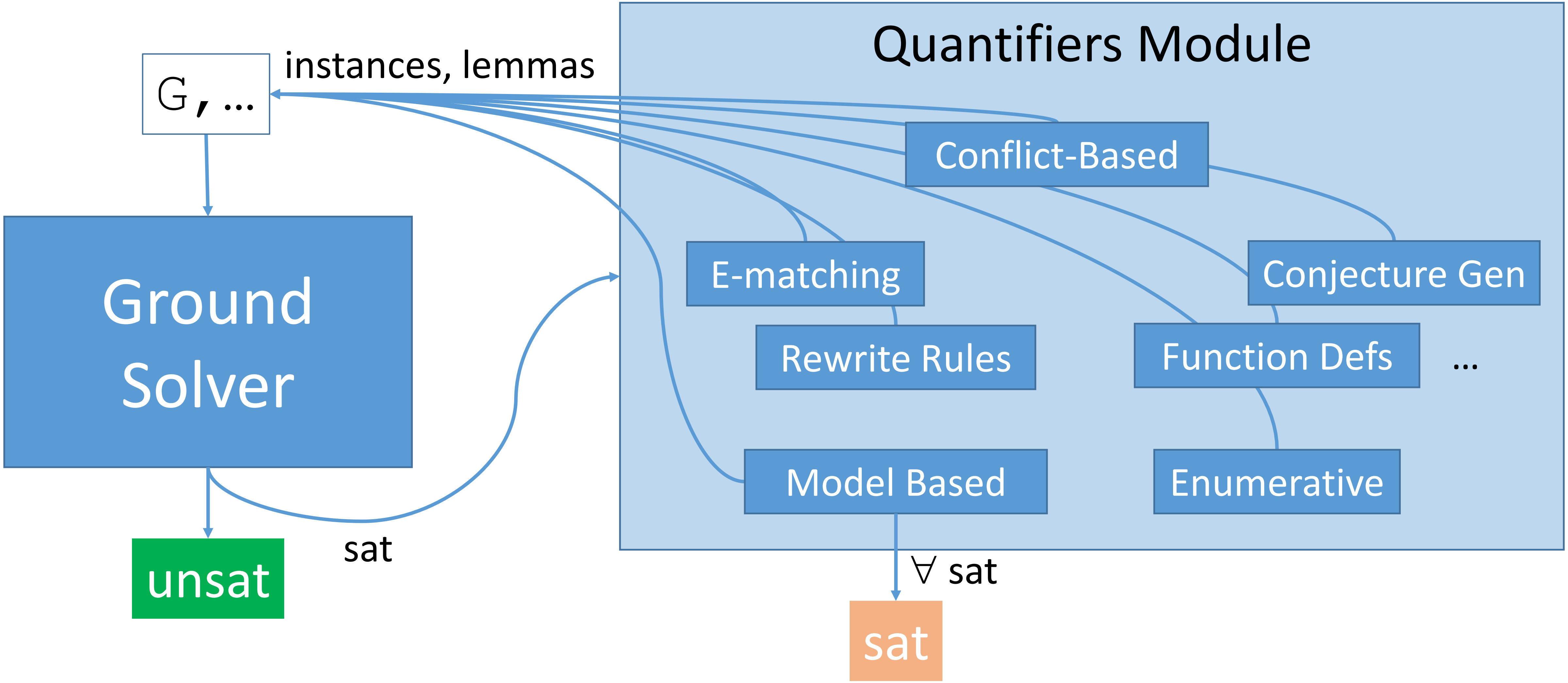
- Heuristic instantiation (good for "unsat"):
  - E-matching [Detlefs et al 2003, Ge et al 2007, de Moura/Bjorner 2007]
  - Conflict-based [Reynolds/Tinelli/de Moura 2014]
- Complete approaches (may answer "sat"):
  - Local theory extensions [Sofronie-Stokkermans 2005, Bansal et al 2015]
  - Inst-Gen [Ganzinger/Korovin 2003]
  - Array fragments [Bradley et al 2006, Alberti et al 2014]
  - Complete instantiation [Ge/de Moura 2009]
  - Finite model finding [Reynolds et al 2013]

  $\Rightarrow$ Each limited to a particular fragment

Instantiation-Based Approaches for ∀

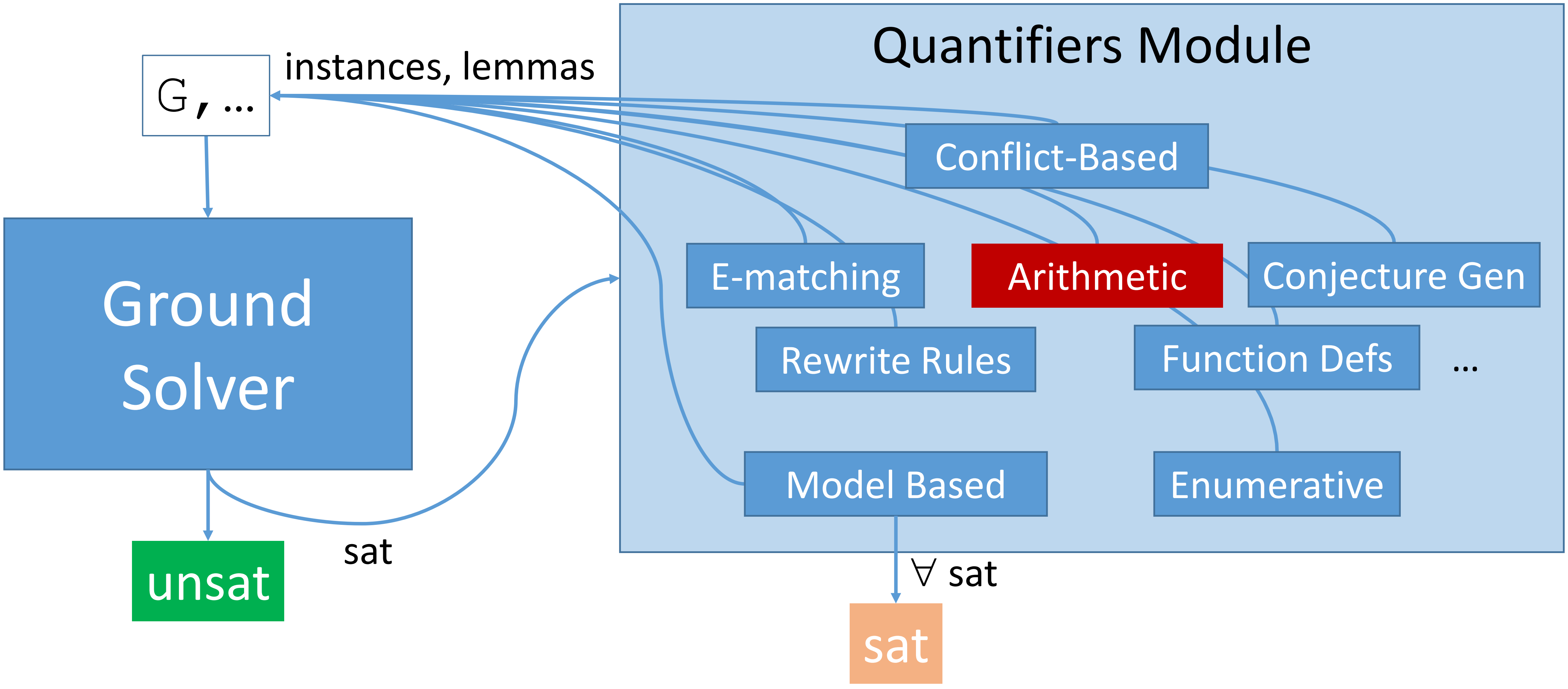
- Heuristic instantiation (good for "unsat"):
  - E-matching [Detlefs et al 2003, Ge et al 2007, de Moura/Bjorner 2007]
  - Conflict-based [Reynolds/Tinelli/de Moura 2014]
- Complete approaches (may answer "sat"):
  - Local theory extensions [Sofronie-Stokkermans 2005]
  - Inst-Gen [Ganzinger/Korovin 2003]
  - Array fragments [Bradley et al 2006, Alberti et al 2014]
  - Complete instantiation [Ge/de Moura 2009]
  - Finite model finding [Reynolds et al 2013]
  - Fragments that admit quantifier elimination (e.g. pure arithmetic)
    - Studied in the context of SMT solving [Monniaux 2010, Bjorner 2012]

Quantifiers Module of CVC4



- CVC4's quantifiers module contains numerous strategies and techniques

Quantifiers Module of CVC4



- Specialized techniques for quantified arithmetic

Synthesis: Motivation

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

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

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:

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

How does an SMT solver handle Max example?

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

- Challenge: quantification over function **f**
  - No SMT solvers directly support second-order quantification

How does an SMT solver handle Max example?

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

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

- Direct approach:
  - Treat **f** as an *uninterpreted function*
  - Succeed if SMT solver can find correct interpretation of **f**

    ⇒*This is challenging*
    - How does the solver know the right interpretation for **f** to pick?

How does an SMT solver handle Max example?

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

How does an CVC4 handle Max example?

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

- Alternative:
  - This property is **single invocation**
    - All occurrences of **`f`** are of the form **`f(x,y)`**

… and thus, can be converted to a first-order quantification
    - Introduce first-order variable **`g`**
    - Push quantification downwards “anti-skolemization”

[Reynolds/Deters/Kuncak/Tinelli/Barrett CAV15]

How does an CVC4 handle Max example?

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

Convert to first-order

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

[Reynolds/Deters/Kuncak/Tinelli/Barrett CAV15]

How does an CVC4 handle Max example?

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

⇓ Convert to first-order

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

- Problem is now:
  - First-order, linear (integer) arithmetic, with one quantifier alternation

    ⇒ CVC4 has specialized instantiation procedure

[Reynolds/Deters/Kuncak/Tinelli/Barrett CAV15]

Max Example

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

Max Example

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

Max Example

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

- Goal: show the above formula is sat

Max Example

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

- Since F is LIA-sat if and only if $\neg$F is LIA-unsat,
  $\Rightarrow$ Suffices to show that negation is unsat

Max Example

$(\exists ab.)$

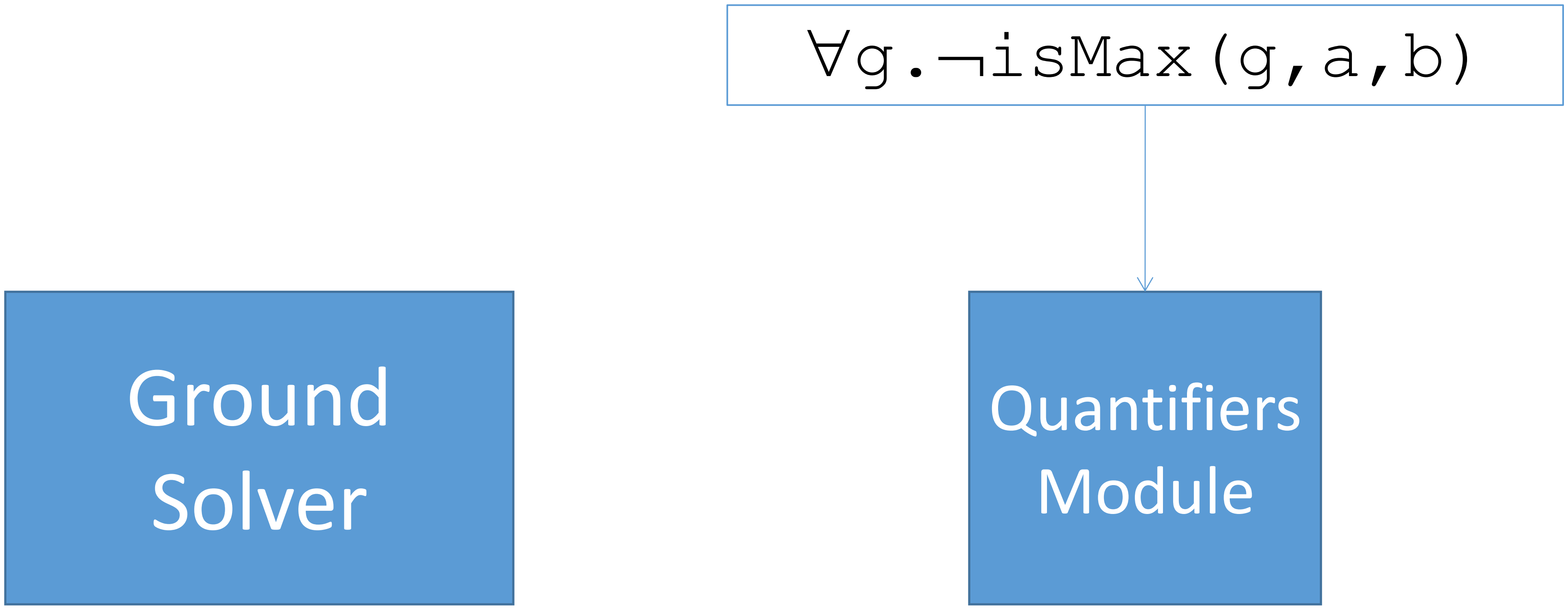
$\forall g.\neg isMax(g,\mathbf{a},\mathbf{b})$

Ground Solver

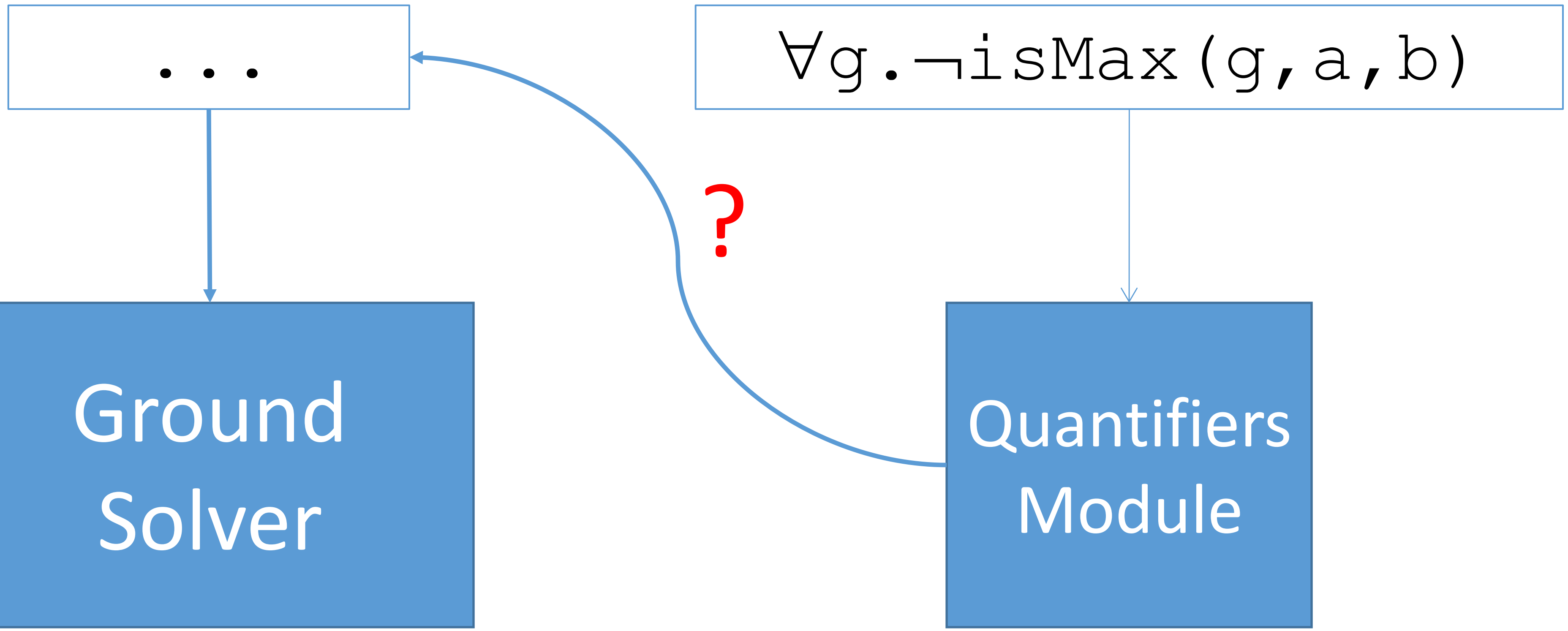
Quantifiers Module

- Skolemize, for fresh constants **a** and **b**

Max Example

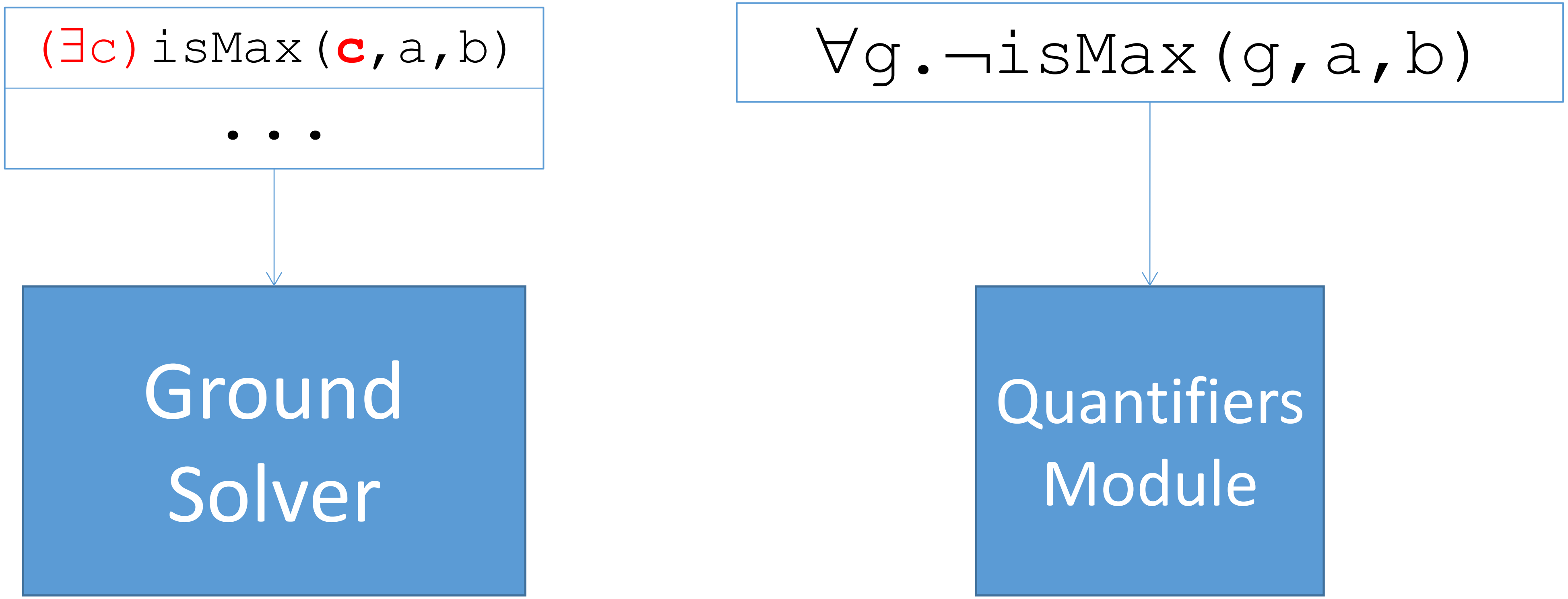


Max Example



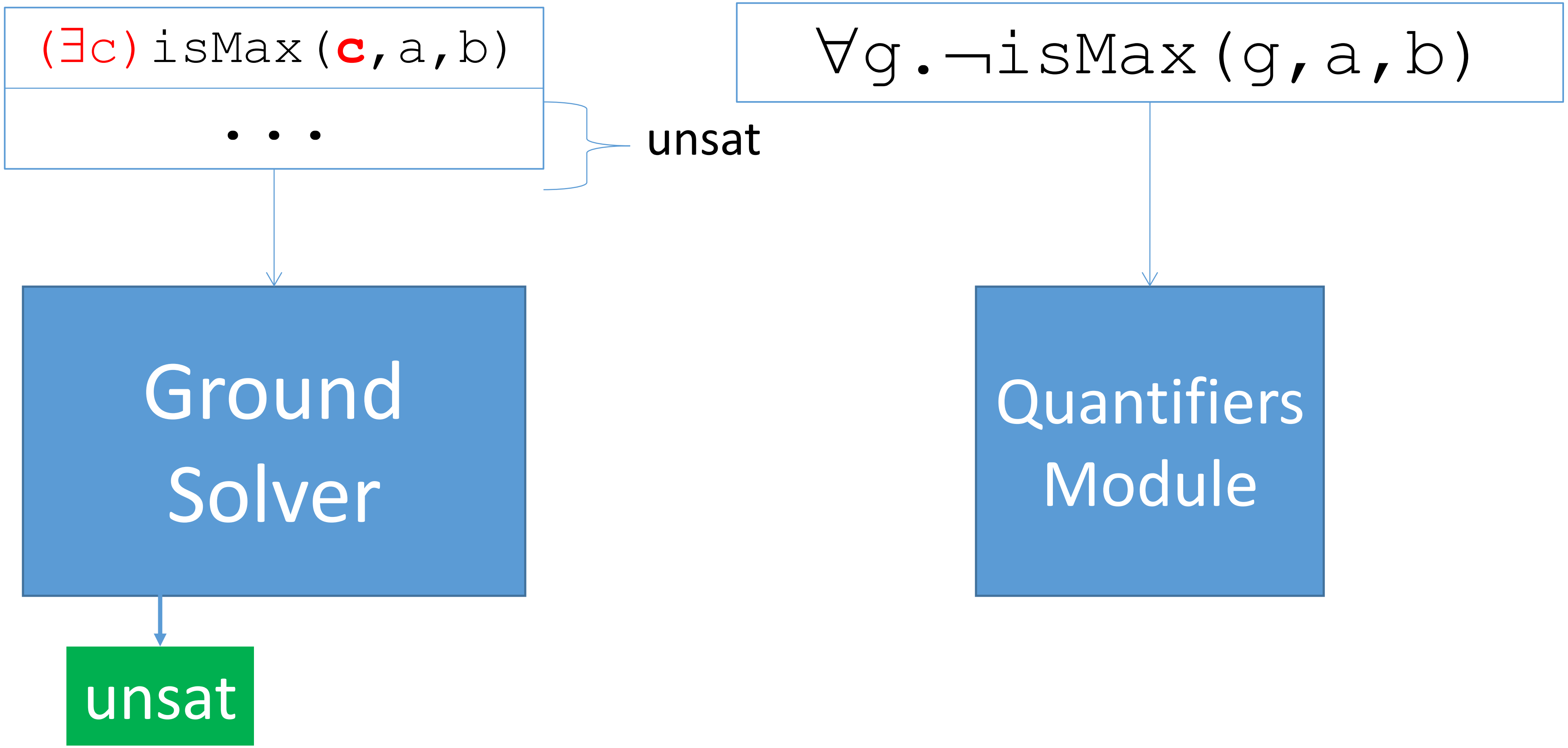
- Which instances of `∀g.¬isMax(g,a,b)` do we consider?

Counterexample-Guided Instantiation



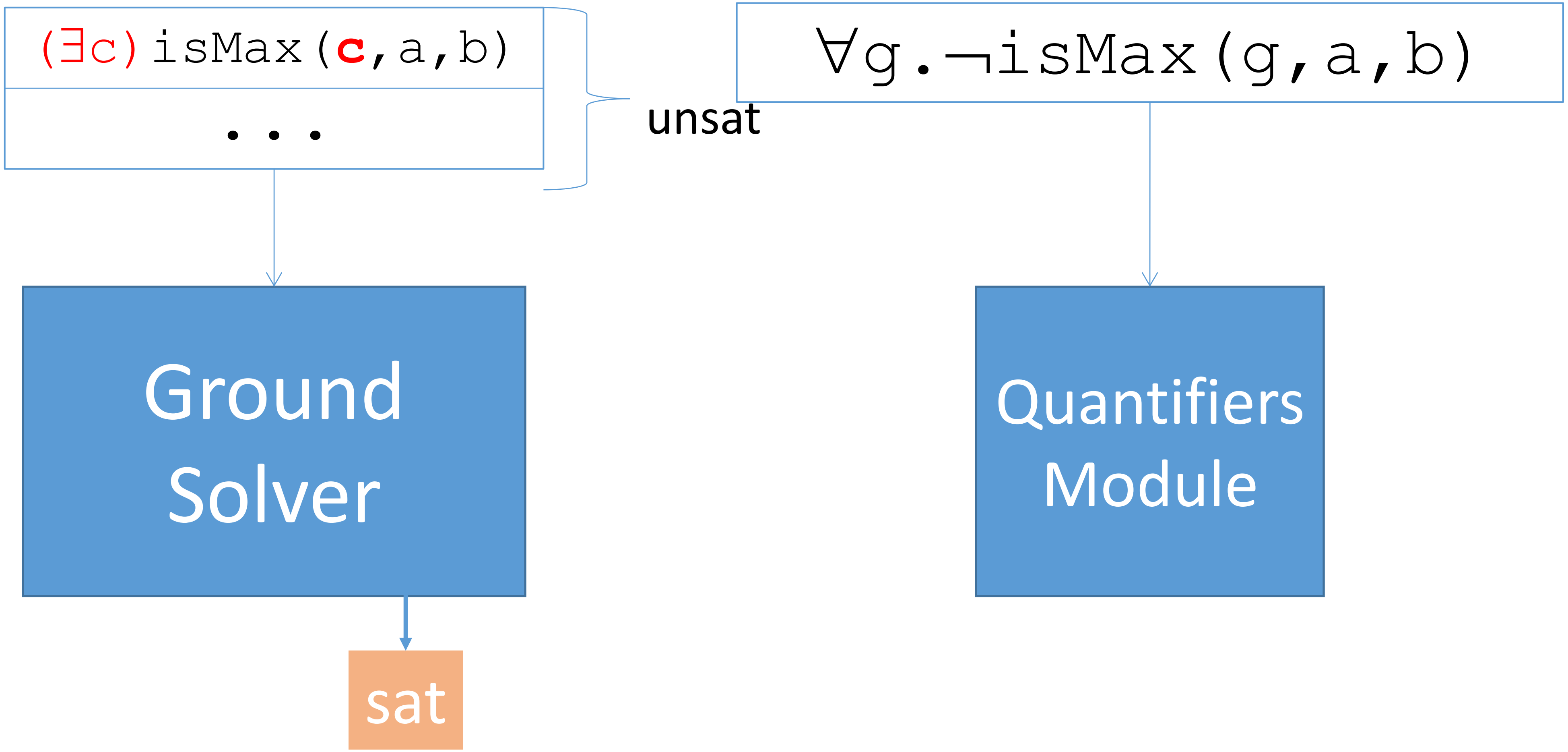
- **Idea**: choose instances of `∀g.¬isMax(g,a,b)` based on models for "counterexample" fresh constant **c**

Counterexample-Guided Instantiation



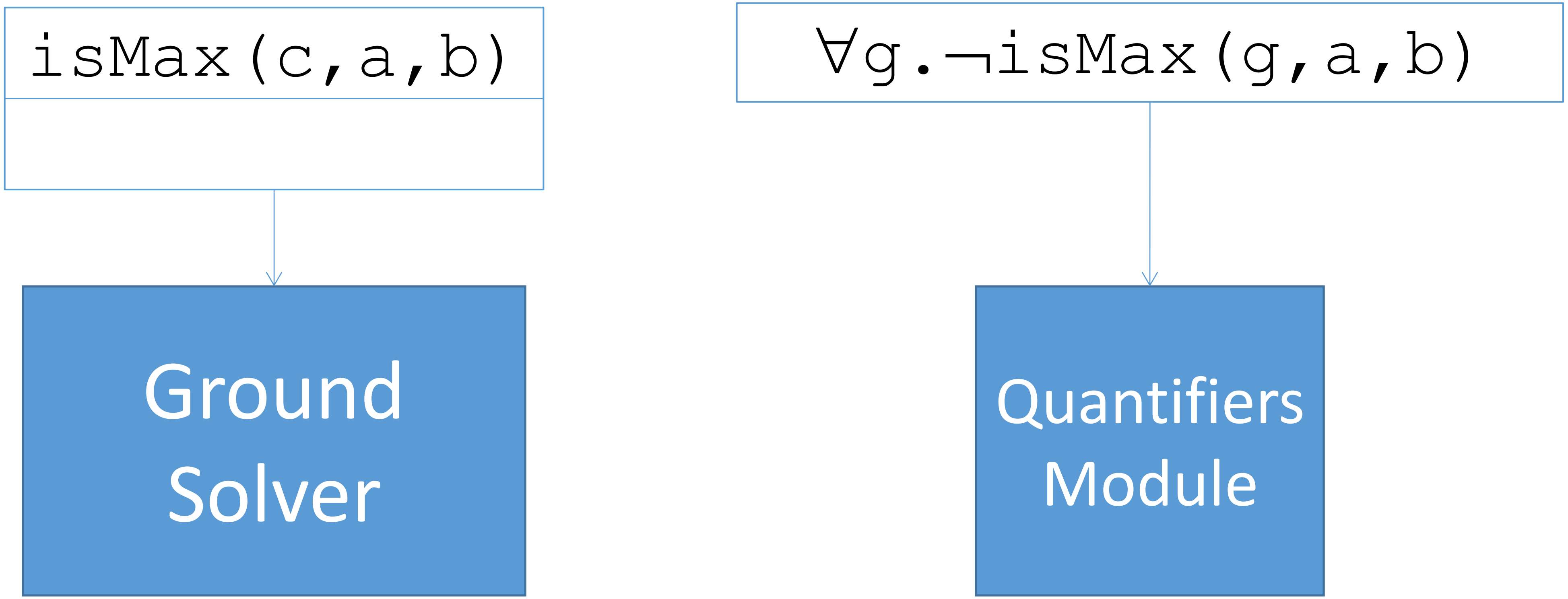
- If ground constraints without CE is unsat, answer "unsat"

Counterexample-Guided Instantiation

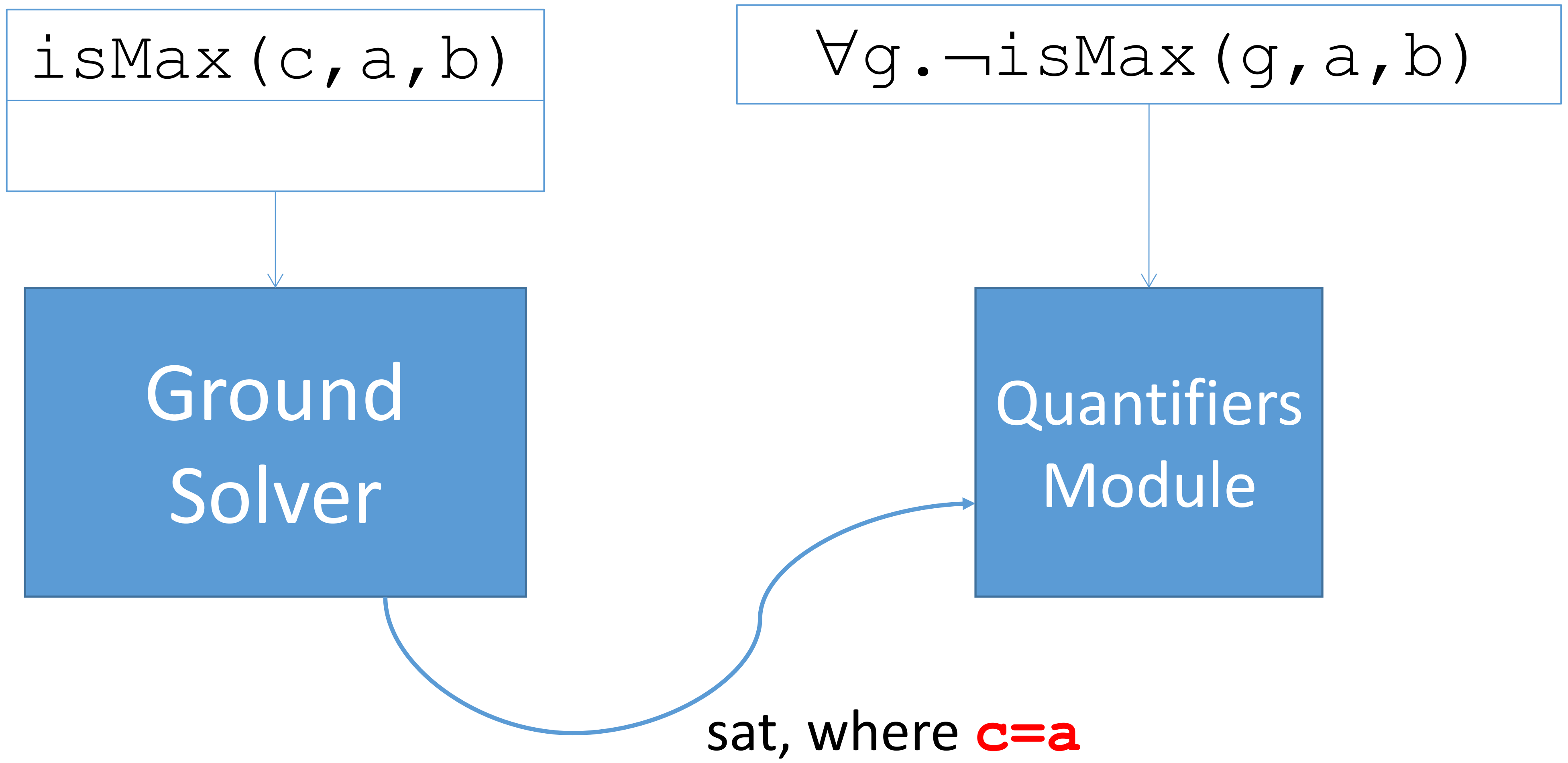


- Else, if ground constraints with CE is unsat, answer "sat"

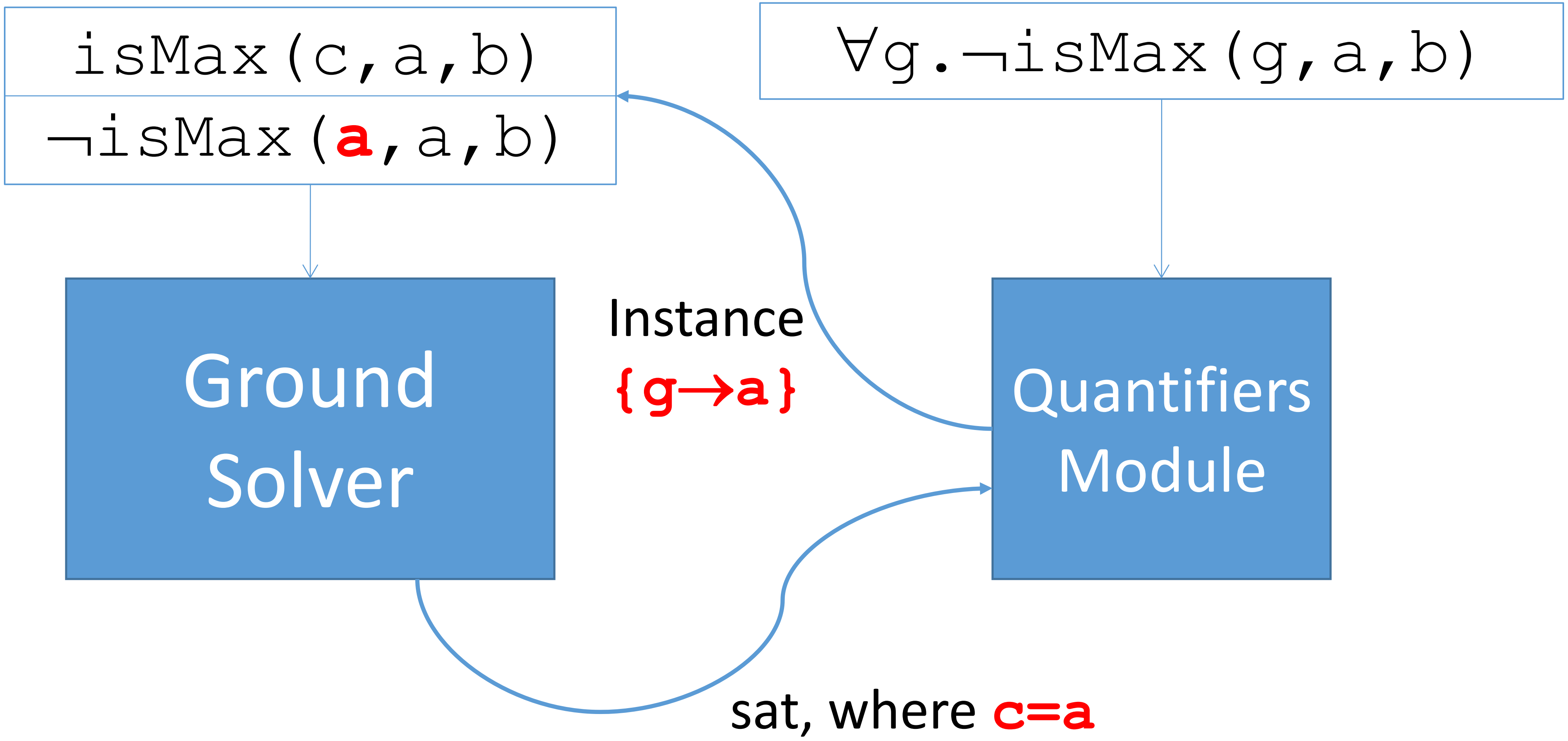
Counterexample-Guided Instantiation



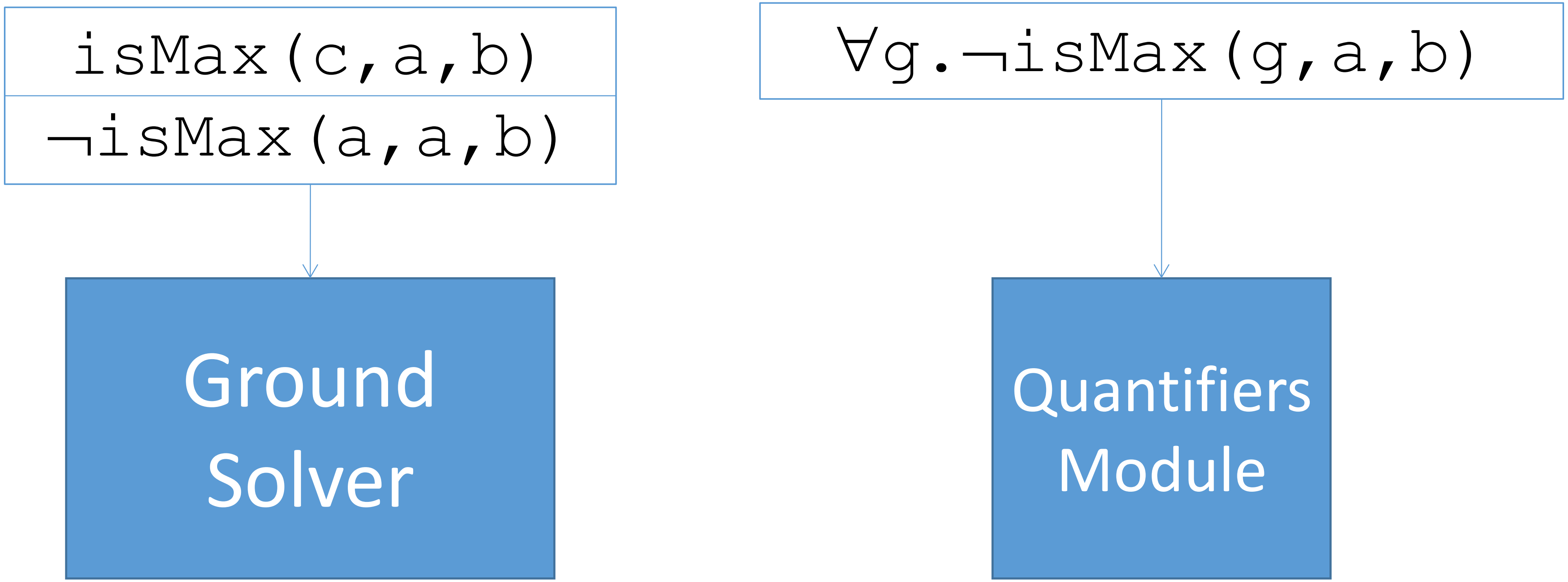
Counterexample-Guided Instantiation



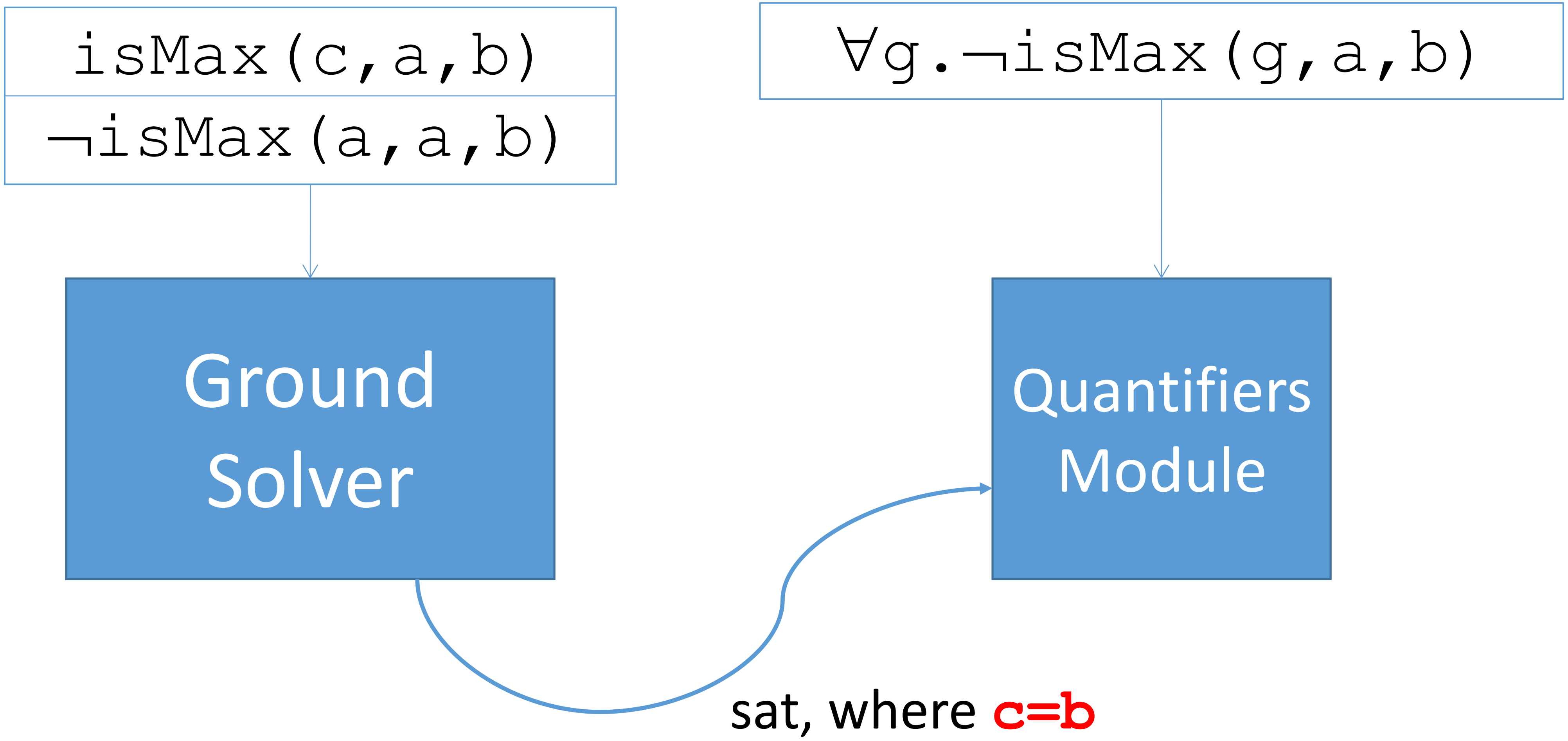
Counterexample-Guided Instantiation



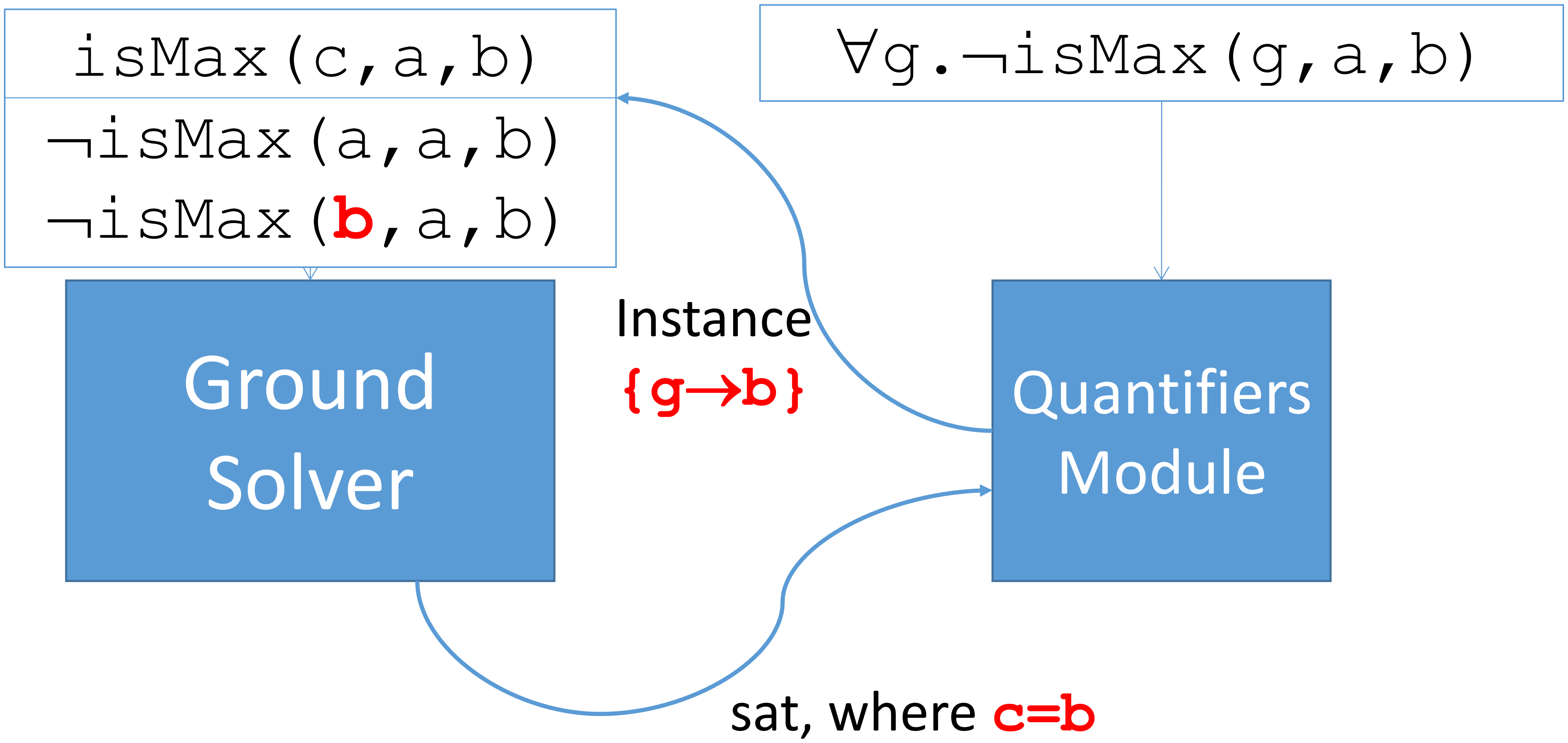
Counterexample-Guided Instantiation



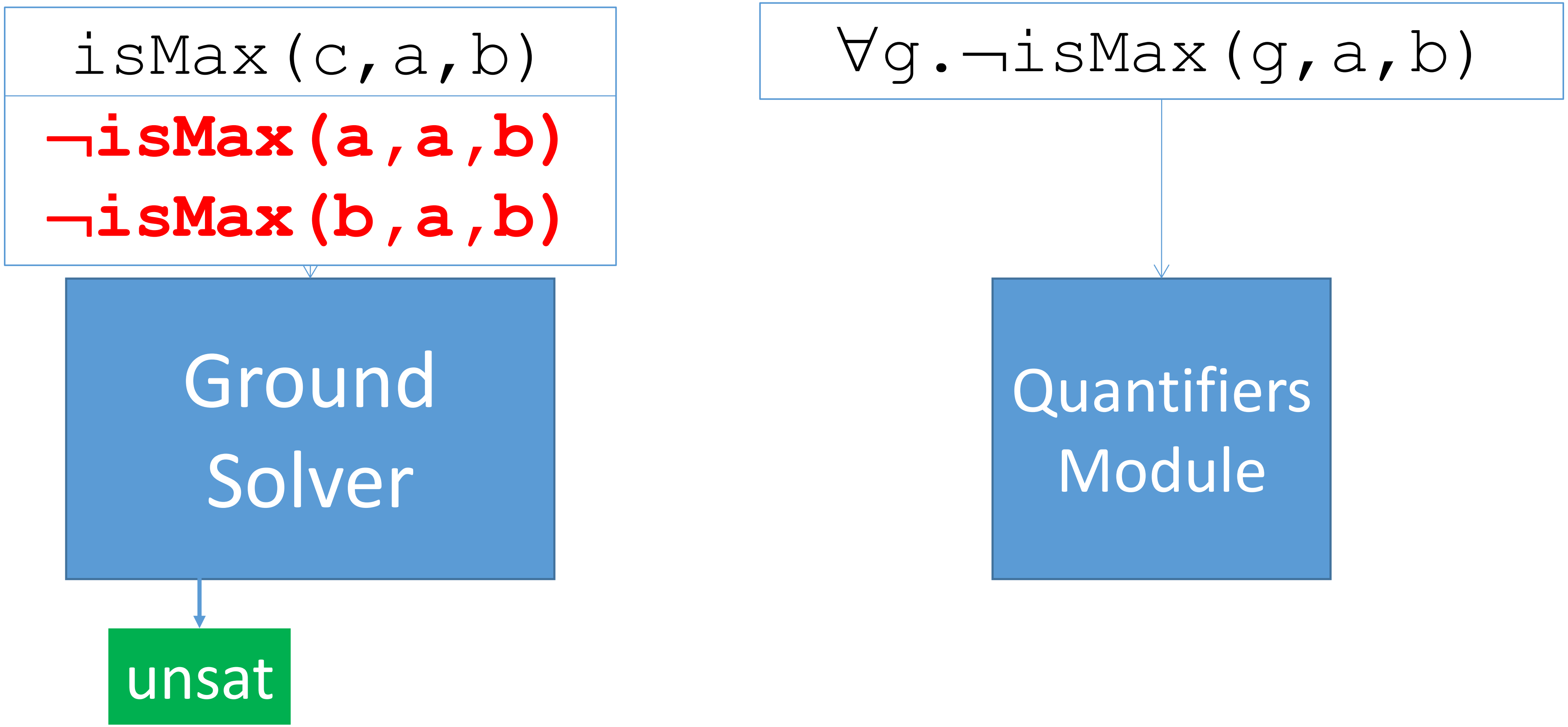
Counterexample-Guided Instantiation



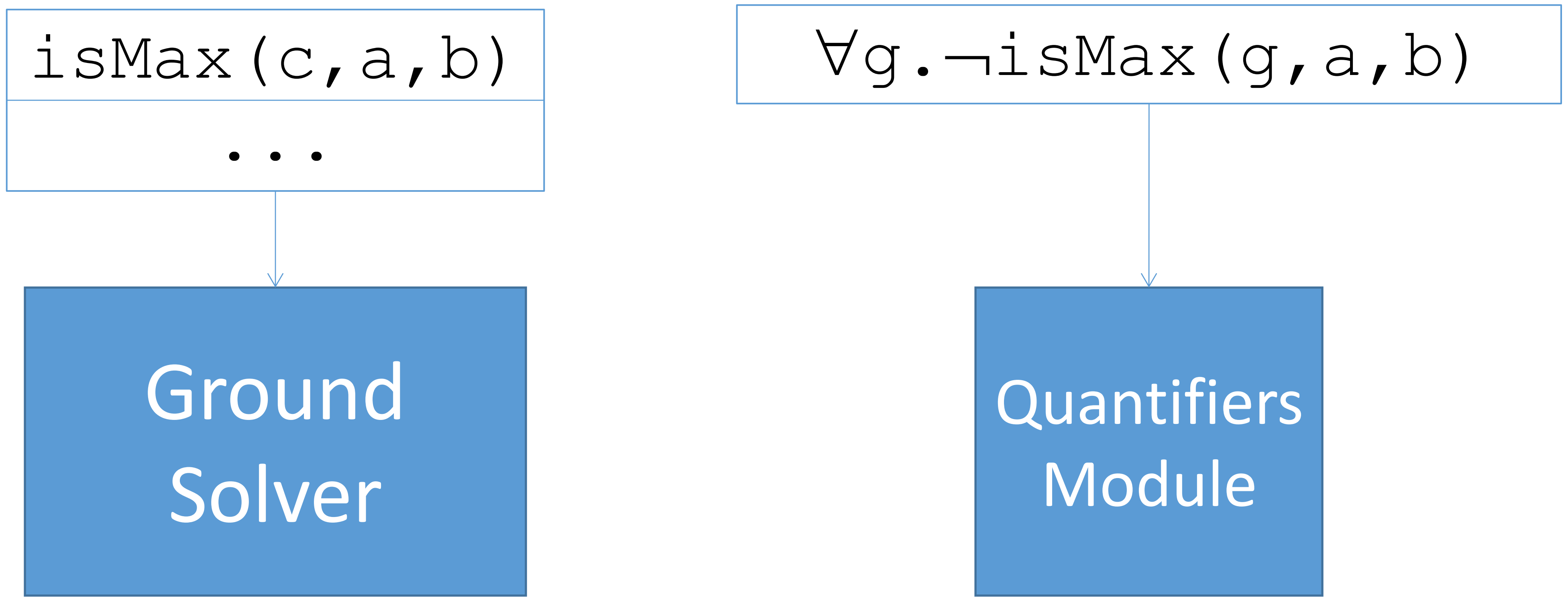
Counterexample-Guided Instantiation



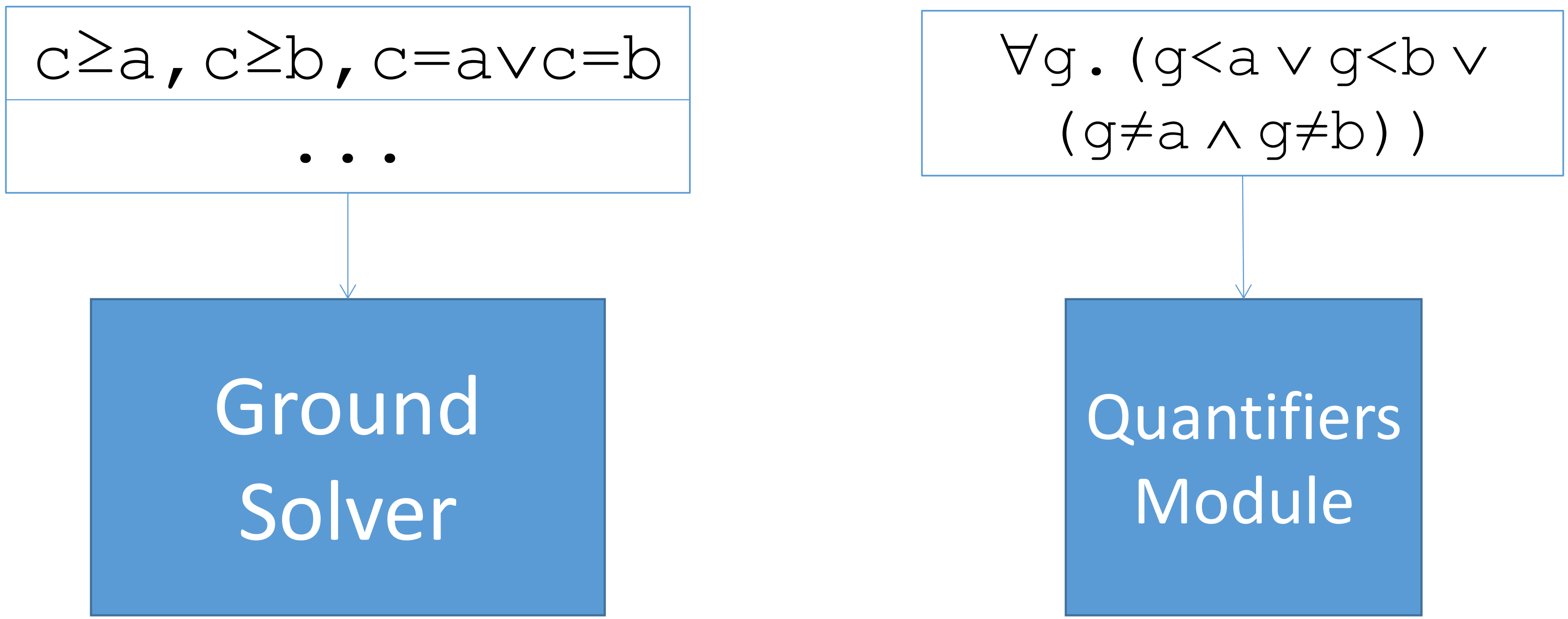
Counterexample-Guided Instantiation



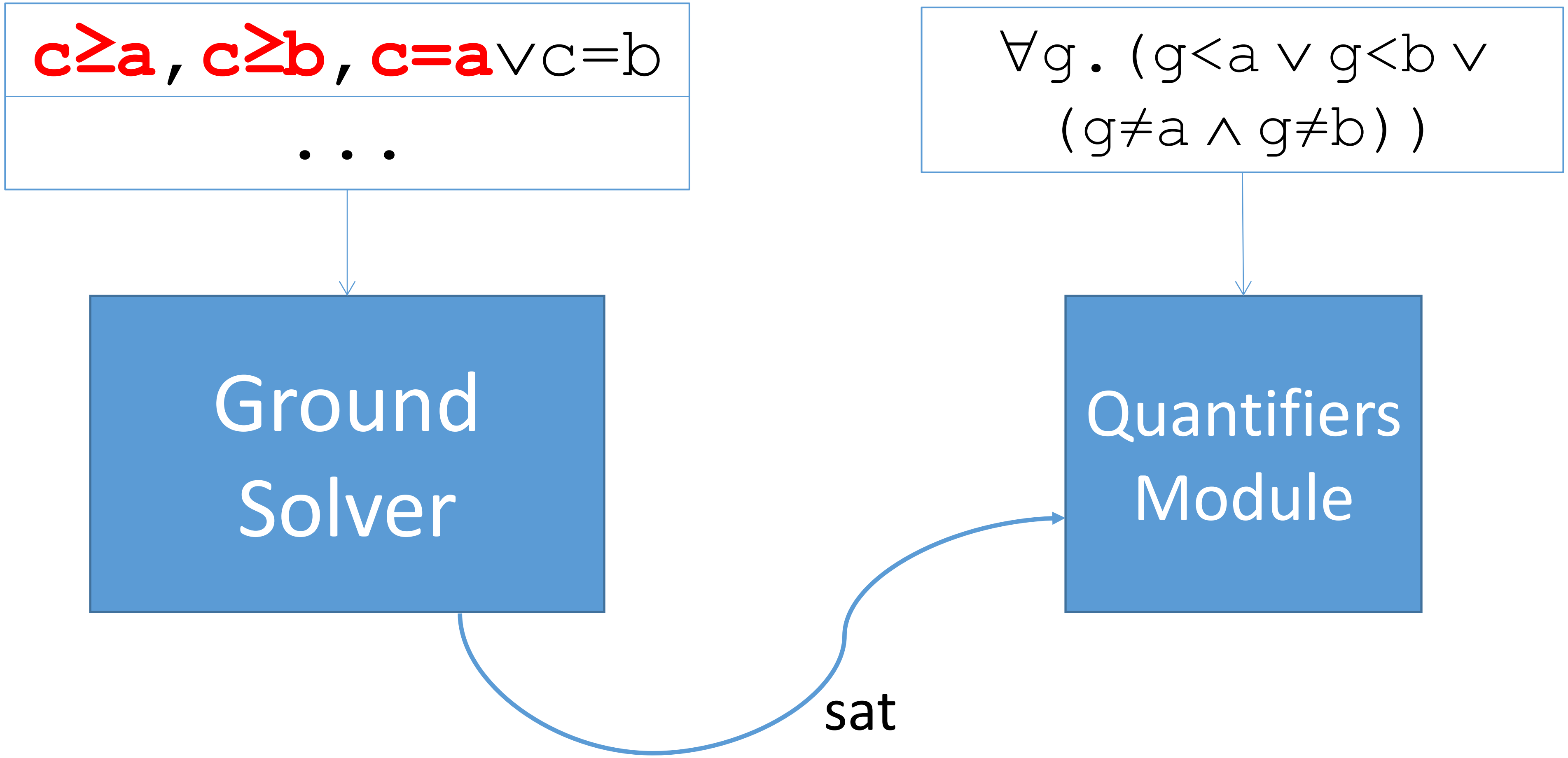
Counterexample-Guided Instantiation



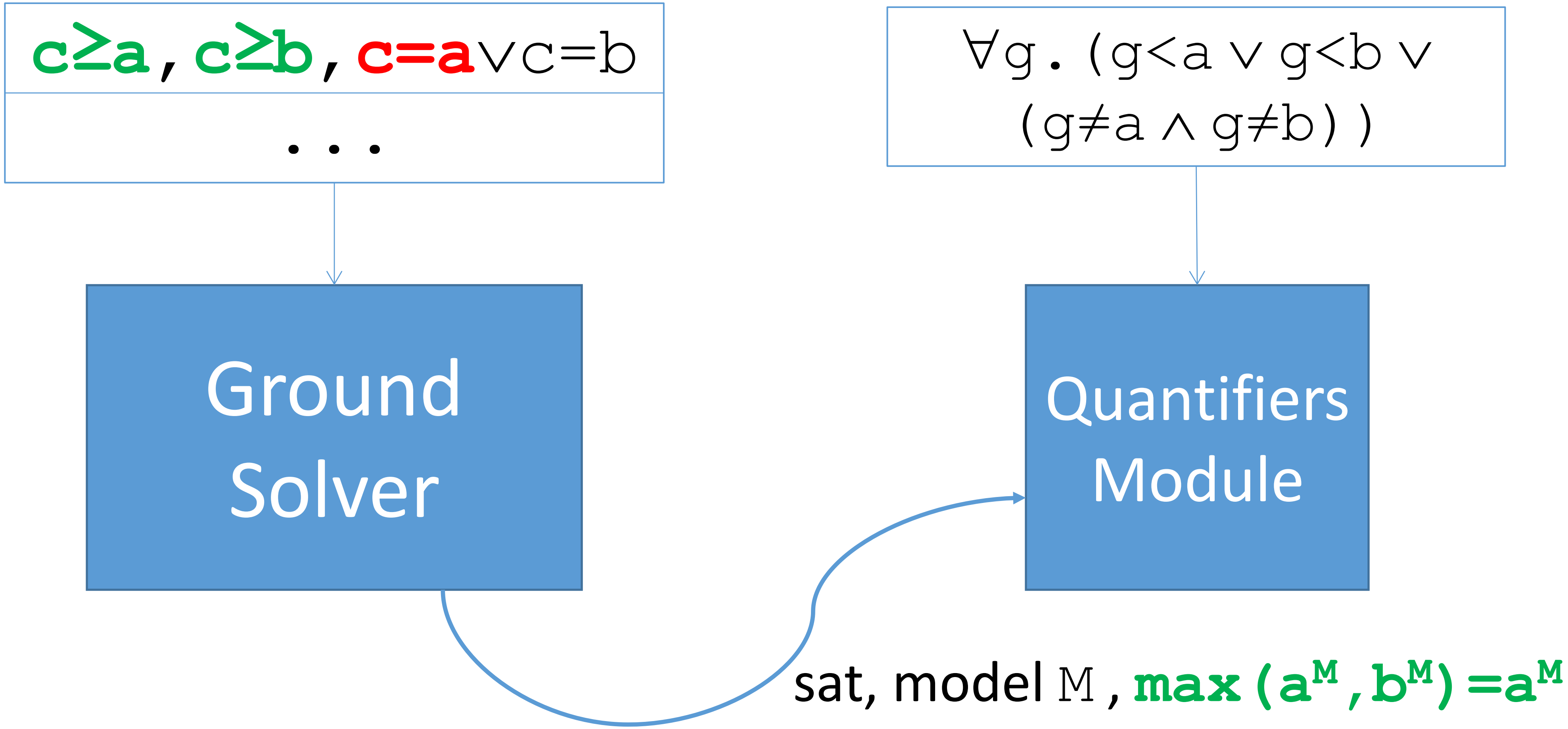
Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

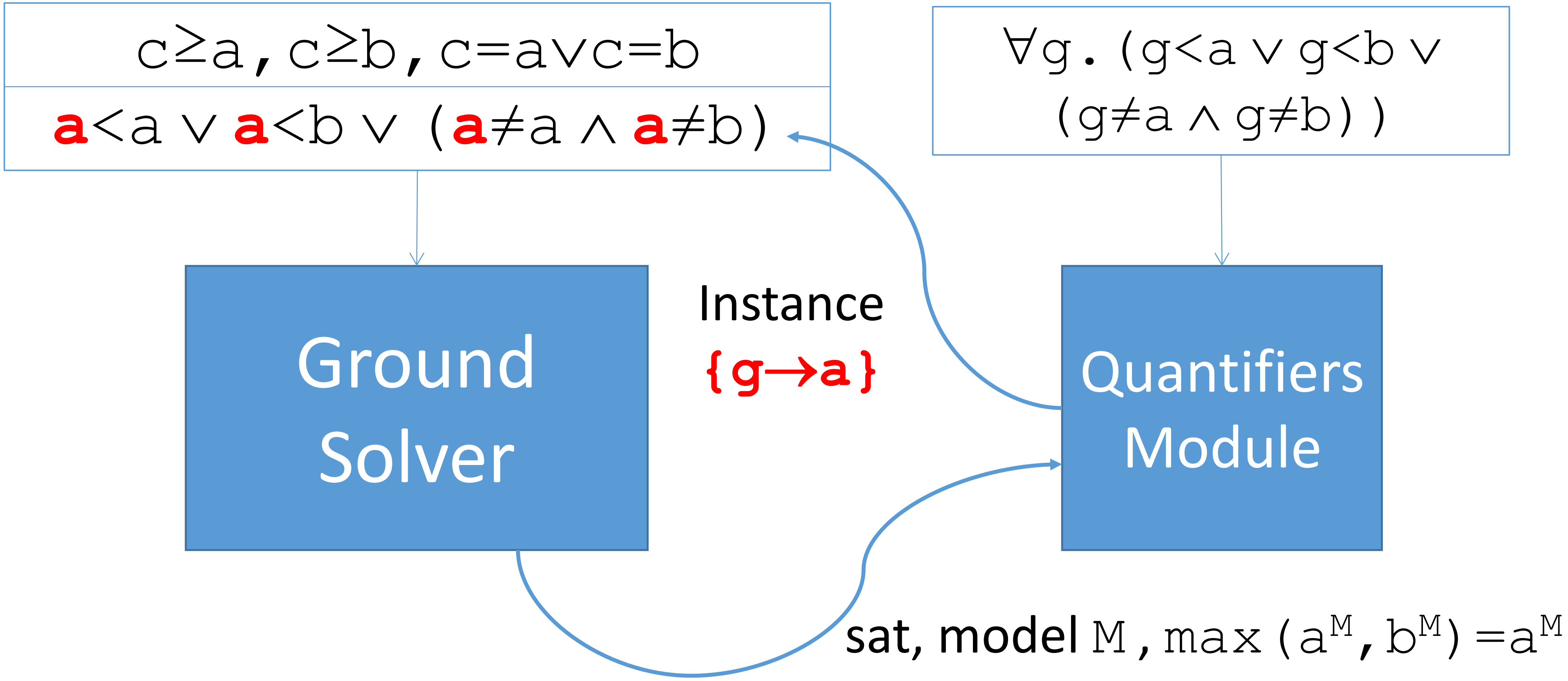


Counterexample-Guided Instantiation

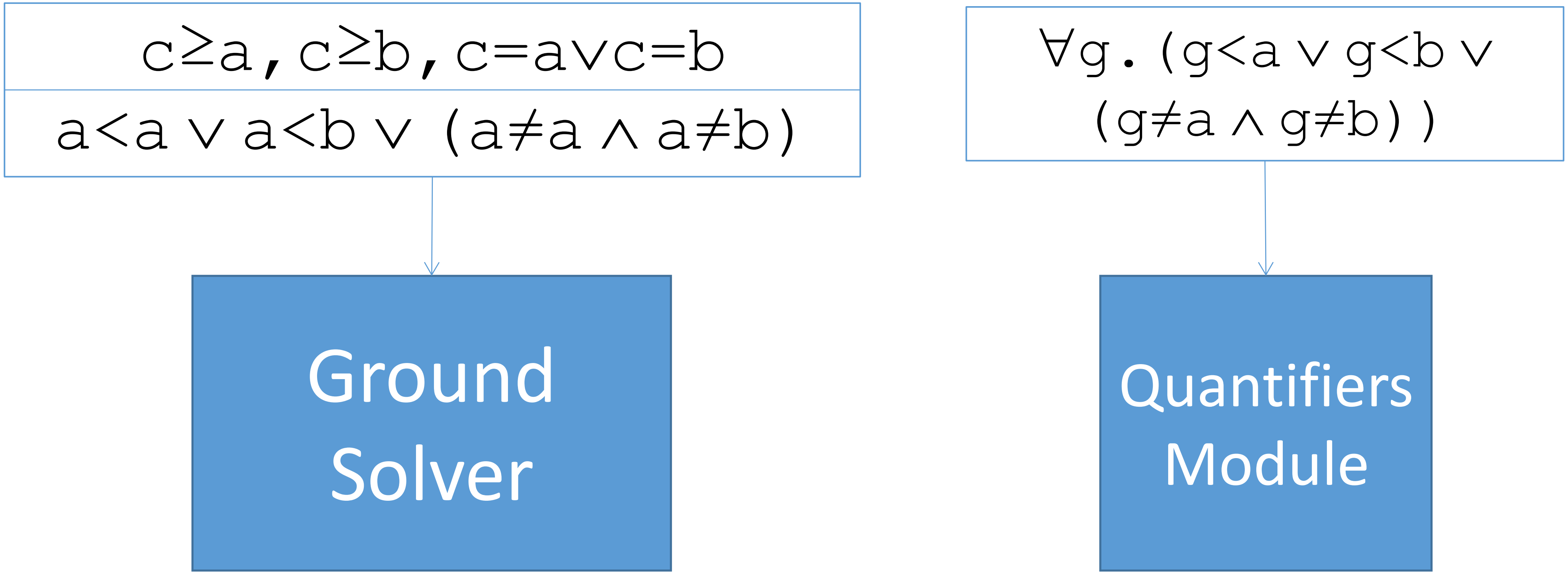


- Take maximal lower bound for c in model M

Counterexample-Guided Instantiation



Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

$c \geq a, c \geq b, c = a \lor c = b$

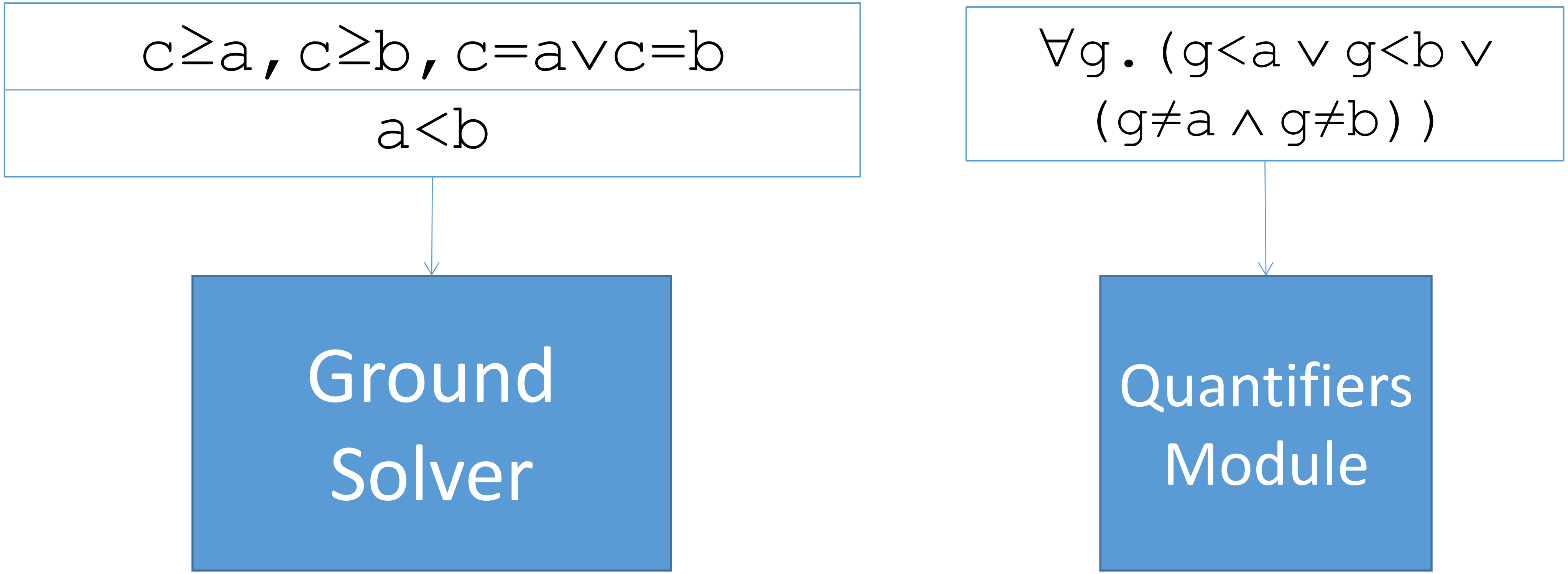
$a < a \lor a < b \lor (a \neq a \land a \neq b)$

$\forall g. (g < a \lor g < b \lor (g \neq a \land g \neq b))$

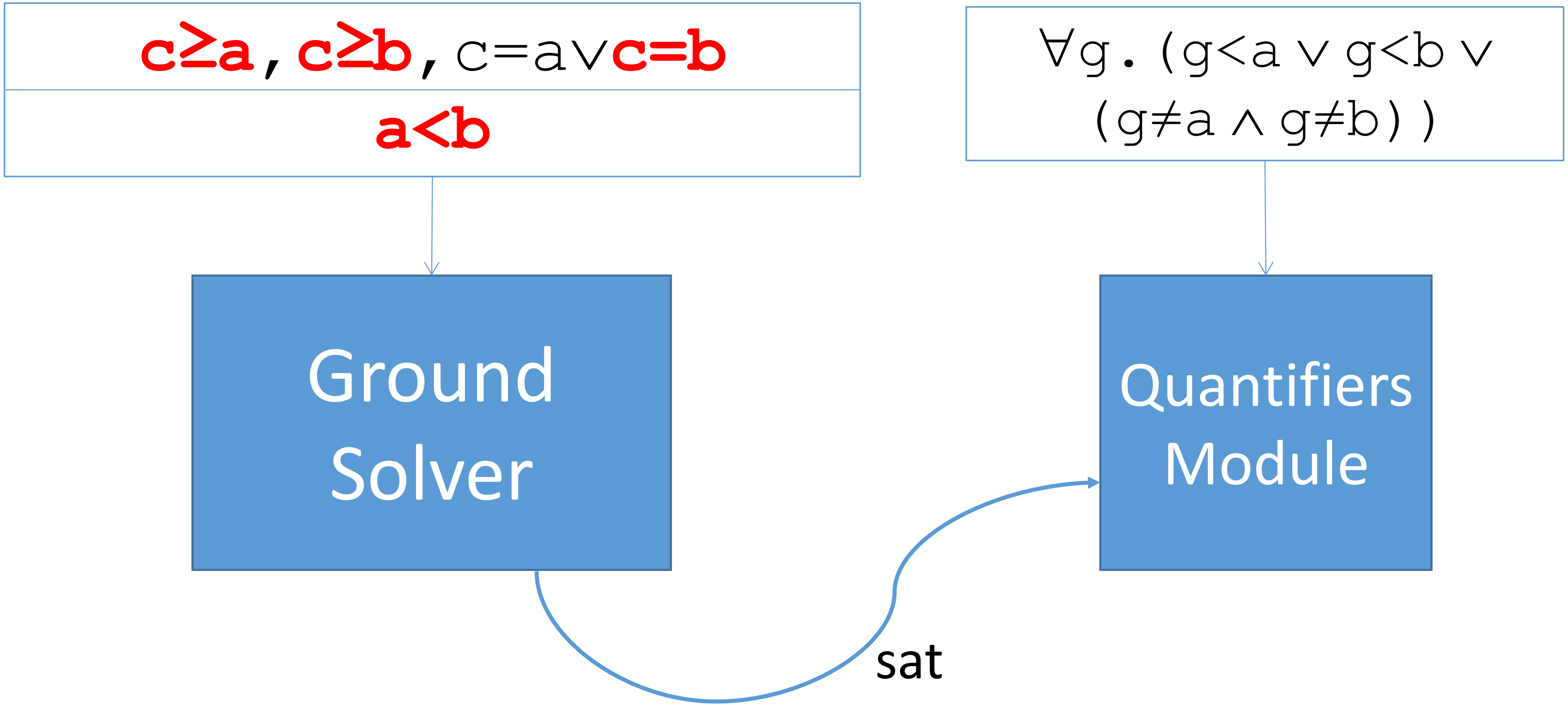
Ground Solver

Quantifiers Module

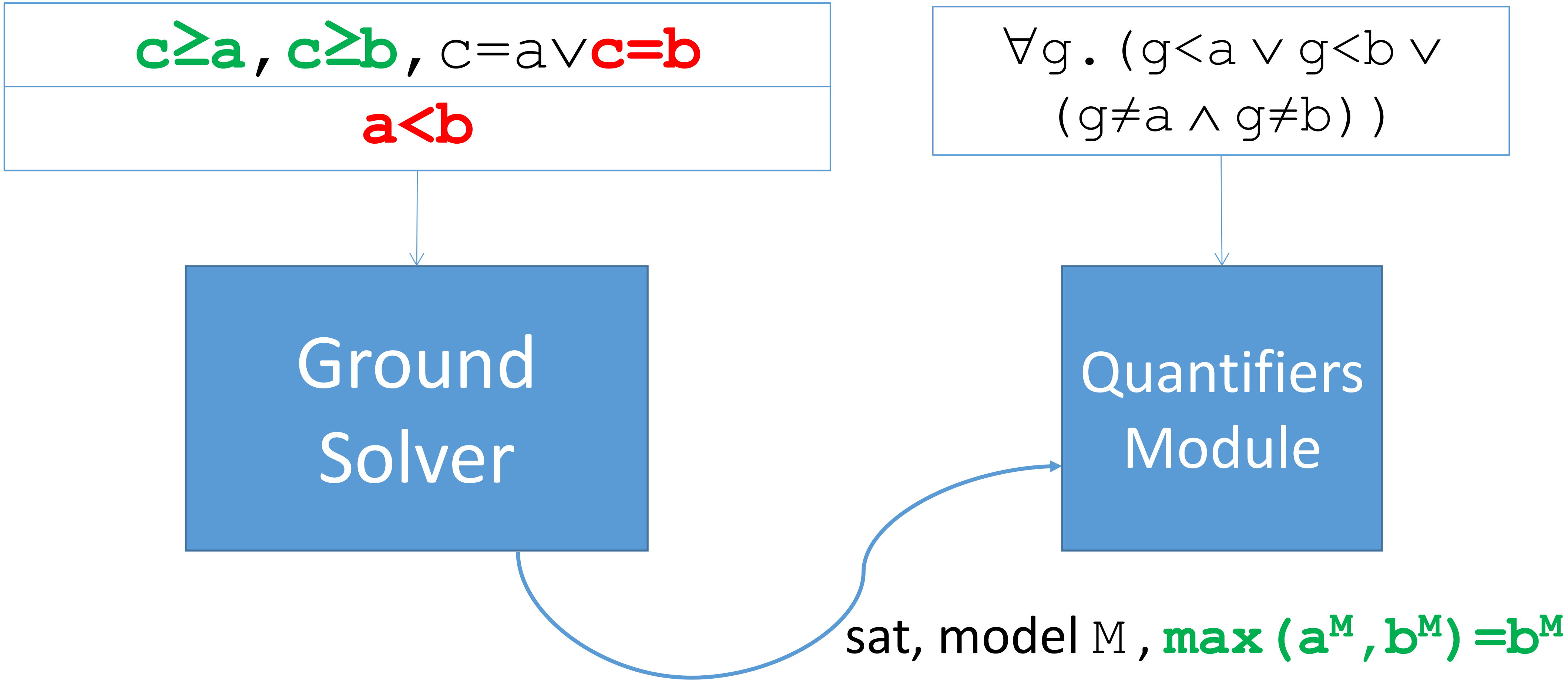
Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

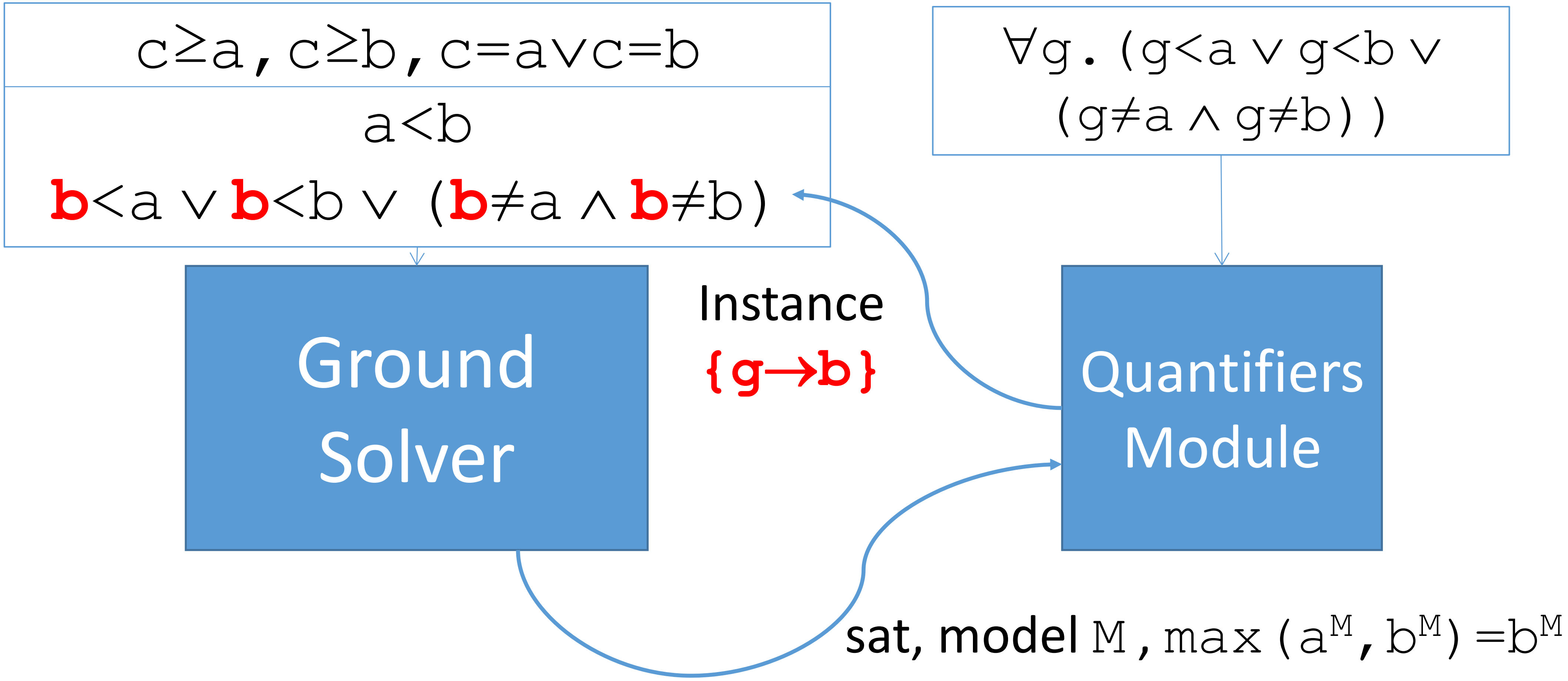


Counterexample-Guided Instantiation

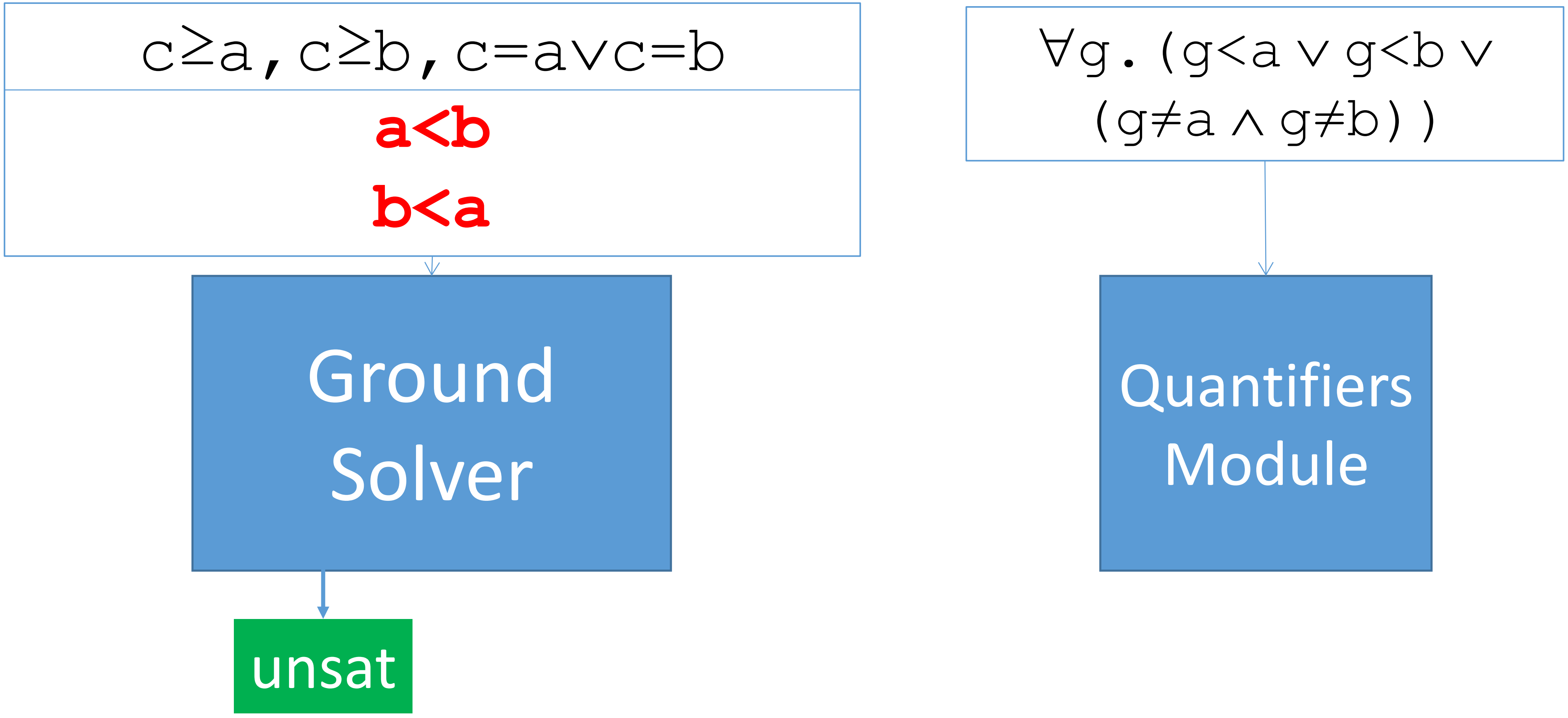


- Take maximal lower bound for c in model M

Counterexample-Guided Instantiation



Counterexample-Guided Instantiation



Results

	keymaera (222)		scholl (371)		tptp (25)		Total (621)	
	#	time	#	time	#	time	#	time
CVC4	**222**	1.5	**348**	1401.1	**25**	0.1	**595**	1402.7
Z3	**222**	1.6	327	360.1	**25**	0.3	574	362.0
VampireZ3	220	51.2	57	393.2	**25**	2.3	302	446.7
Beagle	**222**	377.9	53	577.9	**25**	29.7	300	985.5
Vampire	218	57.8	43	31.1	**25**	1.3	286	90.1
Yices	**222**	0.4	–	0.0	**25**	0.04	247	0.5
ZenonArith	205	13.8	25	452.9	14	0.9	244	467.7
Princess	202	1136.2	0	0.0	**25**	67.4	227	1203.6

Quantified
Linear Real Arithmetic

	psyco (189)		tptp (46)		uauto (155)		sygus (71)		Total (461)	
	#	time	#	time	#	time	#	time	#	time
CVC4	**189**	89.1	**46**	0.3	**155**	1.6	**71**	22.0	**461**	112.9
Z3	183	31.6	**46**	0.6	**155**	1.2	**71**	18.3	455	51.7
Beagle	28	900.0	**46**	48.4	153	343.6	57	617.7	284	1909.7
Princess	13	513.4	**46**	48.0	**155**	201.9	68	418.8	282	1182.1
VampireZ3	4	3.1	36	4.7	**155**	106.3	55	151.8	250	265.9
Vampire	6	196.0	36	2.0	**155**	378.0	46	262.8	243	838.7
ZenonArith	0	0.0	30	1.9	154	15.0	28	1374.6	212	1391.5

Quantified
Linear Integer Arithmetic

Synthesis: Solutions

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

Ground Solver

Quantifiers Module

Synthesis: Solutions

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

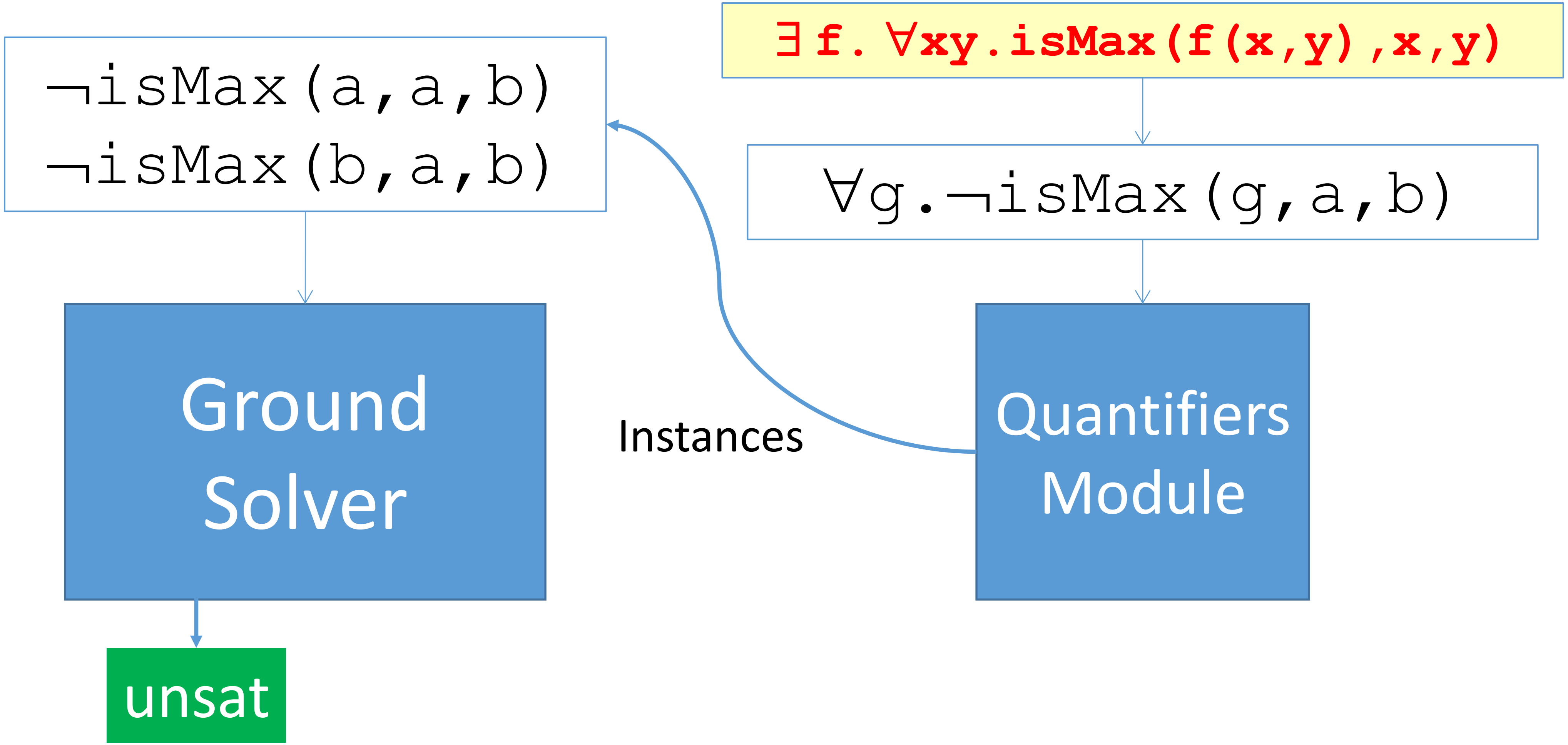
Negate, convert to FO

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

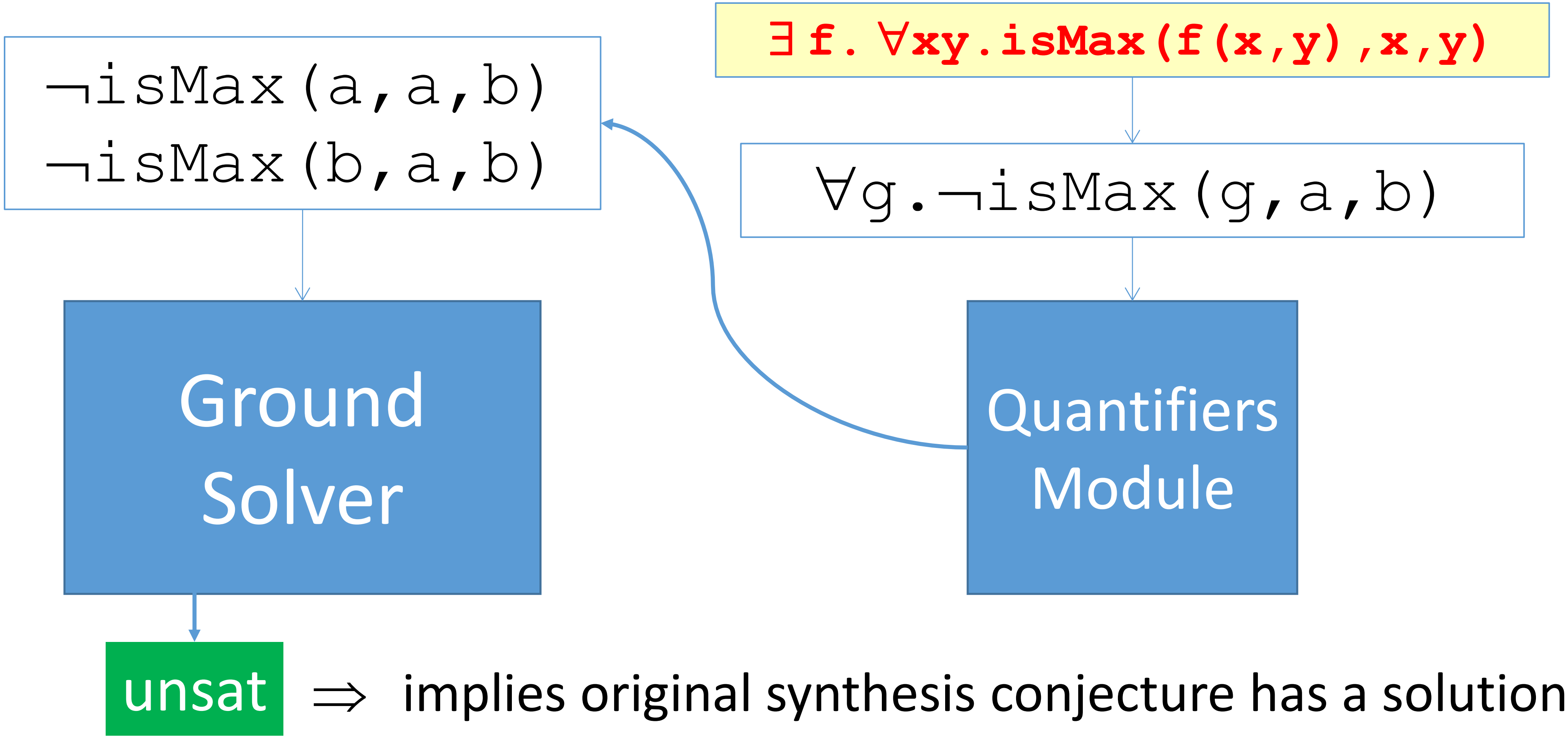
Ground Solver

Quantifiers Module

Synthesis: Solutions

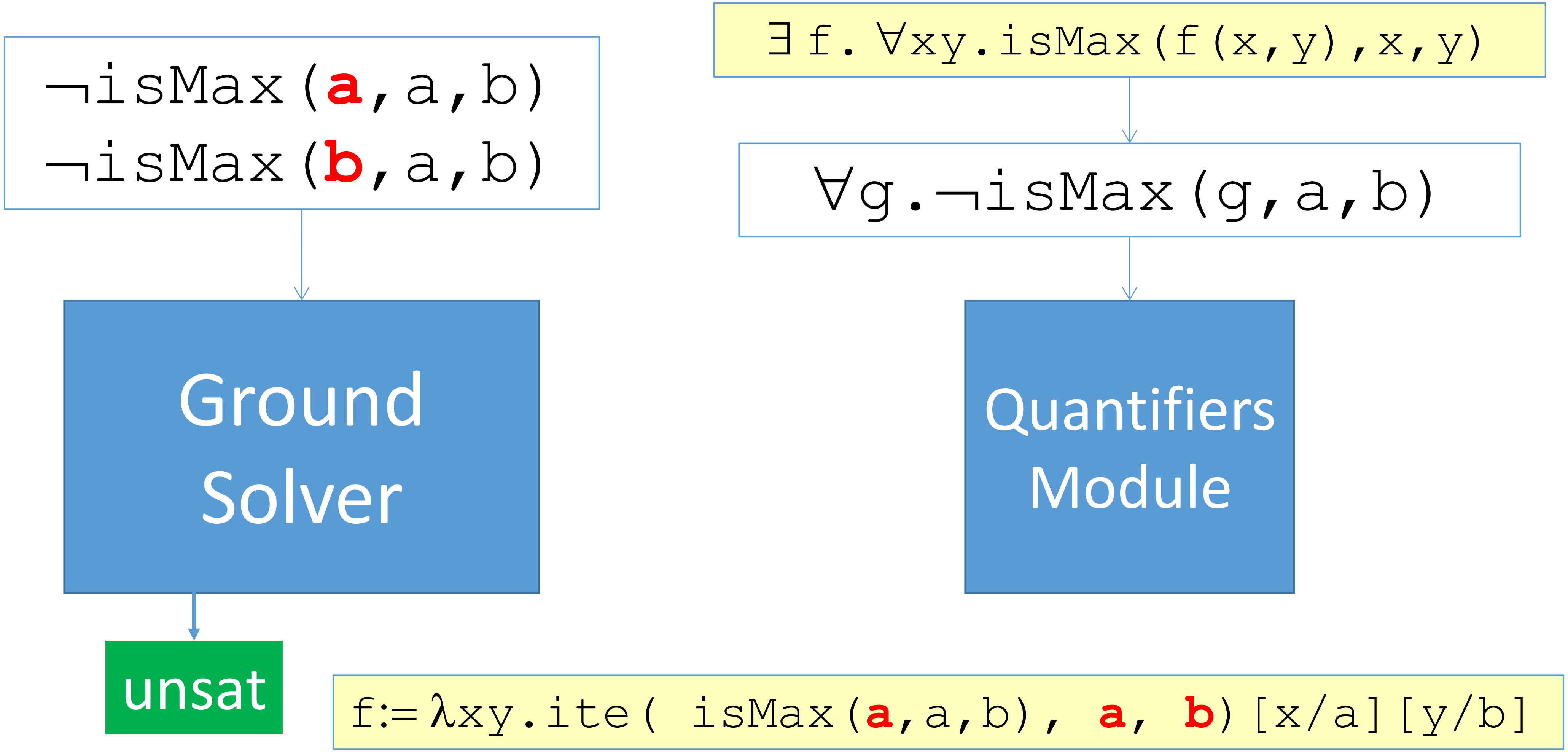


Synthesis: Solutions



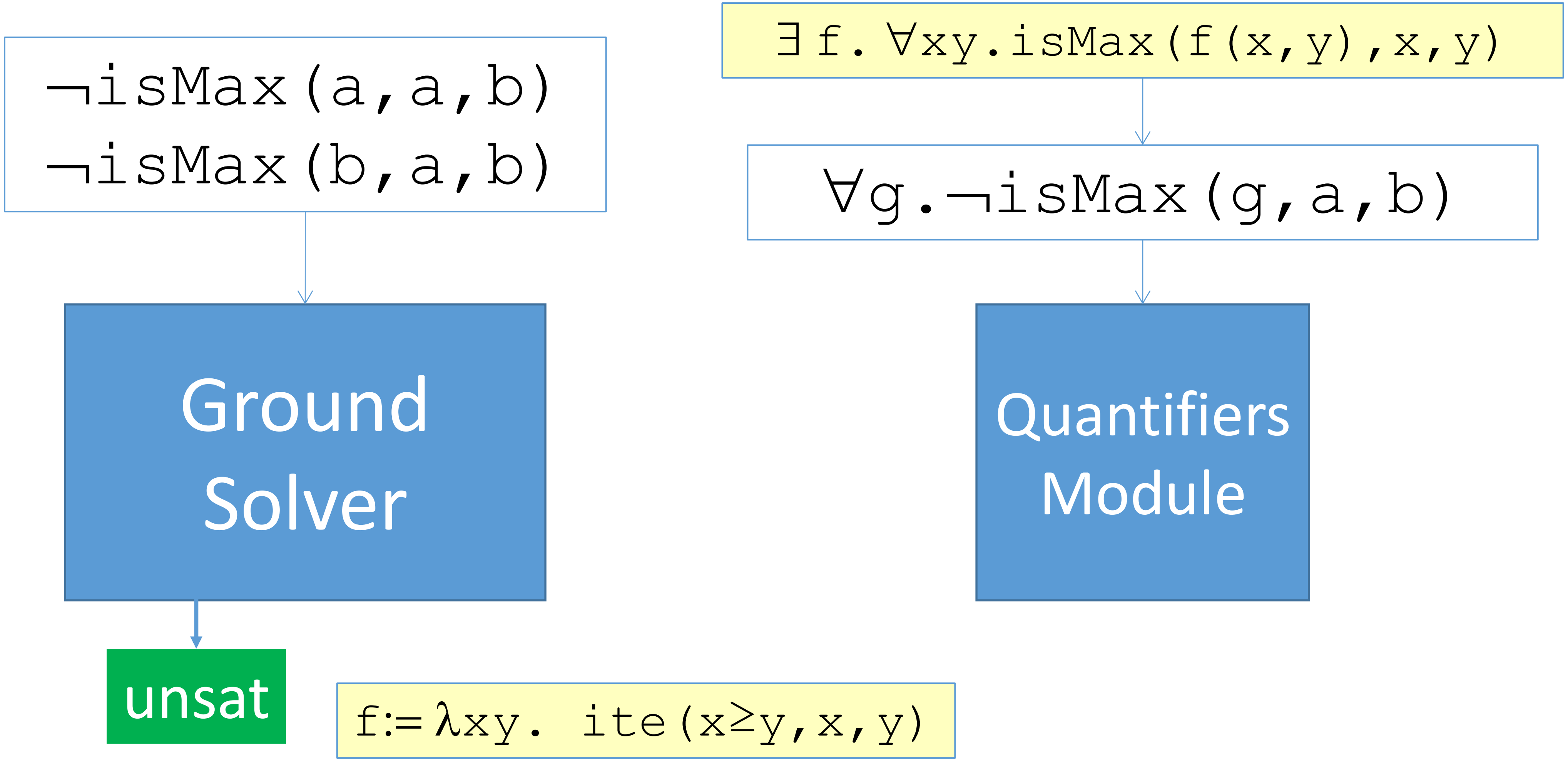
⇒ implies original synthesis conjecture has a solution

Synthesis: Solutions



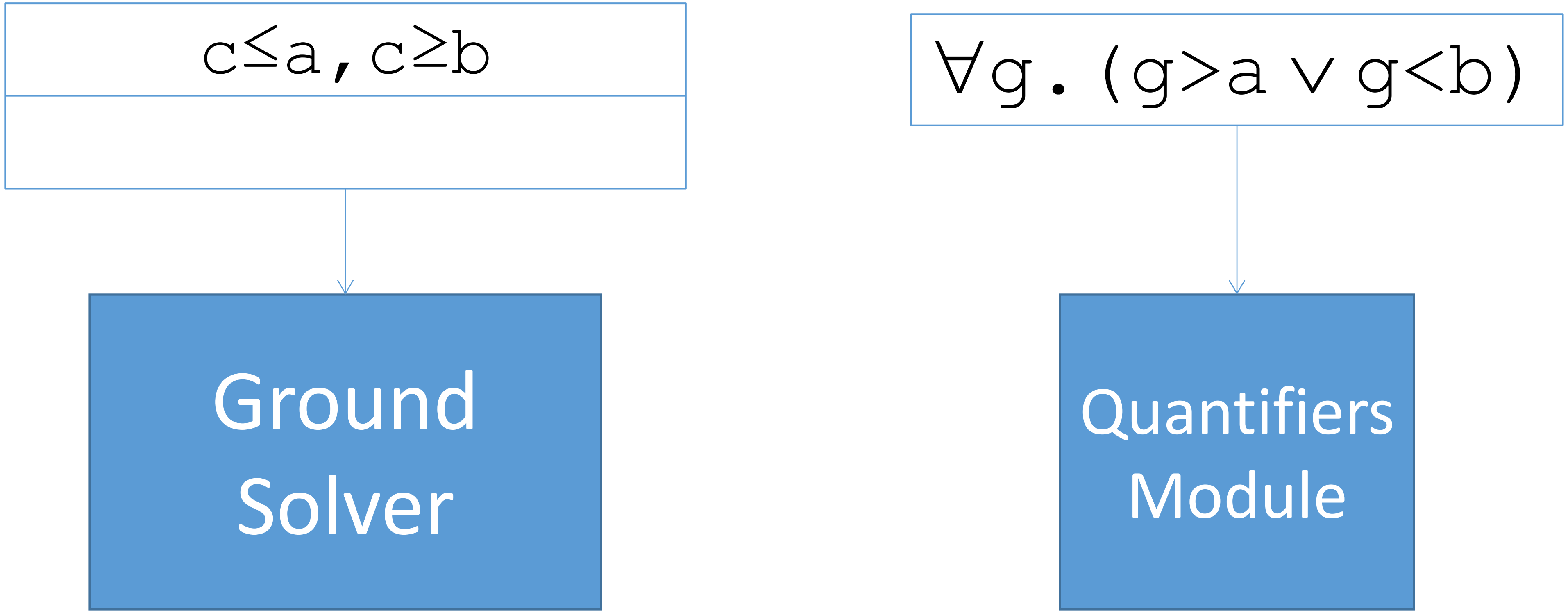
⇒Solution can be extracted from unsatisfiable core of instantiations a/g, b/g

Synthesis: Solutions



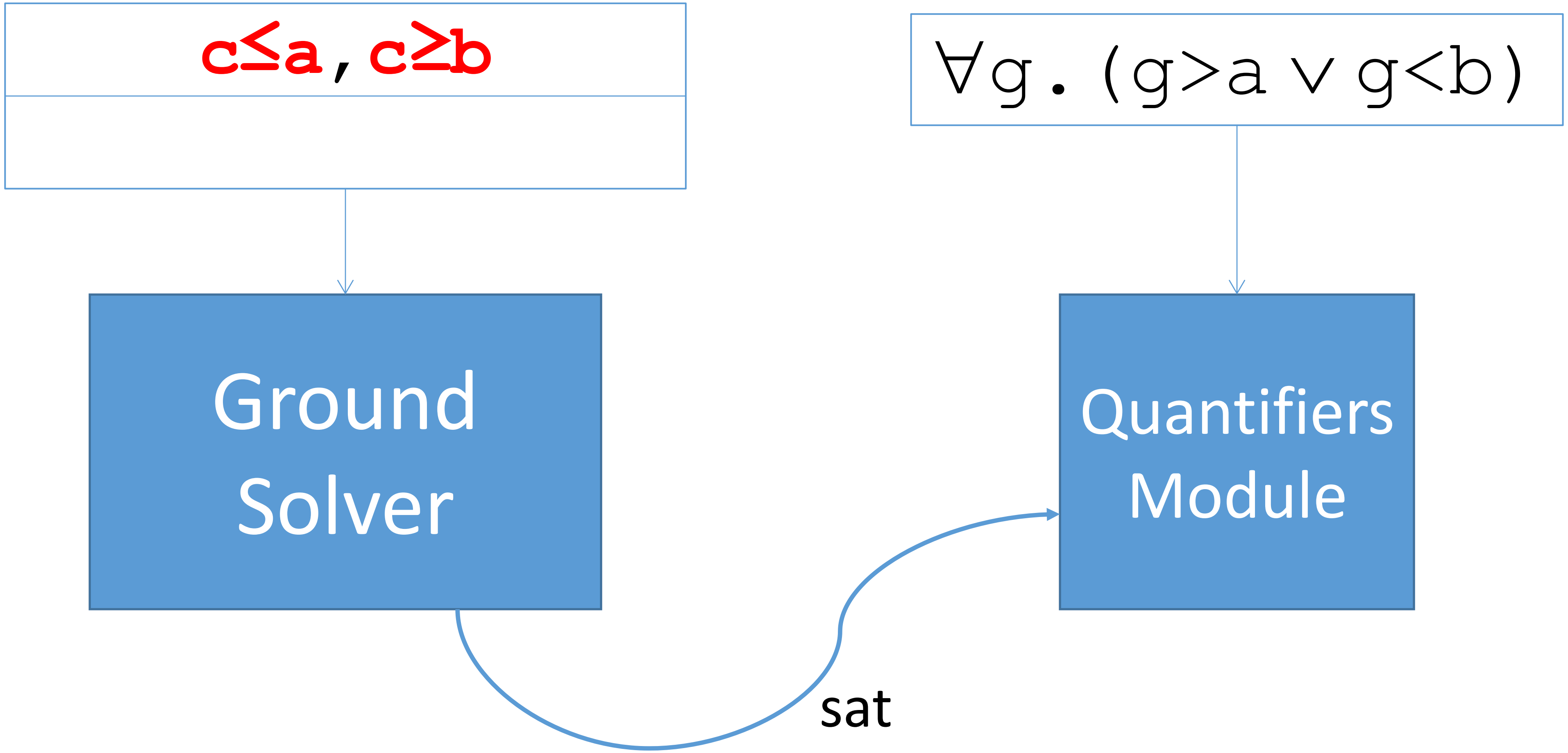
⇒ Desired function, after simplification

Counterexample-Guided Instantiation

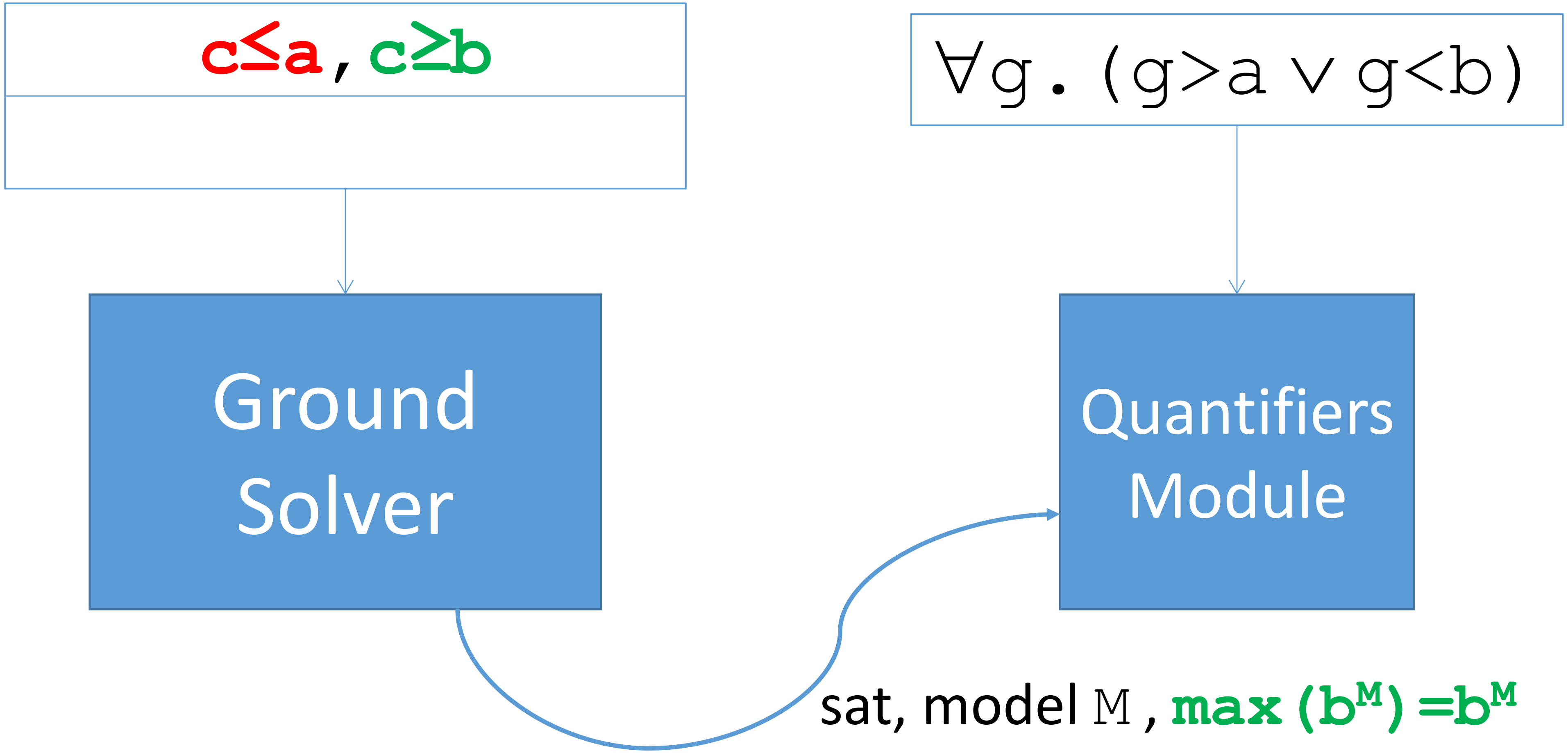


- Consider example: $\forall g. (g>a \vee g<b)$

Counterexample-Guided Instantiation

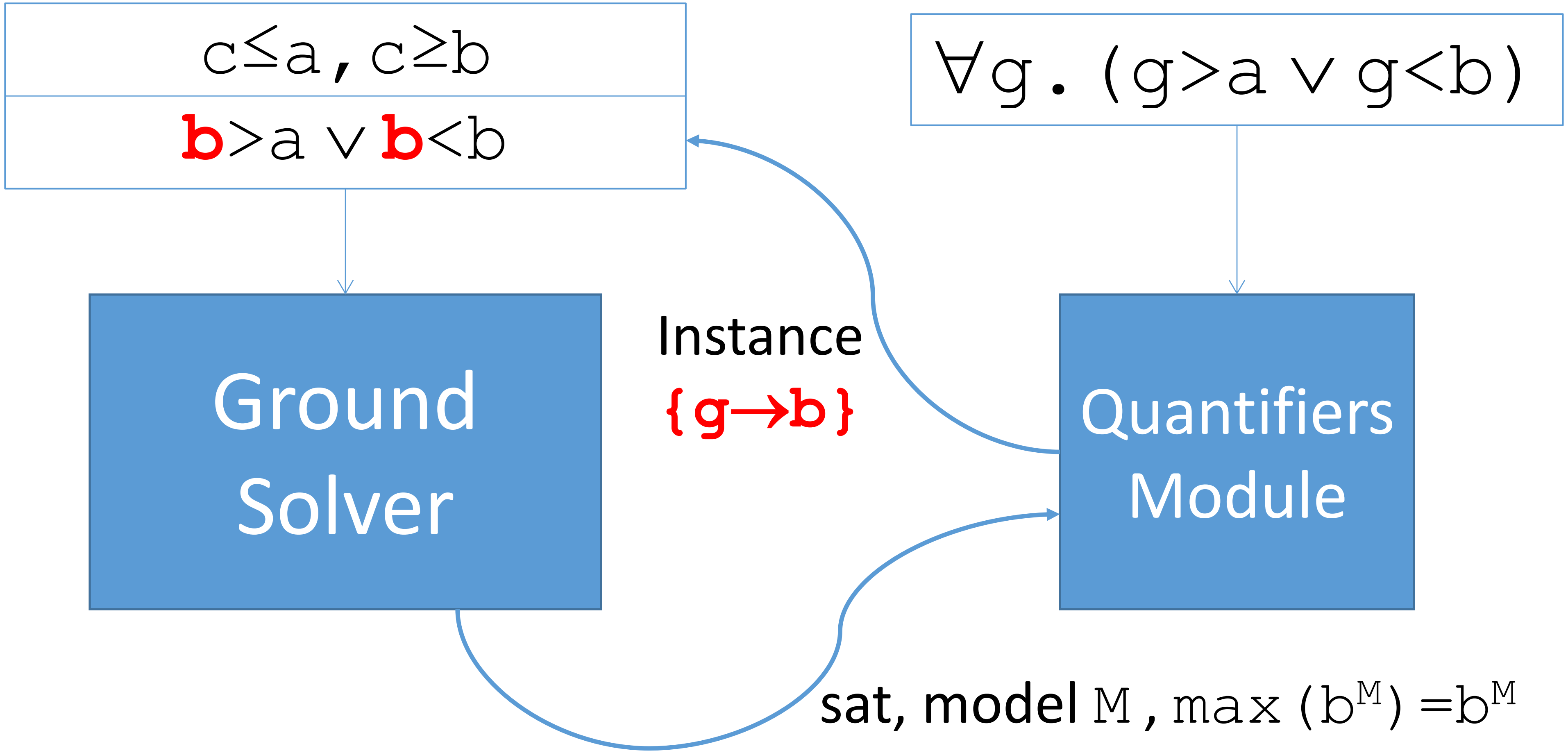


Counterexample-Guided Instantiation

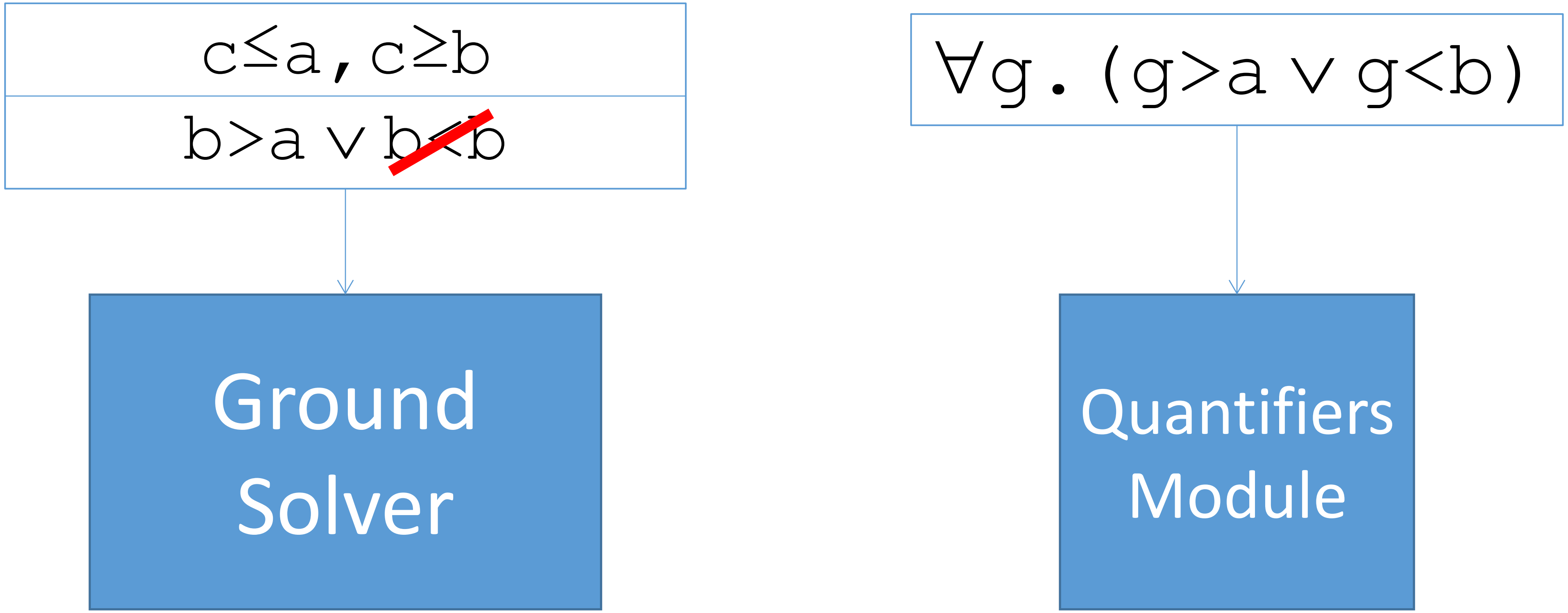


- Take maximal lower bound for c in model M

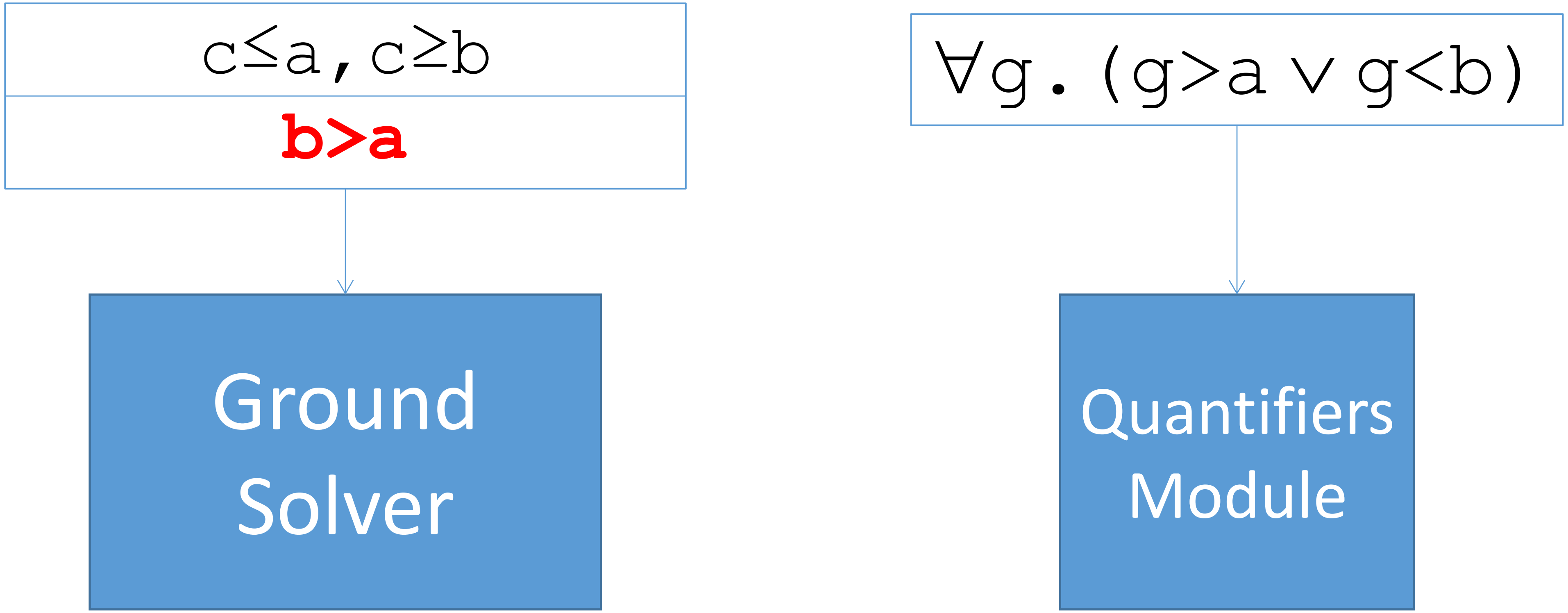
Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

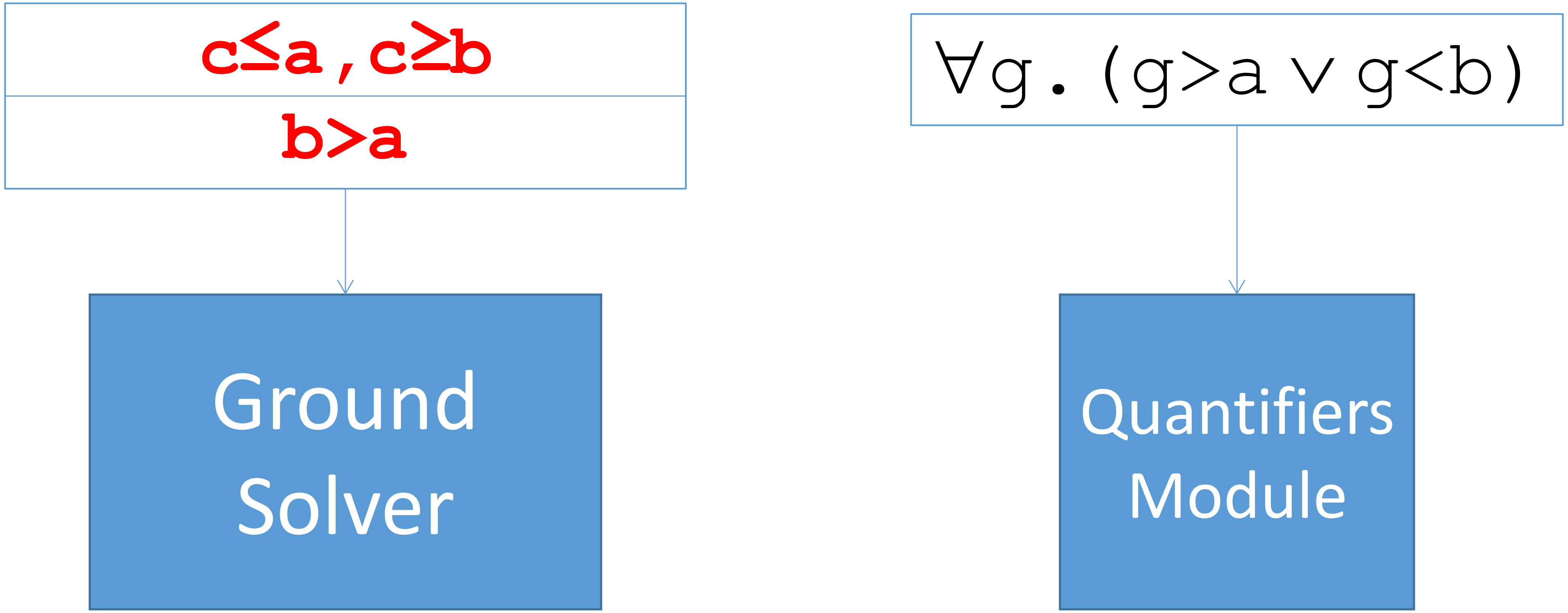


Counterexample-Guided Instantiation



- {b>a} is sat

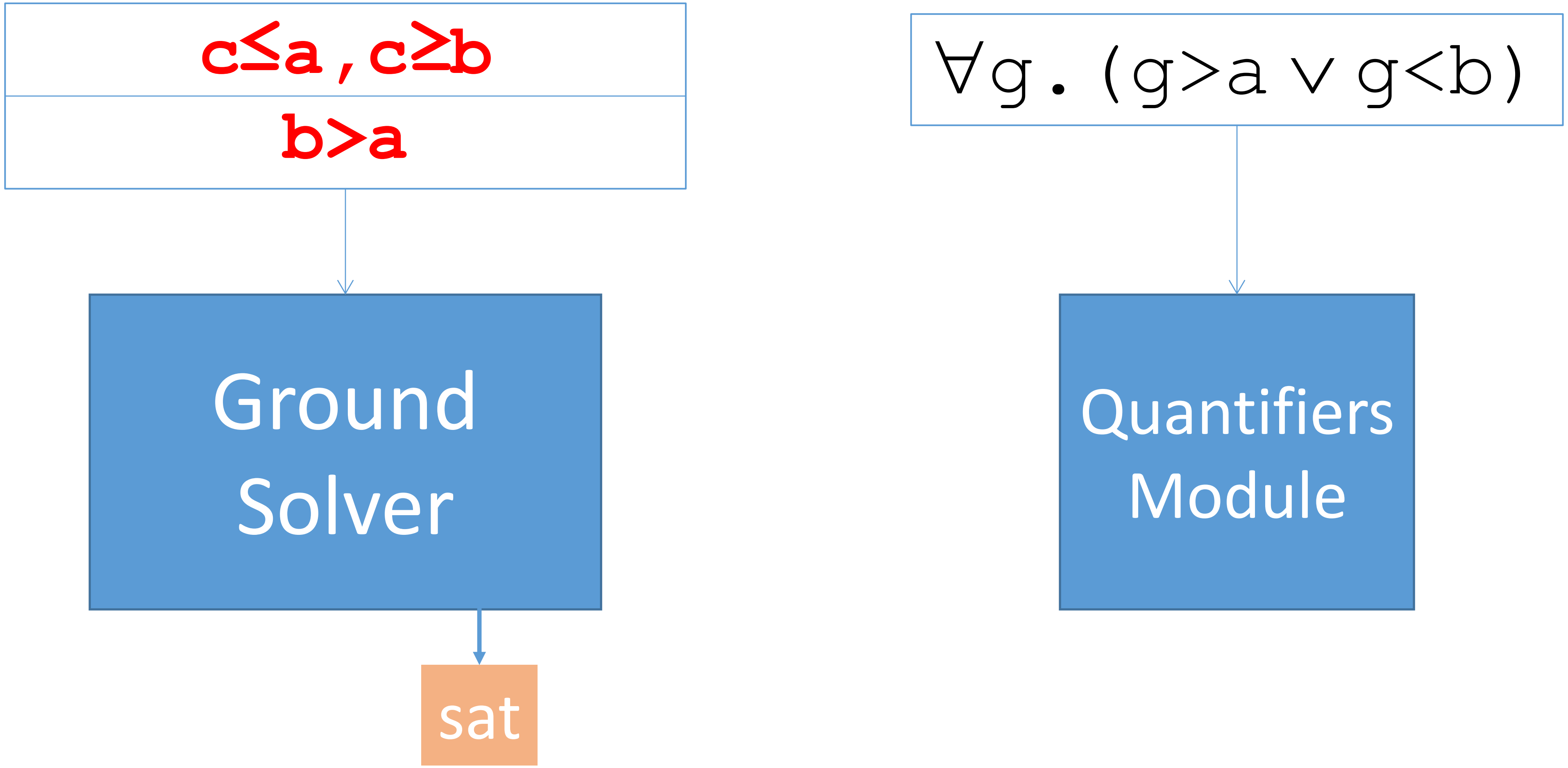
Counterexample-Guided Instantiation



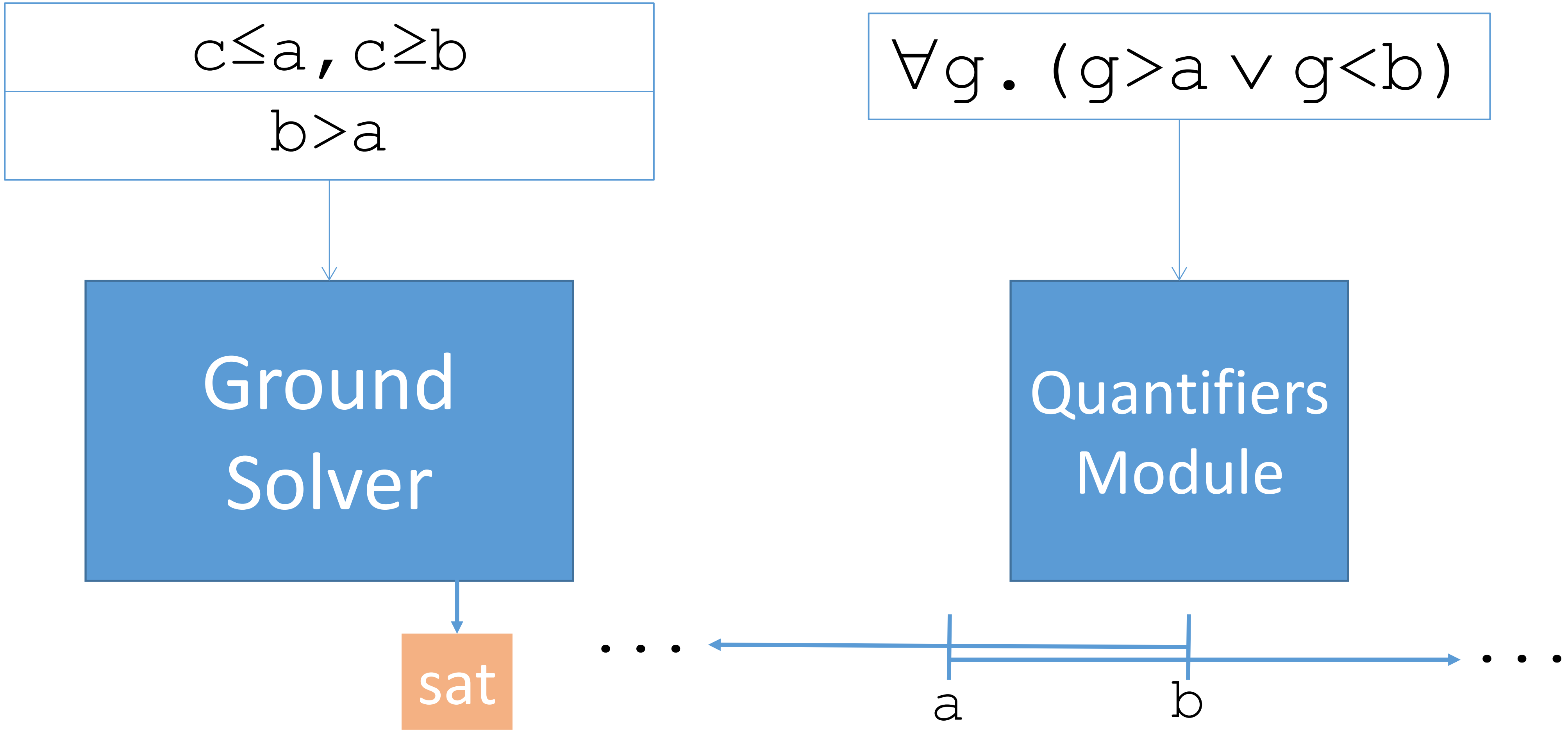
- {b>a} is sat
- ...but {c≤a, c≥b, b>a} is unsat

  ⇒ In other words, there is no model for counterexample c

Counterexample-Guided Instantiation



Counterexample-Guided Instantiation



$\Rightarrow$ All models satisfying b>a also satisfy ∀g. (g>a ∨ g<b)

Counterexample-Guided Instantiation

- For linear real and integer arithmetic:
  - With one quantifier alternation:
    - **Sound** and **complete** (terminating) **[Reynolds/King/Kuncak, in submission 2015]**

      $\Rightarrow$ Procedure enumerates a finite set of instances
  - With arbitrary quantifier alternations:
    - Effective in practice, for both “sat” and “unsat”
  - Supports mixed integer/real arithmetic

Counterexample-Guided Instantiation in CVC4

- Highly competitive for synthesis applications
  - **Won**, GENERAL/LIA divisions of SygusComp 2015
- Applicable to arbitrary quantified formulas as well
  - **Won**, LIA/LRA divisions of SMT COMP 2015
  - **Won**, first-order theorems division of CASC J7
  - 2^{nd} place, first-order theorems division of CASC 25
  - **Won**, first-order non-theorems division of CASC 25

Future Work

- Instantiation-based approaches for $\forall$ in other theories
  - In particular, those that admit QE: datatypes, bitvectors
- Combinations of theories
- Best heuristics for multiple quantifier alternations
- Extend approach for non-linear

Thanks!

- CVC4 is publicly available at:

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

For Multiple Alternations

`∀x∃y∀z. F(x,y,z)`

For Multiple Alternations

$\forall x \exists y \forall z.\ F(x,y,z)$

$G_1 \Rightarrow (\exists e_1) \forall y \exists z.\ \neg F(e_1,y,z)$

For Multiple Alternations

$$\forall x \exists y \forall z.\ F(x,y,z)$$

$$G_1 \Rightarrow (\exists e_1)\forall y \exists z.\ \neg F(e_1,y,z)$$

$$G_2 \Rightarrow (\exists e_2)\forall z.\ F(e_1,e_2,z)$$

For Multiple Alternations

$$\forall x \exists y \forall z.\ F(x,y,z)$$

$$G_1 \Rightarrow (\exists e_1) \forall y \exists z.\ \neg F(e_1,y,z)$$

$$G_2 \Rightarrow (\exists e_2) \forall z.\ F(e_1,e_2,z)$$

$$G_3 \Rightarrow (\exists e_3)\ \neg F(e_1,e_2,e_3)$$

For Multiple Alternations

$$\forall x \exists y \forall z.\ F(x,y,z)$$

$$G_1 \Rightarrow (\exists e_1) \forall y \exists z.\ \neg F(e_1,y,z)$$

$$G_2 \Rightarrow (\exists e_2) \forall z.\ F(e_1,e_2,z)$$

$$G_3 \Rightarrow (\exists e_3)\ \neg F(e_1,e_2,e_3)$$

For Multiple Alternations

$\forall x \exists y \forall z.\ F(x,y,z)$

$G_1 \Rightarrow (\exists e_1) \forall y \exists z.\ \neg F(e_1,y,z)$

$G_2 \Rightarrow (\exists e_2) F(e_1,e_2,t_1) \wedge \ldots \wedge F(e_1,e_2,t_n)$

$G_3 \Rightarrow (\exists e_3)\ \neg F(e_1,e_2,e_3)$

For Multiple Alternations

$\forall x \exists y \forall z.\ F(x,y,z)$

$G_1 \Rightarrow (\exists e_1)\forall y \exists z.\ \neg F(e_1,y,z)$

$G_2 \Rightarrow (\exists e_2) F(e_1,e_2,t_1) \wedge \ldots \wedge F(e_1,e_2,t_n)$

$G_3 \Rightarrow (\exists e_3)\ \neg F(e_1,e_2,e_3)$

For Multiple Alternations

$\forall x \exists y \forall z.\ F(x,y,z)$

$G_1 \Rightarrow (\exists e_1) \neg F(e_1,s_1,t_1) \wedge \ldots \wedge \neg F(e_1,s_m,t_n)$

$G_2 \Rightarrow (\exists e_2) F(e_1,e_2,t_1) \wedge \ldots \wedge F(e_1,e_2,t_n)$

$G_3 \Rightarrow (\exists e_3)\ \neg F(e_1,e_2,e_3)$

For Multiple Alternations

$$\forall x \exists y \forall z.\ F(x,y,z)$$

$$G_1 \Rightarrow (\exists e_1)\, \neg F(e_1, s_1, t_1) \wedge \ldots \wedge \neg F(e_1, s_m, t_n)$$

$$G_2 \Rightarrow (\exists e_2)\, F(e_1, e_2, t_1) \wedge \ldots \wedge F(e_1, e_2, t_n)$$

$$G_3 \Rightarrow (\exists e_3)\ \neg F(e_1, e_2, e_3)$$