A Taste of CVC4

Part 2: Quantified Formulas

Cesare Tinelli
Andrew Reynolds
Clark Barrett

The University of Iowa
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
NYU

Quantified Formulas in SMT

$$\underbrace{\forall \mathtt{x} . \mathtt{P}(\mathtt{x})}$$

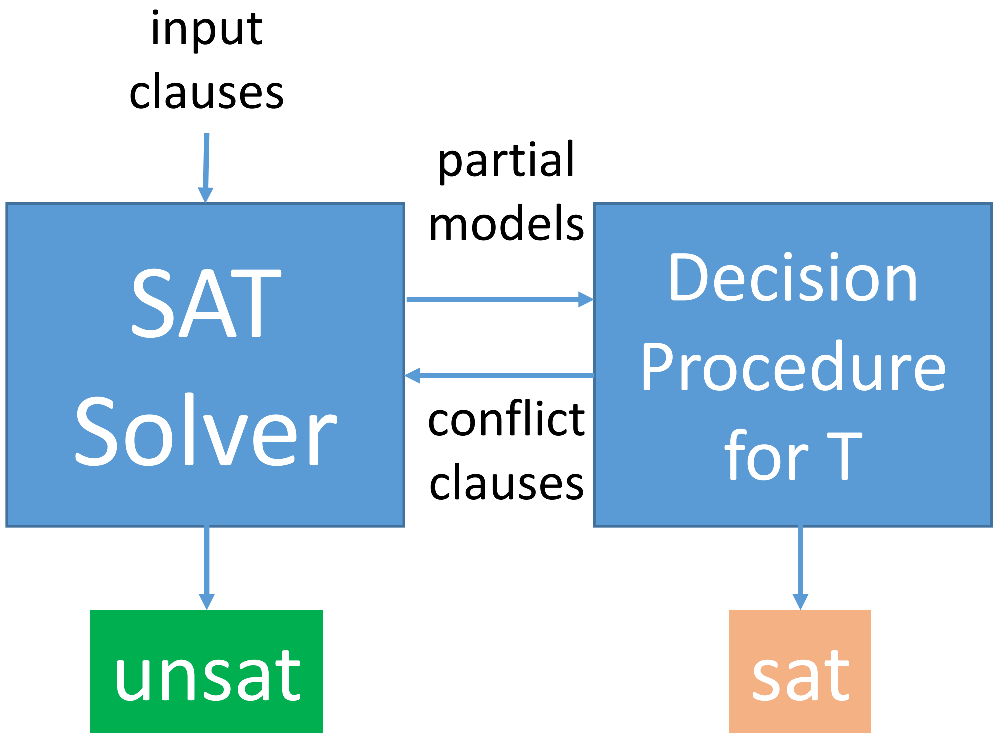
$\mathtt{P}$ is true for all $\mathtt{x}$, where $\mathtt{P}$ is a formula involving some background theory

- Satisfiability problem is undecidable in general
- $\forall$ are critical for applications:
  - Automated Theorem Proving
  - Software/Hardware verification
  - Synthesis, planning, …
- $\forall$ are handled in SMT solvers by a variety of techniques:
  - Complete techniques for certain fragments
  - Heuristic techniques for the general case

Overview

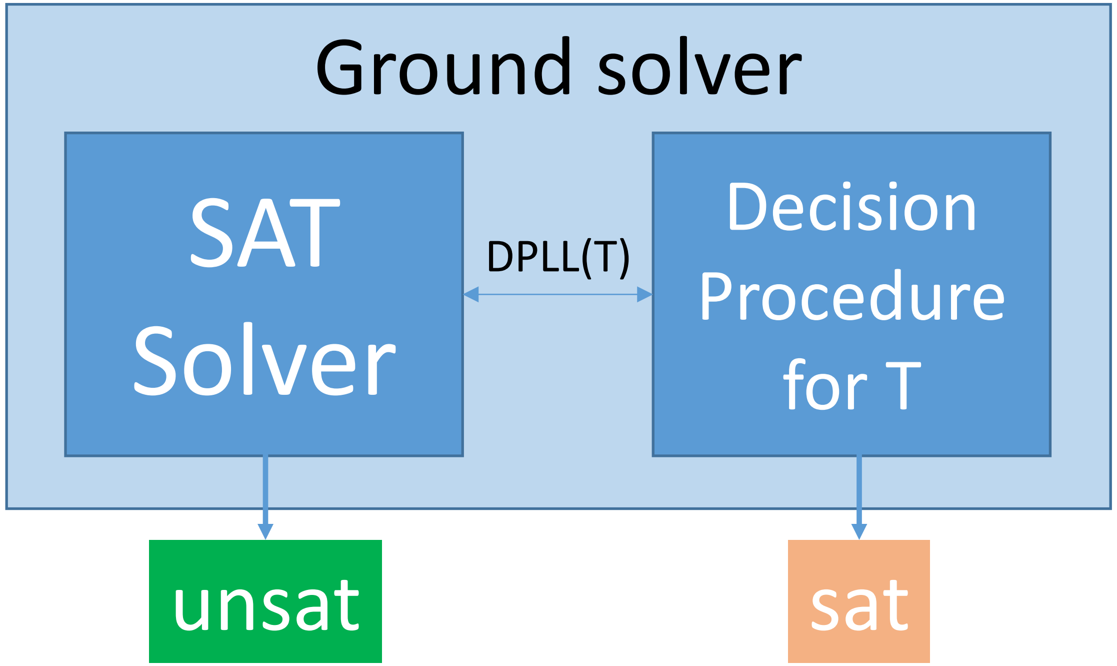
- How do we extend SMT solvers for quantified formulas?
- Quantifier Instantiation in CVC4:
  - Heuristic (E-matching)
  - Model-based
  - Conflict-based
- More advanced techniques in CVC4:
  - Finite Model Finding
  - Function synthesis

DPLL(T)-based SMT Solver



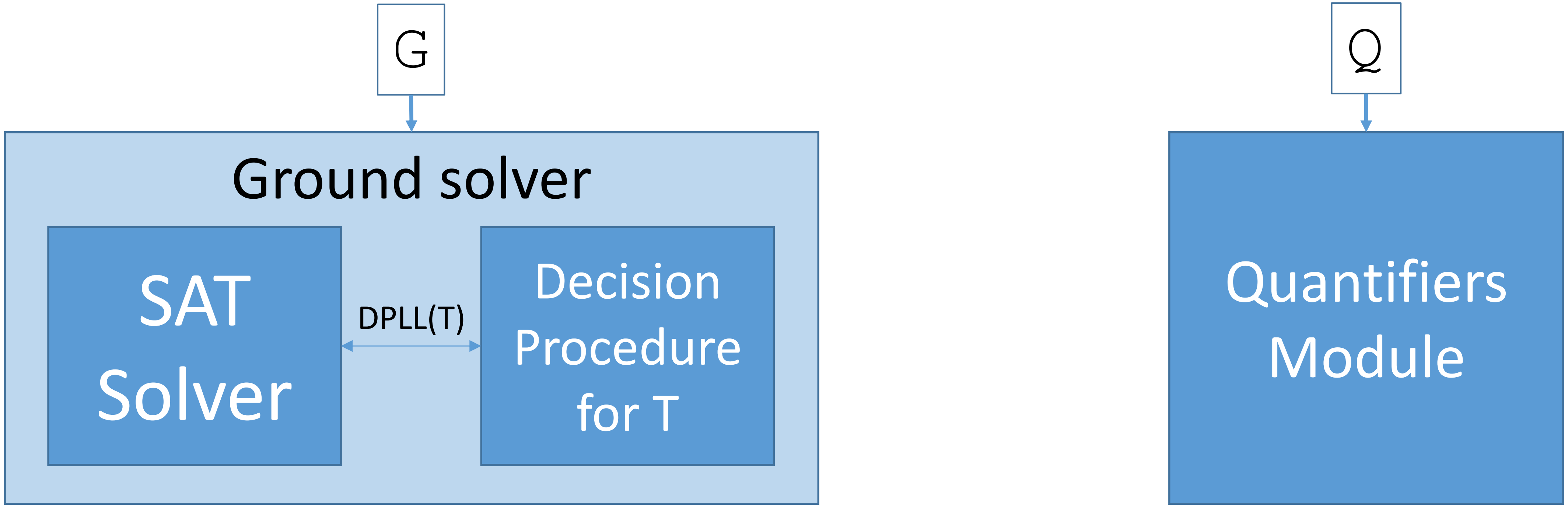
- DPLL(T)-based SMT solver
  - SAT solver maintains a set of propositional clauses
  - Decision Procedure for T determines satisfiability of conjunctions of T-literals

DPLL(T)-based SMT Solver



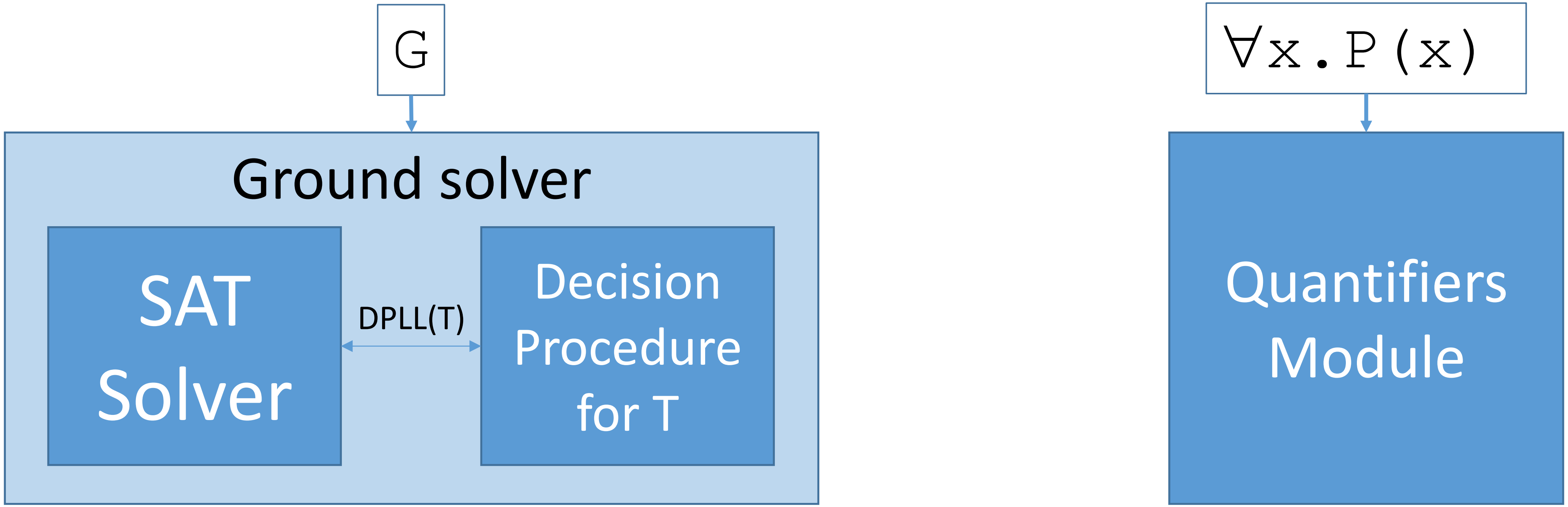
- Ground solver = SAT solver + Decision Procedure for T

DPLL(T) + Quantifiers



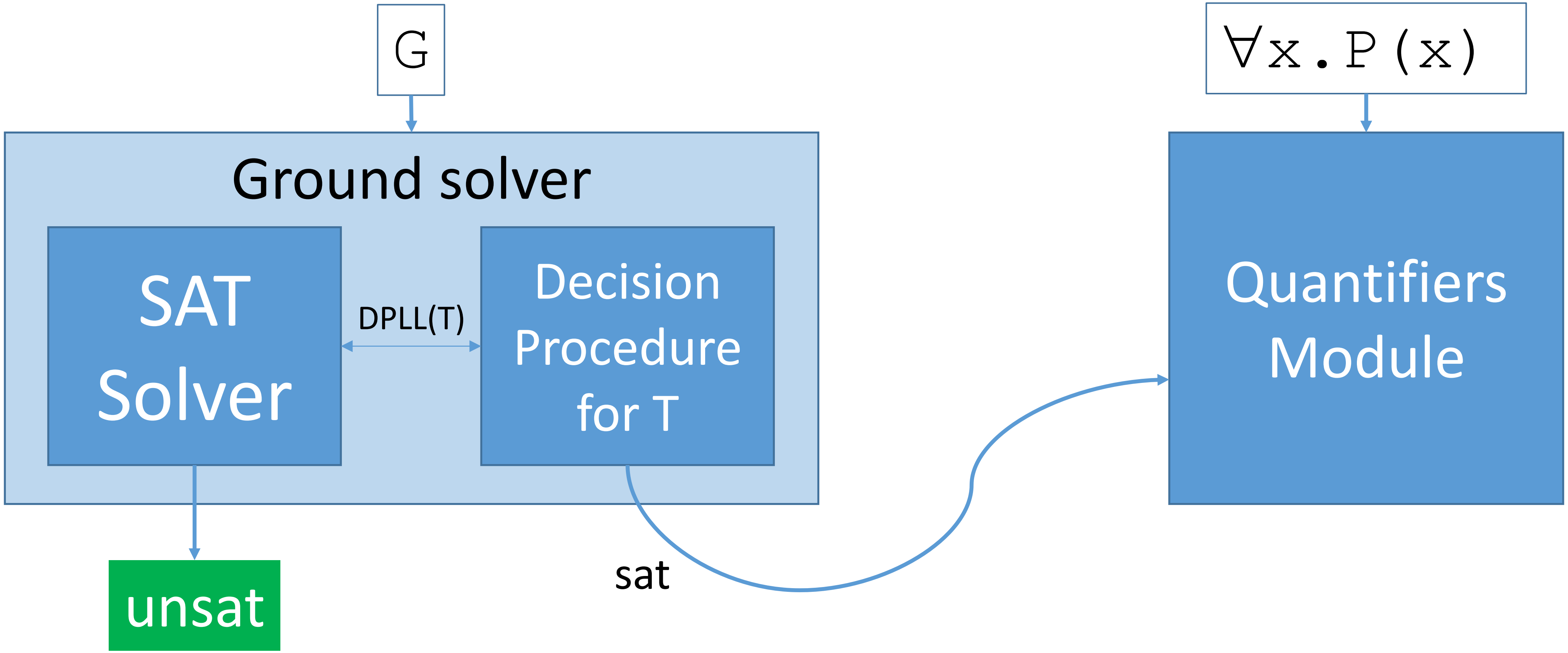
- SMT solver consists of:
  - Ground solver maintains a set of ground (quantifier-free) constraints $\mathbb{G}$
  - Quantifiers Module maintains a set of quantified formulas $\mathbb{Q}$

DPLL(T) + Quantifier Instantiation



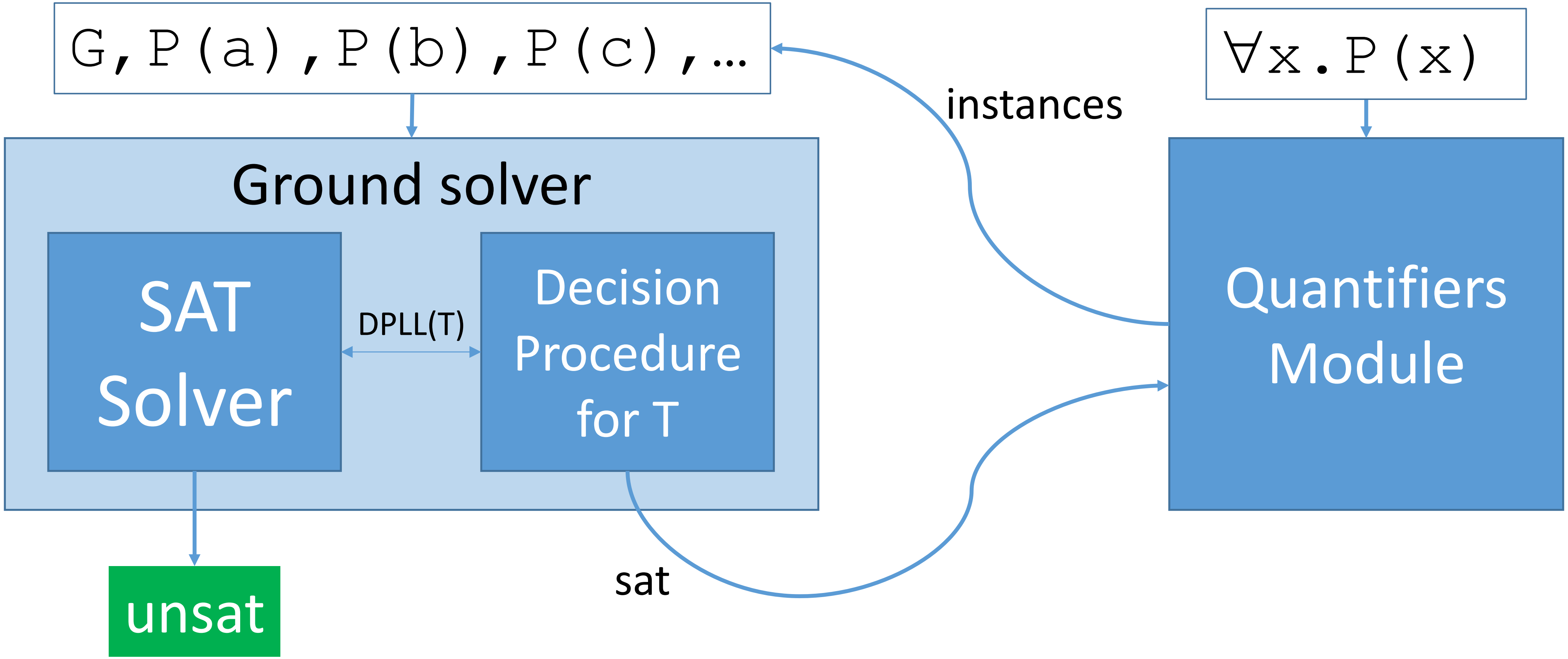
- Primary technique for quantifiers in this talk: **Quantifier Instantiation**

DPLL(T) + Quantifier Instantiation



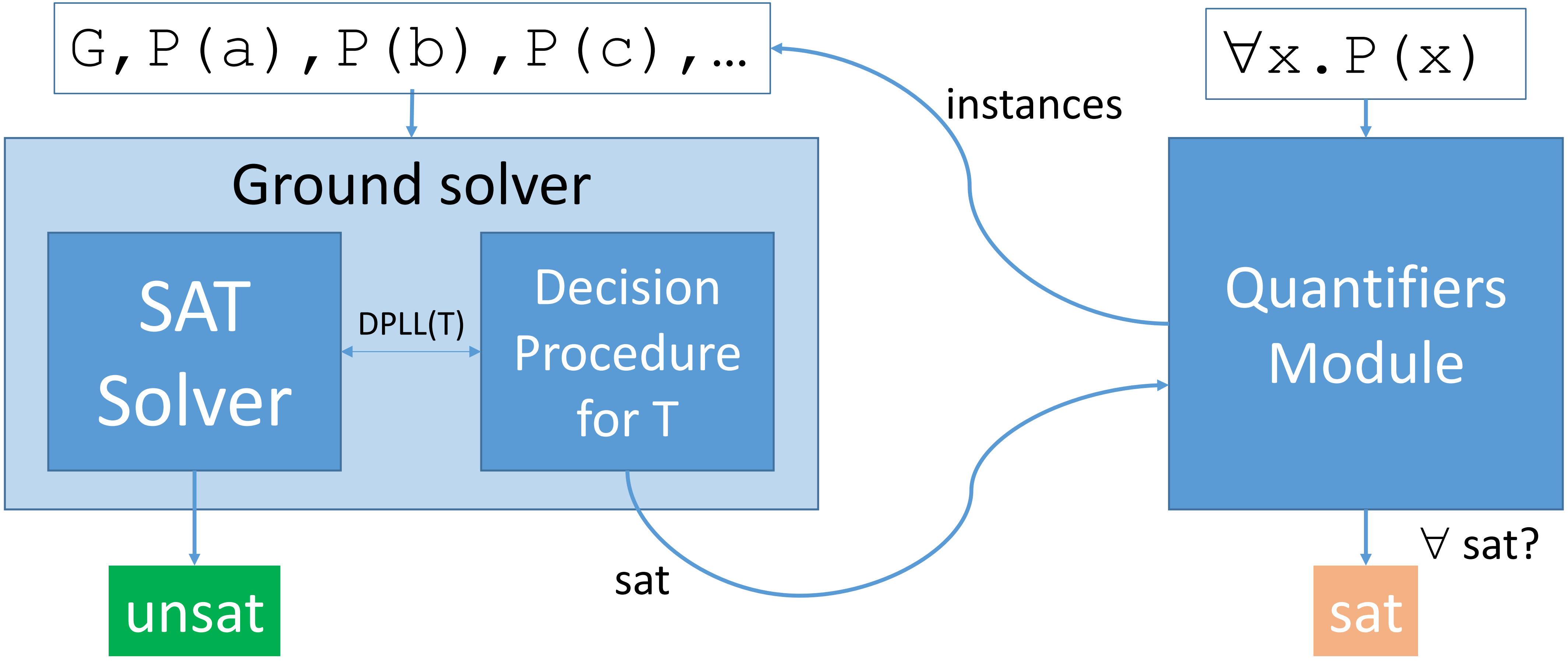
- If G is T-satisfiable, invoke quantifiers module

DPLL(T) + Quantifier Instantiation



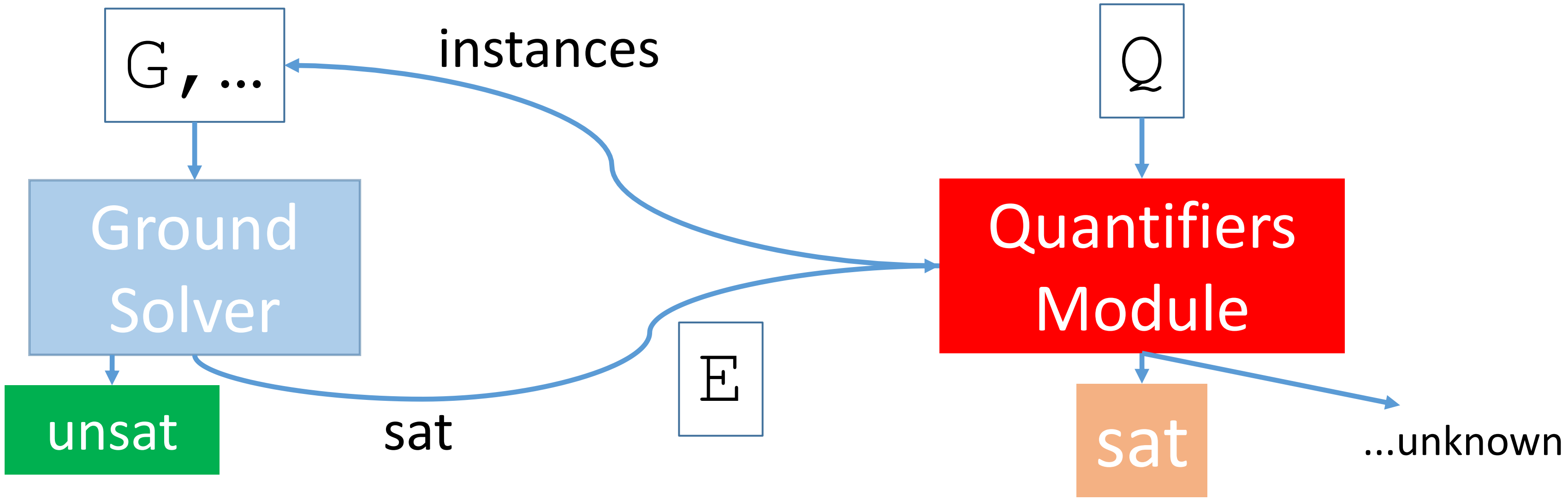
- Add instances of axioms to G

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)`?)

Quantifiers Module: Overview



- Inputs:
  - Set of ground formulas $\mathtt{G}$
- Outputs:
  - "G is T-unsat", or
  - "G is T-sat", set of literals $\mathtt{E} \models_p \mathtt{G}$

- Inputs:
  - Set of ground T-literals $\mathtt{E}$
  - Set of quantified T-formulas $\mathtt{Q}$
- Outputs:
  - "$\mathtt{E} \wedge \mathtt{Q}$ is T-sat"
  - Set of instances of $\mathtt{Q}$ to add to $\mathtt{G}$
  - …"unknown" (give up)

Quantifier Instantiation : Design Decisions

- When do we invoke it?
  - Eagerly, during the DPLL(T) search **[deMoura/Bjorner CAV07]**, or
  - Lazily, only after ground solver answers “sat”

Quantifier Instantiation : Design Decisions

- When do we invoke it?
  - Eagerly, during the DPLL(T) search **[deMoura/Bjorner CAV07]**, or
  - Lazily, only after ground solver answers "sat"
- Which instances do we add?
  - …

Quantifier Instantiation : Design Decisions

- When do we invoke it?
  - Eagerly, during the DPLL(T) search **[deMoura/Bjorner CAV07]**, or
  - Lazily, only after ground solver answers “sat”
- Which instances do we add?
  - …
- Can we terminate?

  i.e. can we ever answer “sat”?

Quantifier Instantiation : in CVC4

- When do we invoke it?
  - Eagerly, during the DPLL(T) search [deMoura/Bjorner CAV09], or
  - Lazily, only after ground solver answers "sat"
- Which instances do we add?
  - …
- Can we terminate?

  i.e. can we ever answer "sat"?

Quantifier Instantiation : in CVC4

- When do we invoke it?
  - Eagerly, during the DPLL(T) search [deMoura/Bjorner CAV09], or
  - Lazily, only after ground solver answers "sat"
- Which instances do we add?
  - E-matching **[Detlefs et al 03]**
  - Model-based quantifier instantiation **[Ge/de Moura CAV09]**
  - Conflict-based quantifier instantiation **[Reynolds et al FMCAD14]**
- Can we terminate?

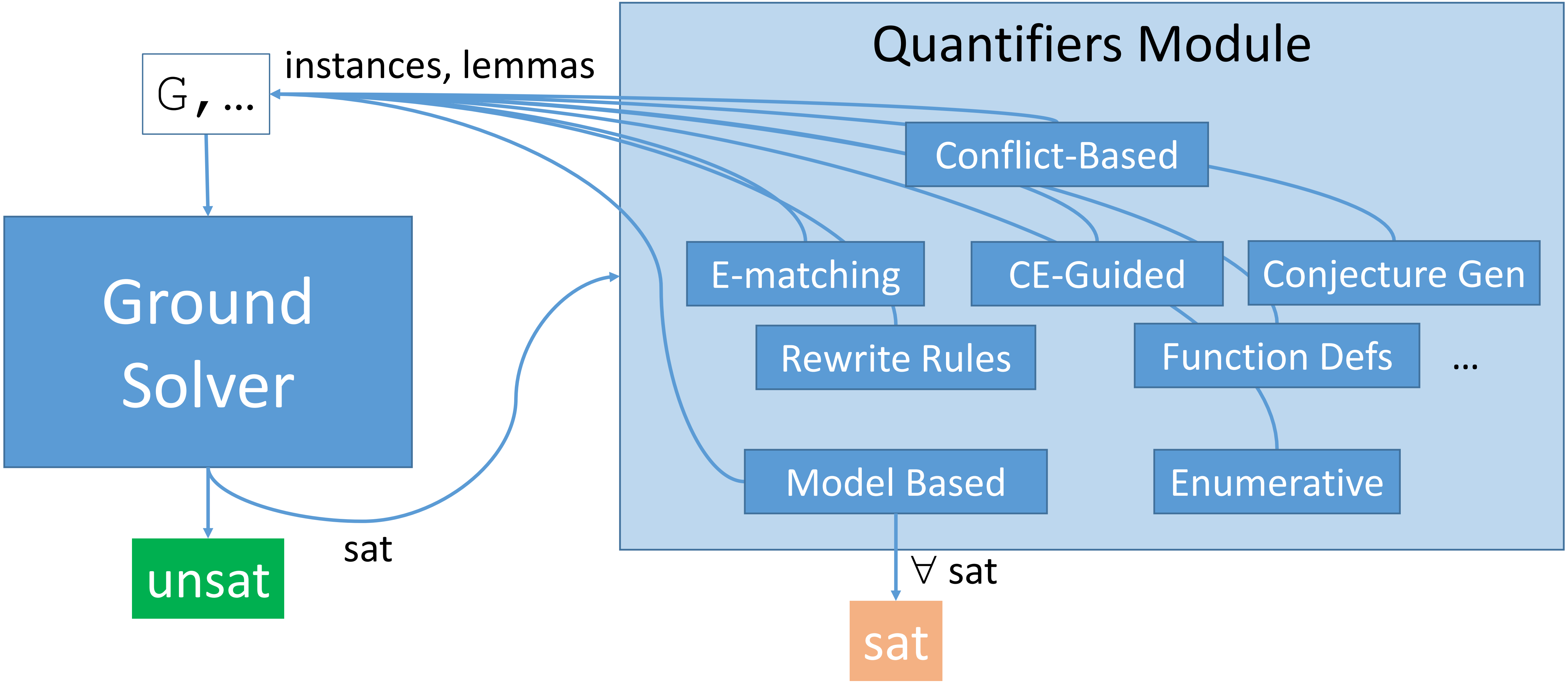
  i.e. can we ever answer "sat"?

Quantifier Instantiation : in CVC4

- When do we invoke it?
  - Eagerly, during the DPLL(T) search [deMoura/Bjorner CAV09], or
  - Lazily, only after ground solver answers "sat"
- Which instances do we add?
  - E-matching **[Detlefs et al 03]**
  - Model-based quantifier instantiation **[Ge/de Moura CAV09]**
  - Conflict-based quantifier instantiation **[Reynolds et al FMCAD14]**
- Can we terminate?

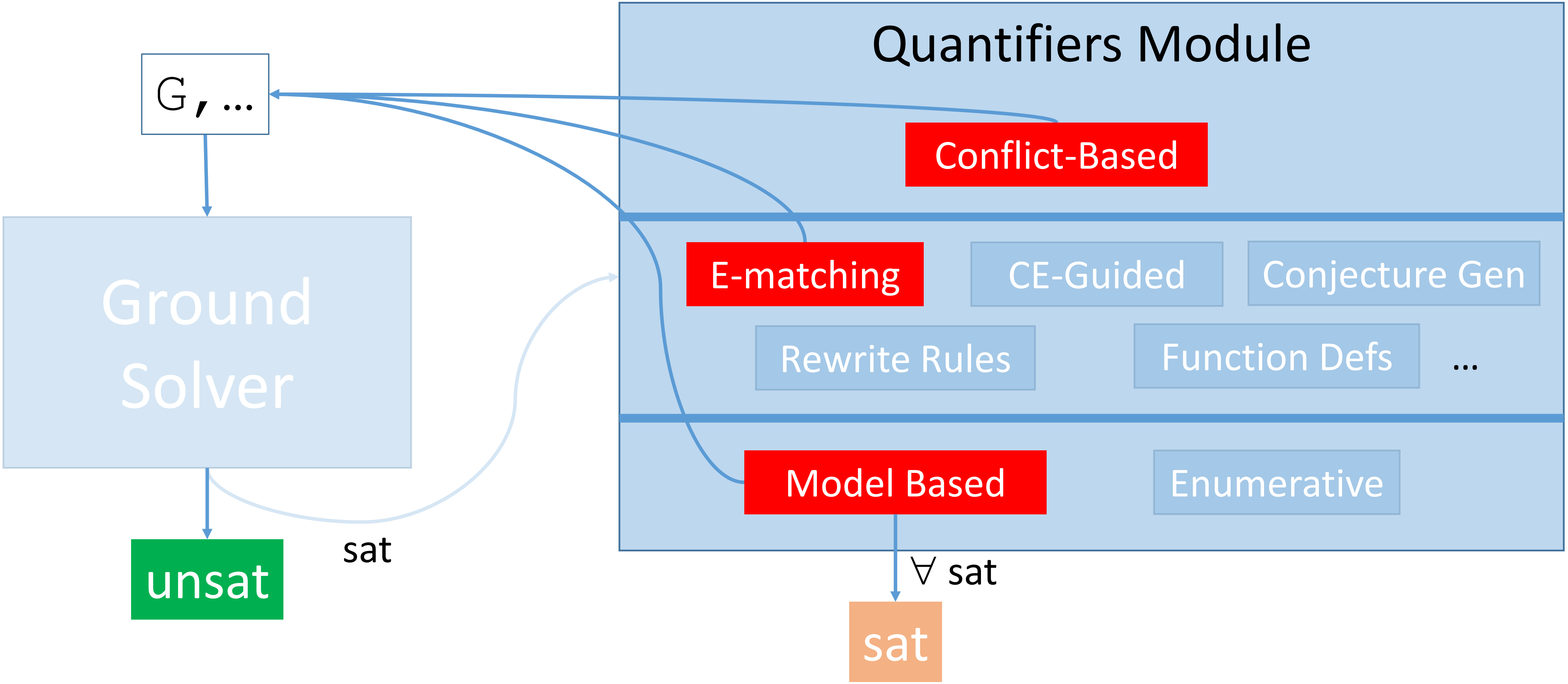
  i.e. can we ever answer "sat"?
  - Finite Model Finding **[Reynolds et al CADE13]**
  - Instantiation for linear arithmetic

Quantifiers Module of CVC4



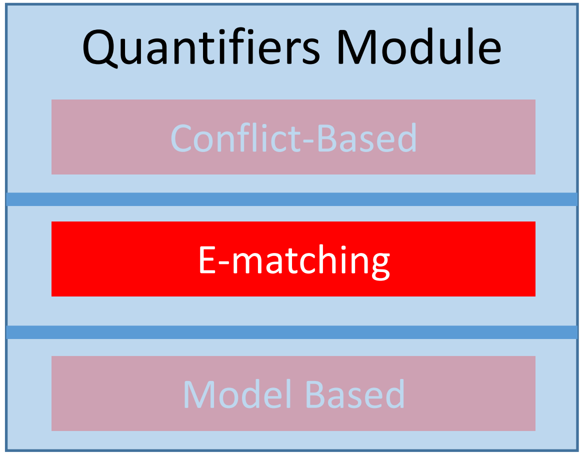
- CVC4's quantifiers module contains numerous strategies and techniques

Quantifiers Module of CVC4



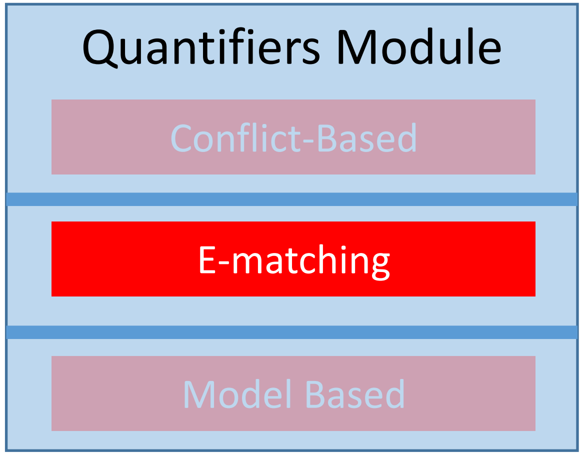
- Core techniques: Conflict-based, Heuristic (e.g. E-matching), Model-based

E-matching



- E-matching:
  - Most widely used and successful technique for quantifiers in SMT
  - Implemented in numerous solvers:
    - Z3, CVC3, CVC4, VeriT, Alt-Ergo, …

E-matching: Example



$$\left.\begin{array}{l} \texttt{a,b,c:S} \\ \texttt{f,g:S}\rightarrow\texttt{S} \\ \texttt{f(a)=a, f(b)=b, f(c)=c, g(a)}\neq\texttt{a} \end{array}\right\} \texttt{E}$$

$$\left.\forall \texttt{x.f(x)=g(x)}\right\} \texttt{Q}$$

Quantifiers Module

Conflict-Based

E-matching

Model Based

E-matching: Example

$a,b,c : S$

$f,g : S \to S$

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a) \neq a$

$$\forall x. \underbrace{\mathbf{f(x)}}_{\text{Pattern}} = g(x)$$

- **Idea**: choose instances based on pattern matching

E-matching: Example

Quantifiers Module

Conflict-Based

E-matching

Model Based

$a, b, c : S$

$f, g : S \to S$

$f(a) = a,\ f(b) = b,\ f(c) = c,\ g(a) \neq a$

$\{x \to a\}$ $\{x \to b\}$ $\{x \to c\}$

$\forall x. f(x) = g(x)$

E-matching: Example

Quantifiers Module

Conflict-Based

E-matching

Model Based

$a,b,c:S$

$f,g:S\rightarrow S$

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)\neq a$

$\mathbf{f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c)}$

$\forall x.f(x)=g(x)$

Quantifiers Module

Conflict-Based

E-matching

Model Based

E-matching: Example

`a,b,c:S`

`f,g:S→S`

`f(a)=a`, `f(b)=b, f(c)=c,` **`g(a)≠a`**

`f(a)=g(a)`, `f(b)=g(b), f(c)=g(c)`

`∀x.f(x)=g(x)`

unsat

EXAMPLE…

E-matching: Challenges

- What happens when there too many instances to add?
  - E-matching adds many instances, degrades performance for solver to continue

- What happens when there are no instances to add?
  - E-matching is an incomplete procedure, cannot answer SAT even when saturated

E-matching: Challenges

- What happens when there too many instances to add?
  - E-matching adds many instances, degrades performance for solver to continue

$\Rightarrow$*Use conflict-based instantiation* **[Reynolds/Tinelli/deMoura FMCAD14]**

- What happens when there are no instances to add?
  - E-matching is an incomplete procedure, cannot answer SAT even when saturated

$\Rightarrow$*Use model-based instantiation* **[Ge/deMoura CAV09]**

Model-based Instantiation: Example

Quantifiers Module

Conflict-Based

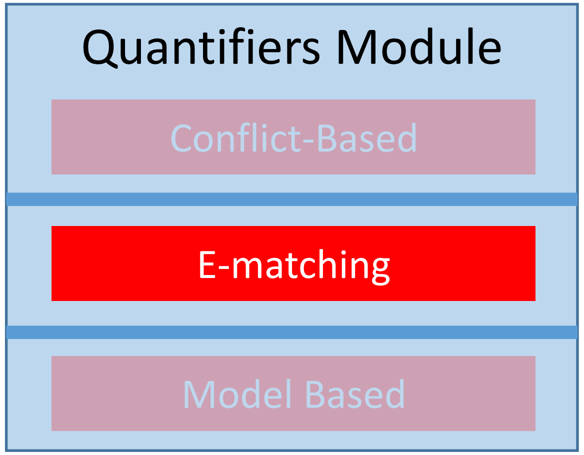
E-matching

Model Based

`f(a)=a,  f(b)=b,  f(c)=c,` **`g(a)=a`**

$\forall x. f(x)=g(x)$

Model-based Instantiation: Example



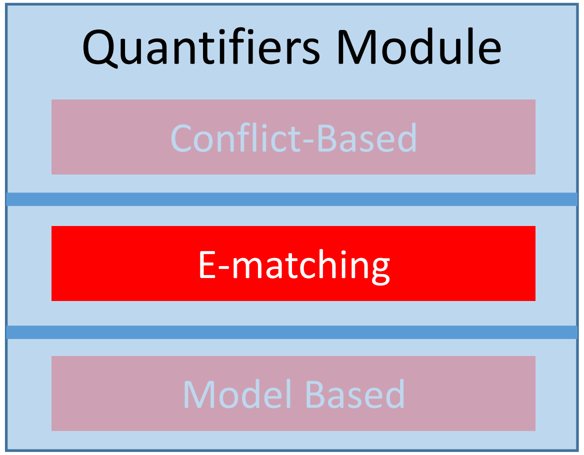
`f(a)=a, f(b)=b, f(c)=c, g(a)=a`

`f(a)=g(a), f(b)=g(b), f(c)=g(c)`

`∀x.f(x)=g(x)`

- Add instances by E-matching, as before

Model-based Instantiation: Example



$$\left.\begin{array}{l} \mathtt{f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)=a} \\ \mathtt{f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c)} \end{array}\right\}\ \text{sat}$$

$$\forall \mathtt{x.f(x)=g(x)}$$

- E-matching saturates, but ground constraints are satisfiable
    - Can we check that $\forall \mathtt{x.f(x)=g(x)}$ is also satisfiable?

Model-based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model-Based

`f(a)=a, f(b)=b, f(c)=c, g(a)=a`

`f(a)=g(a), f(b)=g(b), f(c)=g(c)`

`∀x.f(x)=g(x)`

- **Idea**: construct candidate model `M` for functions `f` and `g`
  - Check if `∀x.f(x)=g(x)` satisfied by `M`

Model-based Instantiation: Example

Quantifiers Module

- Conflict-Based
- E-matching
- **Model-Based**

f(a)=a, f(b)=b, f(c)=c, g(a)=a

f(a)=g(a), f(b)=g(b), f(c)=g(c)

∀x.f(x)=g(x)

$$\underbrace{f^M := \lambda x.\mathrm{ite}(x=a, a, \mathrm{ite}(x=b, b, c))}_{M}$$

Model-based Instantiation: Example

Quantifiers Module

- Conflict-Based
- E-matching
- **Model-Based**

f(a)=a, f(b)=b, f(c)=c, g(a)=a

f(a)=g(a), f(b)=g(b), f(c)=g(c)

$\forall$x.f(x)=g(x)

f^M := λx.ite(x=a,a,ite(x=b,b,c))

g^M := λx.ite(x=a,a,ite(x=b,b,c))

M

Model-based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model-Based

`f(a)=a, f(b)=b, f(c)=c, g(a)=a`

`f(a)=g(a), f(b)=g(b), f(c)=g(c)`

$\forall$`x.f(x)=g(x)`

$$\left.\begin{array}{l} f^M := \lambda x.\mathtt{ite}(x=a,a,\mathtt{ite}(x=b,b,c)) \\ g^M := \lambda x.\mathtt{ite}(x=a,a,\mathtt{ite}(x=b,b,c)) \end{array}\right\} M$$

- Does M satisfy $\forall$`x.f(x)=g(x)`?

Quantifiers Module

Conflict-Based

E-matching

Model-Based

Model-based Instantiation: Example

`f(a)=a, f(b)=b, f(c)=c, g(a)=a`

`f(a)=g(a), f(b)=g(b), f(c)=g(c)`

$\forall x.f(x)=g(x)$

$$\left.\boxed{\begin{array}{l} f^M := \lambda x.\mathrm{ite}(x=a,a,\mathrm{ite}(x=b,b,c)) \\ g^M := \lambda x.\mathrm{ite}(x=a,a,\mathrm{ite}(x=b,b,c)) \end{array}}\right\} M$$

- Does M satisfy $\forall x.f(x)=g(x)$?

$\Rightarrow$ If $\exists x.f^M(x)\neq g^M(x)$ is unsat, then yes

Quantifiers Module

Conflict-Based

E-matching

Model-Based

Model-based Instantiation: Example

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)=a$

$f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c)$

$\forall x. f(x)=g(x)$

$$\left.\boxed{\begin{array}{l} f^M := \lambda x.\mathrm{ite}(x=a,a,\mathrm{ite}(x=b,b,c)) \\ g^M := \lambda x.\mathrm{ite}(x=a,a,\mathrm{ite}(x=b,b,c)) \end{array}}\right\} M$$

- Does M satisfy $\forall x. f(x)=g(x)$?

$$\left.\mathrm{ite}(x=a,a,\mathrm{ite}(x=b,b,c)) \neq \mathrm{ite}(x=a,a,\mathrm{ite}(x=b,b,c))\right\} \text{unsat}$$

Model-based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model-Based

$\texttt{f(a)=a, f(b)=b, f(c)=c, g(a)=a}$

$\texttt{f(a)=g(a), f(b)=g(b), f(c)=g(c)}$

$\forall\texttt{x.f(x)=g(x)}$

$$\left.\boxed{\begin{array}{l}\texttt{f}^{\mathrm{M}}\ \texttt{:=}\ \lambda\texttt{x.ite(x=a,a,ite(x=b,b,c))}\\ \texttt{g}^{\mathrm{M}}\ \texttt{:=}\ \lambda\texttt{x.ite(x=a,a,ite(x=b,b,c))}\end{array}}\right\}\ \mathrm{M}$$

- Does M satisfy $\forall\texttt{x.f(x)=g(x)}$?
  - $\Rightarrow$ Yes, return sat with model M

Model-based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model-Based

f(a)=a, f(b)=b, f(c)=c, g(a)=a

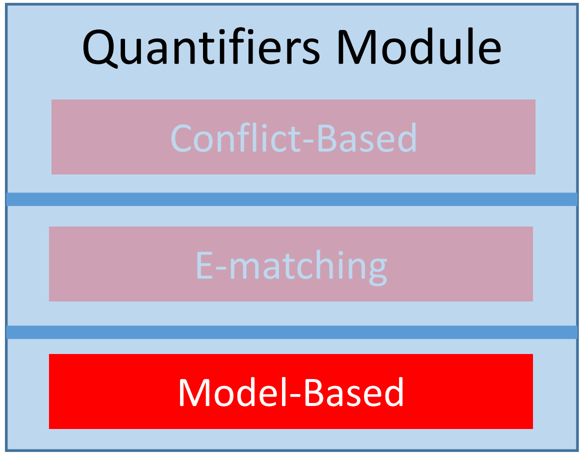
f(a)=g(a), f(b)=g(b), f(c)=g(c), **d∉{a,b,c}**

∀x.f(x)=g(x)

$$\left.\boxed{\begin{array}{l} f^{M} := \lambda x.\mathrm{ite}(x{=}a,a,\mathrm{ite}(x{=}b,b,c)) \\ g^{M} := \lambda x.\mathrm{ite}(x{=}a,a,\mathrm{ite}(\mathbf{x{=}c,c,b})) \end{array}}\right\} M$$

- If M does not satisfy ∀x.f(x)=g(x),

Model-based Instantiation: Example



$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)=a$

$f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c),\ d\notin\{a,b,c\}$

$\forall x.f(x)=g(x)$

$$\left.\boxed{\begin{array}{l} f^M := \lambda x.ite(x=a,a,ite(x=b,b,c)) \\ g^M := \lambda x.ite(x=a,a,ite(x=c,c,b)) \end{array}}\right\}\ M$$

- If M does not satisfy $\forall x.f(x)=g(x)$,

$ite(x=a,a,ite(x=b,b,c))\neq ite(x=a,a,ite(x=c,c,b))$ is unsat?

Model-based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model-Based

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)=a$

$f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c),\ d\notin\{a,b,c\}$

$\forall x.f(x)=g(x)$

$$\left.\boxed{\begin{array}{l} f^{M}\ := \lambda x.ite(x=a,a,ite(x=b,b,c)) \\ g^{M}\ := \lambda x.ite(x=a,a,ite(x=c,c,b)) \end{array}}\right\}\ M$$

- If M does not satisfy $\forall x.f(x)=g(x)$,

$\mathbf{ite(d=a,a,ite(d=b,b,c))\neq ite(d=a,a,ite(d=c,c,b))}$ Take $\mathbf{x=d}$

Model-based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model-Based

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)=a$

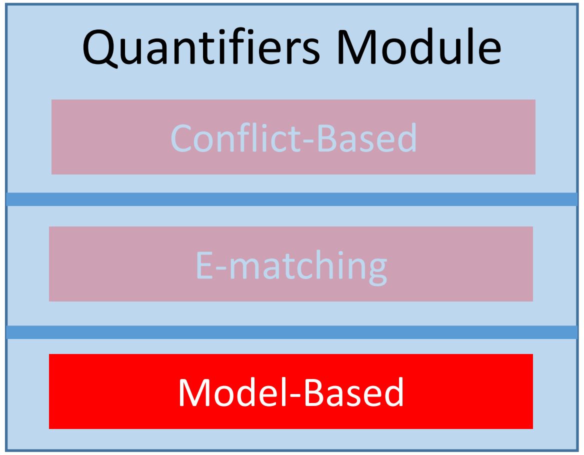
$f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c),\ d\notin\{a,b,c\}$

$\forall x.\, f(x)=g(x)$

$$\left.\boxed{\begin{array}{l} f^M := \lambda x.\,\mathrm{ite}(x=a,a,\mathrm{ite}(x=b,b,c)) \\ g^M := \lambda x.\,\mathrm{ite}(x=a,a,\mathrm{ite}(x=c,c,b)) \end{array}}\right\} M$$

- If M does not satisfy $\forall x.\, f(x)=g(x)$,

$$\left.c\neq b\right\} \text{sat, where } x=d$$



Model-based Instantiation: Example

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)=a$

$f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c),\ d \notin \{a,b,c\},$

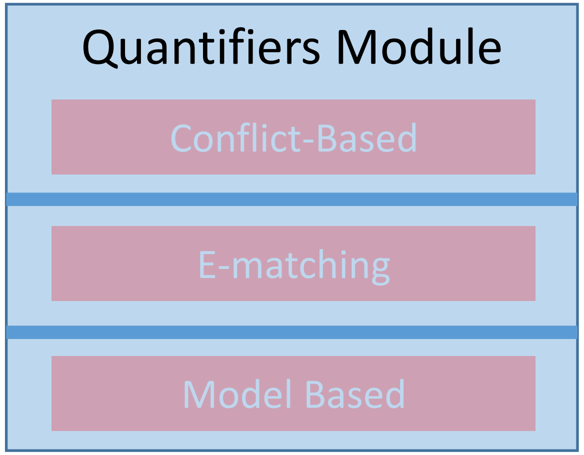
$\forall x.\, f(x)=g(x)$ **$f(d)=g(d)$**

$$\left.\boxed{\begin{array}{l} f^M := \lambda x.\, ite(x=a, a, ite(x=b, b, c)) \\ g^M := \lambda x.\, ite(x=a, a, ite(x=c, c, b)) \end{array}}\right\} M$$

- If M does not satisfy $\forall x.\, f(x)=g(x)$,
  $\Rightarrow$Add instance **$f(d)=g(d)$**, will refine model

EXAMPLE…

Conflict-Based Instantiation: Example



$f(a)=a, \quad f(b)=b, \quad f(c)=c, \quad g(a)\neq a$

$\forall x. f(x)=g(x)$

Conflict-Based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model Based

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)\neq a$

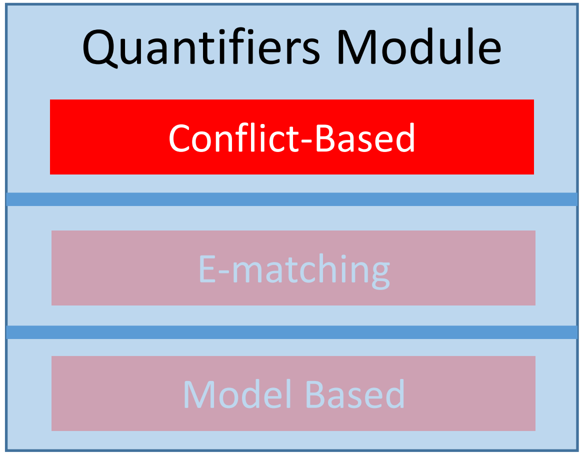
$\mathbf{f(a)=g(a),\ f(b)=g(b),\ f(c)=g(c),\ \ldots}$

$\forall x.f(x)=g(x)$

- E-matching may return with many ground instances
  - In practice, 1000+ instances per invocation

    $\Rightarrow$Degrades solver performance

Conflict-Based Instantiation: Example

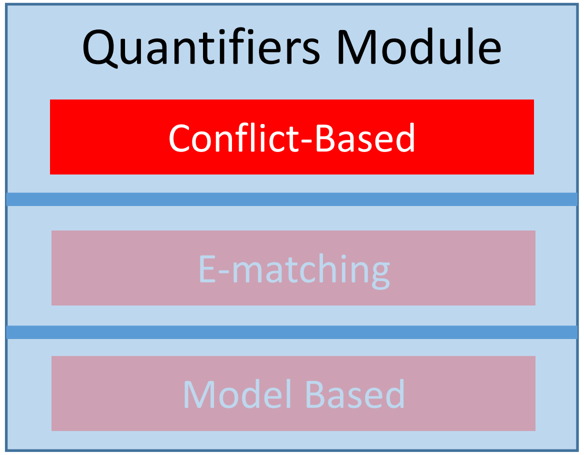


$f(a)=a, \quad f(b)=b, \quad f(c)=c, \quad g(a)\neq a$

$\forall x. f(x)=g(x)$

- **Idea**: find an instance of $\forall x. f(x)=g(x)$ that is conflicting with ground constraints
  - If so, add *only* that instance

Conflict-Based Instantiation: Example



$\mathtt{f(a)=a}$, $\mathtt{f(b)=b}$, $\mathtt{f(c)=c}$, $\mathtt{g(a)\neq a}$

$\forall \mathtt{x.f(x)=g(x)}$

- **Idea**: find an instance of $\forall \mathtt{x.f(x)=g(x)}$ that is conflicting with ground constraints

  $\Rightarrow$ $\mathtt{f(a)=a,\ g(a)\neq a} \models \mathtt{f(x)\neq g(x)\{x\rightarrow a\}}$

Conflict-Based Instantiation: Example

Quantifiers Module

Conflict-Based

E-matching

Model Based

$f(a)=a,\ f(b)=b,\ f(c)=c,\ g(a)\neq a$

$\mathbf{f(a)=g(a)}$

$\forall x. f(x)=g(x)$

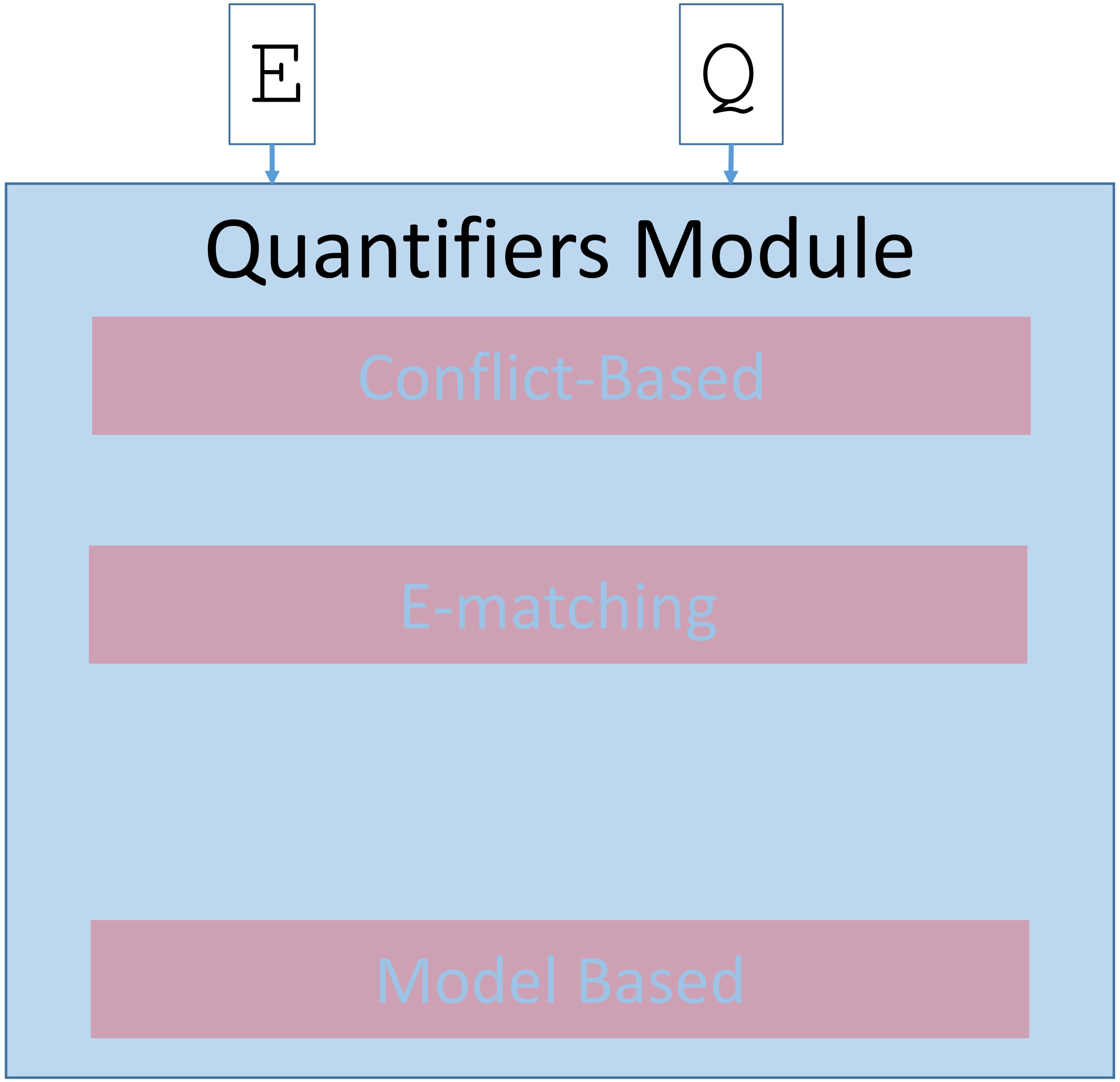
unsat

- **Idea**: find an instance of $\forall x. f(x)=g(x)$ that is conflicting with ground constraints

  $\Rightarrow f(a)=a,\ g(a)\neq a \models f(x)\neq g(x)\{x \to \mathbf{a}\}$

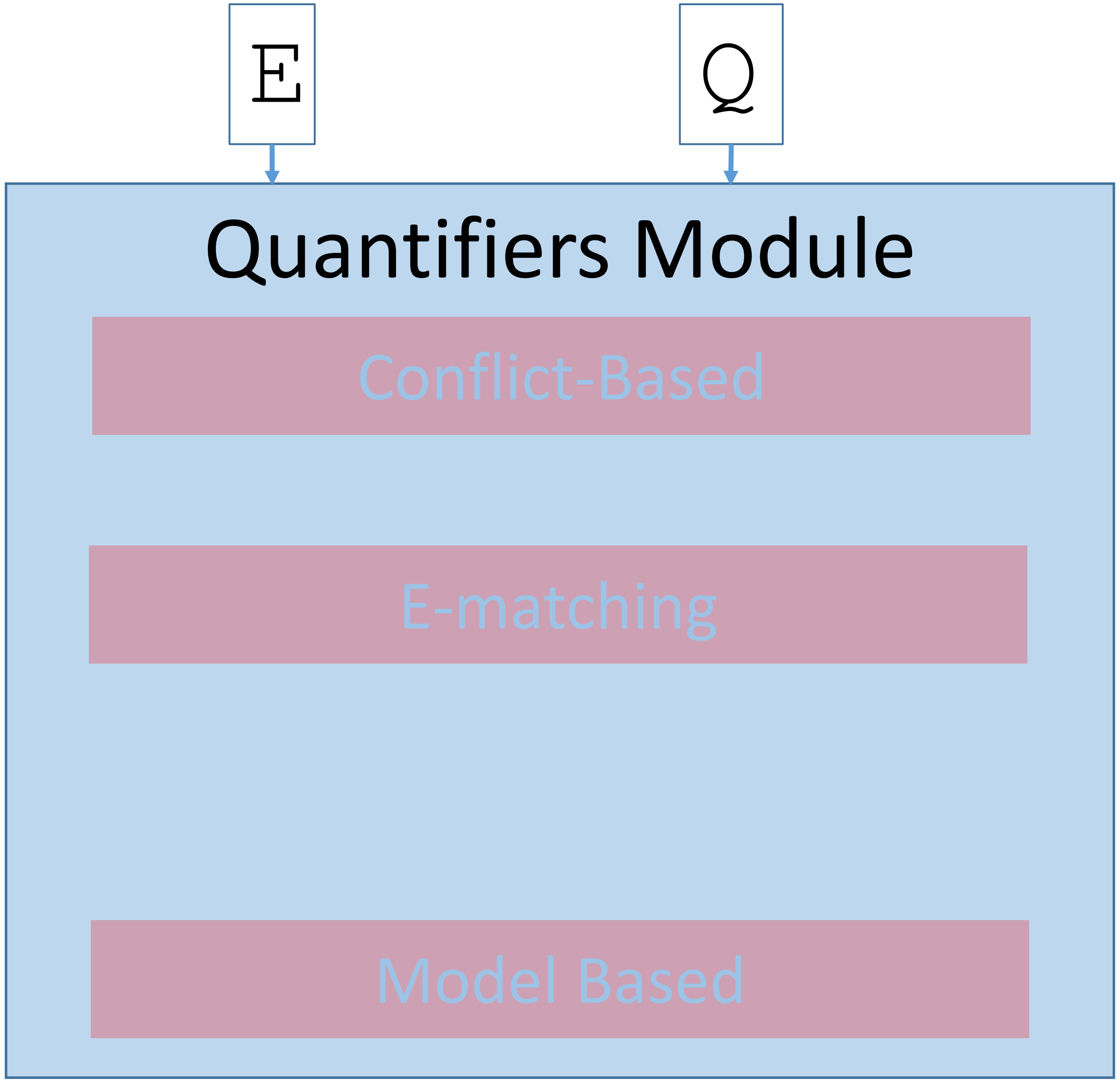
EXAMPLE...

Putting it Together

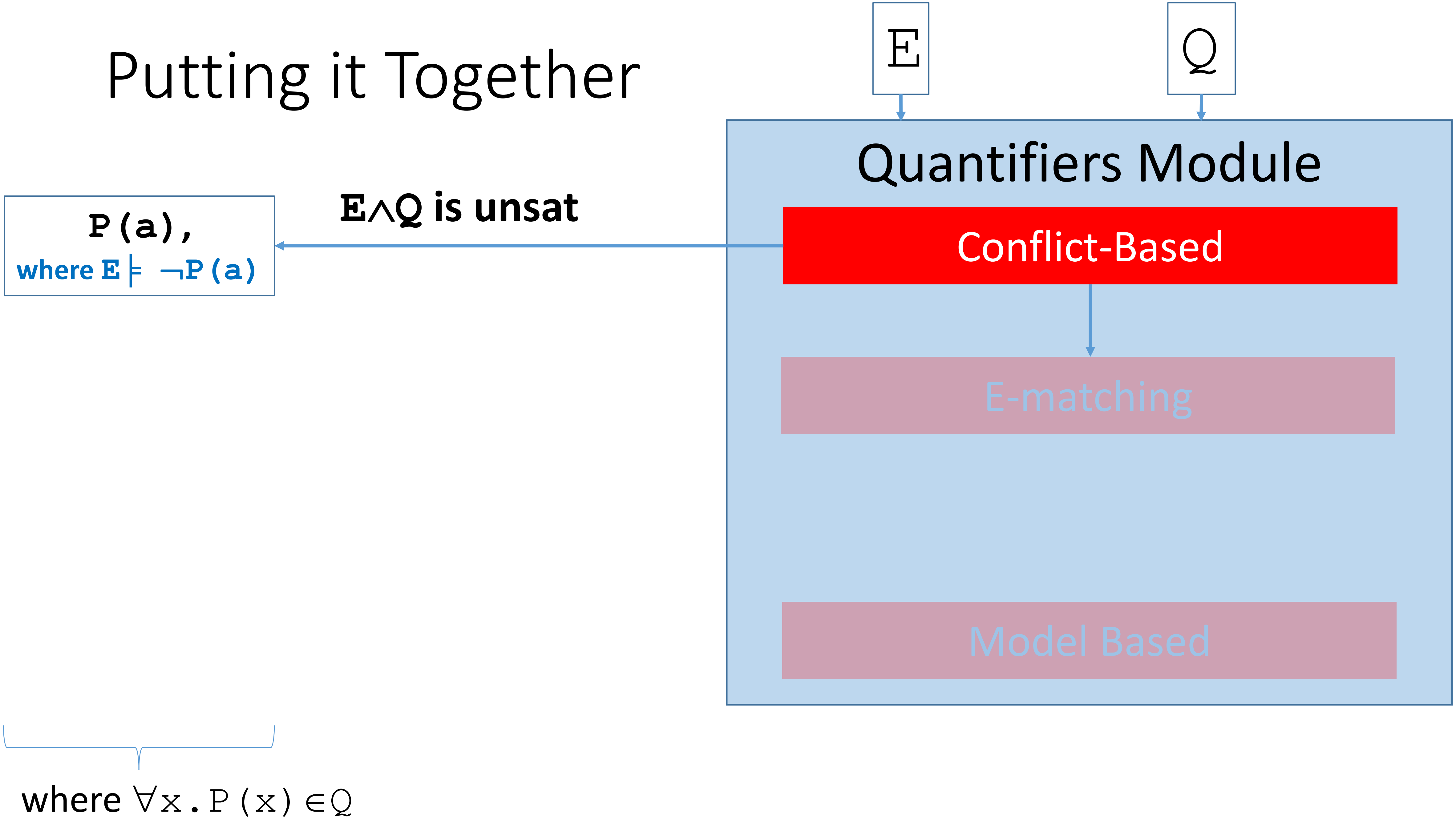


Putting it Together

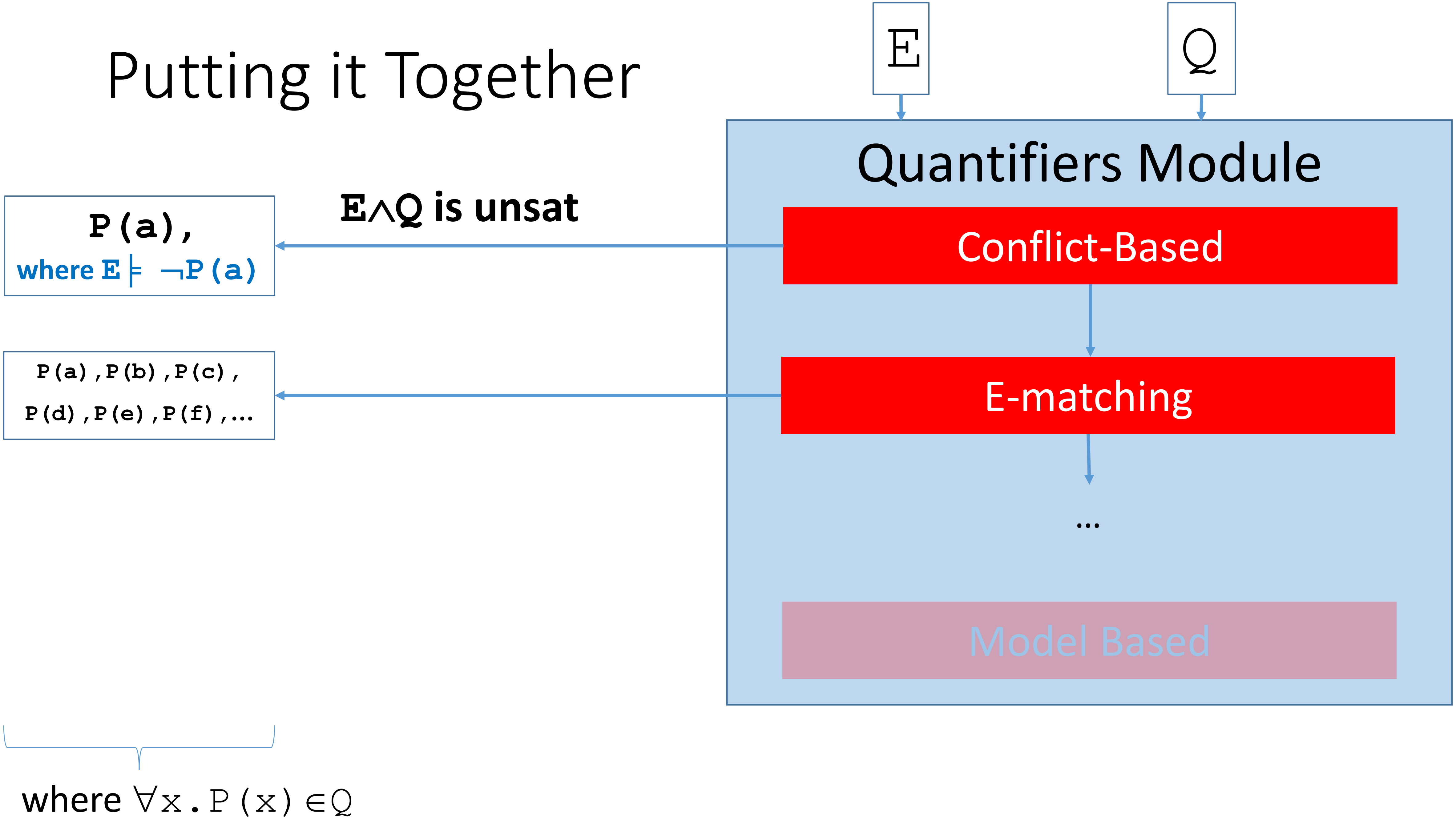
- Input:
  - Ground literals $\mathbb{E}$
  - Quantified formulas $\mathbb{Q}$



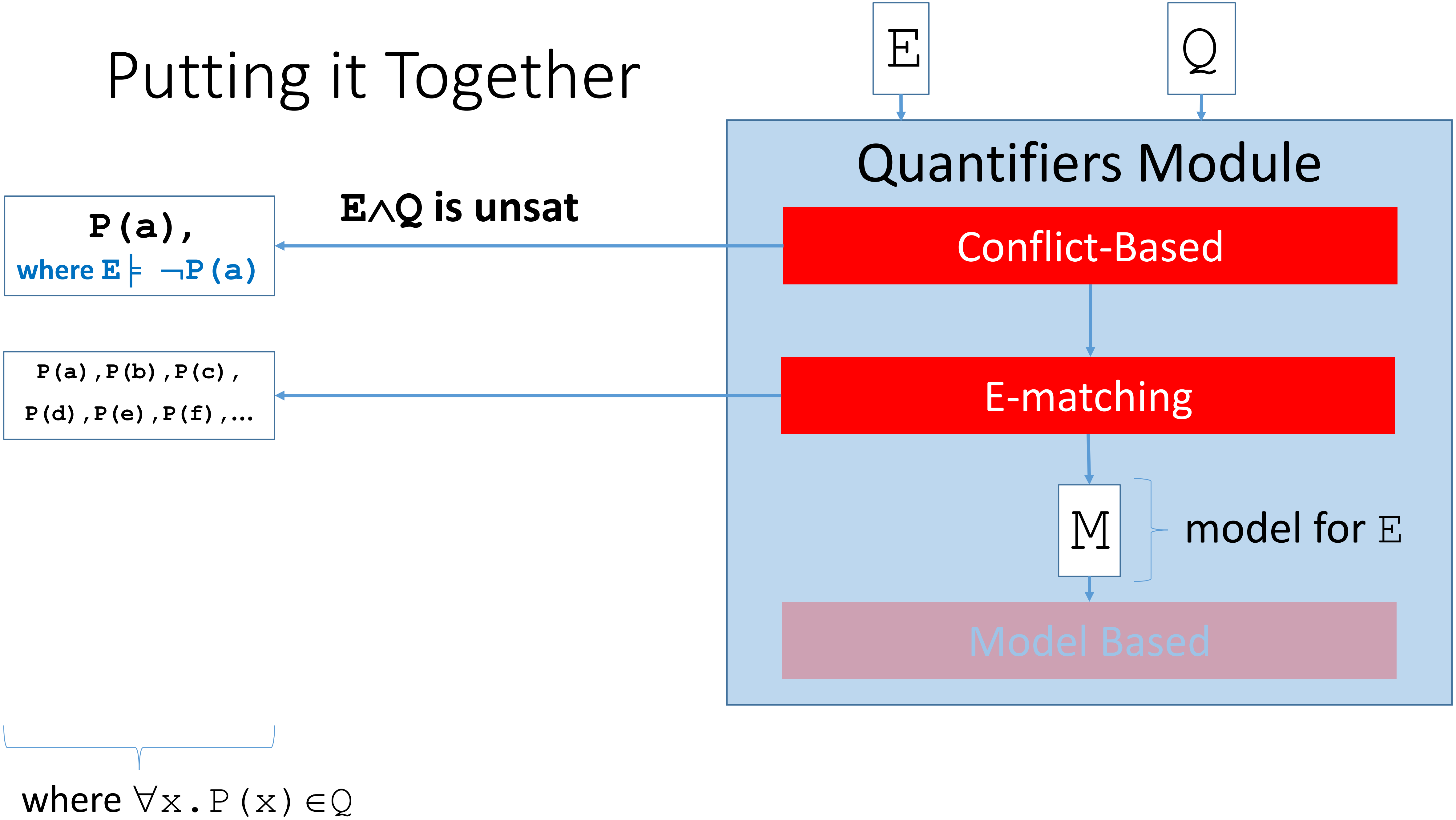
Putting it Together



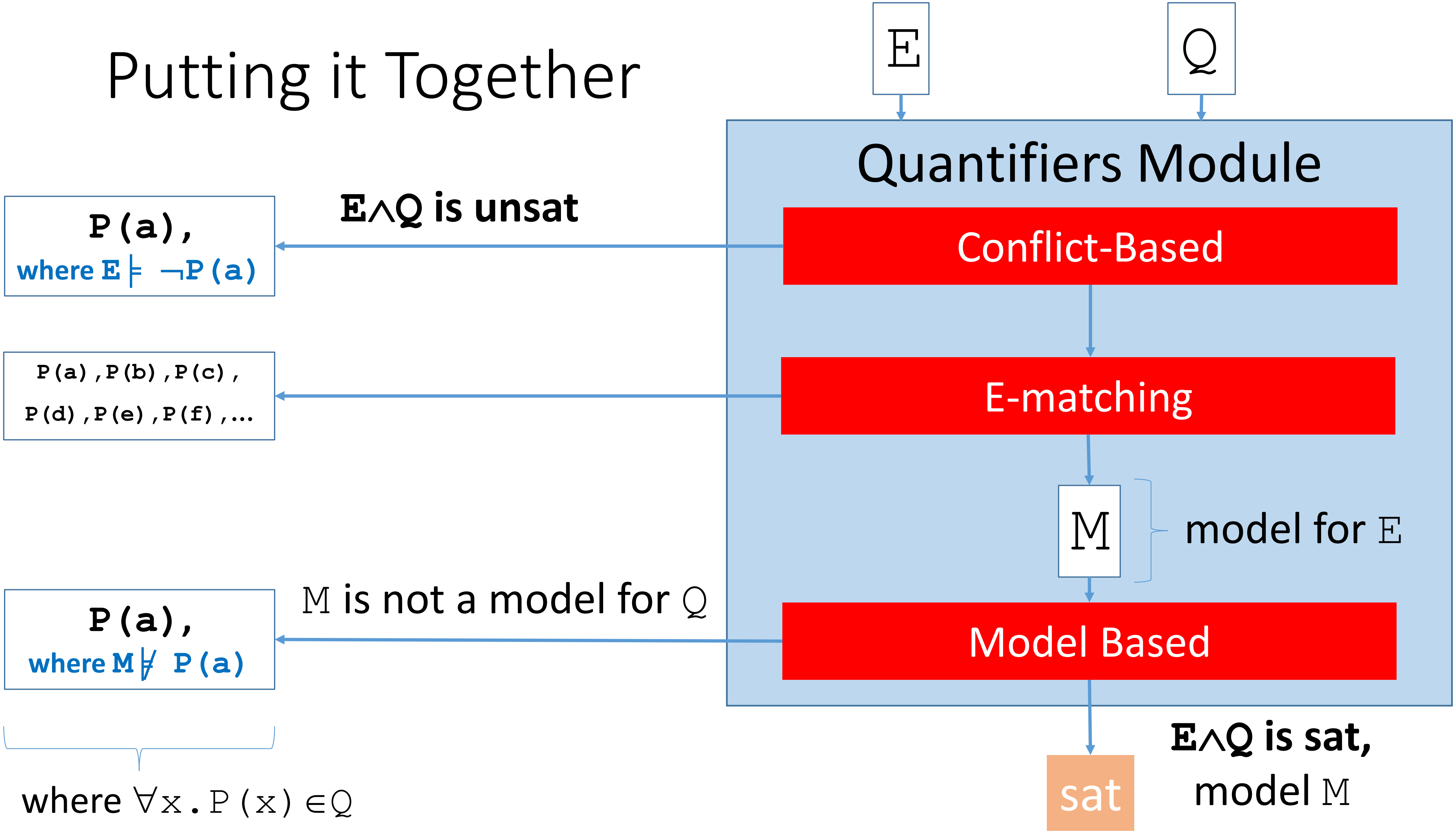
Putting it Together



Putting it Together



Putting it Together



Other techniques for Quantified Formulas

- Advanced techniques in CVC4:
  - Rewrite Rules
  - Automated Induction **[Reynolds/Kuncak VMCAI15]**
  - Finite Model Finding **[Reynolds et al CADE13]**
  - Synthesis **[Reynolds et al CAV15]**

  $\Rightarrow$ Each target a particular type of quantified formulas

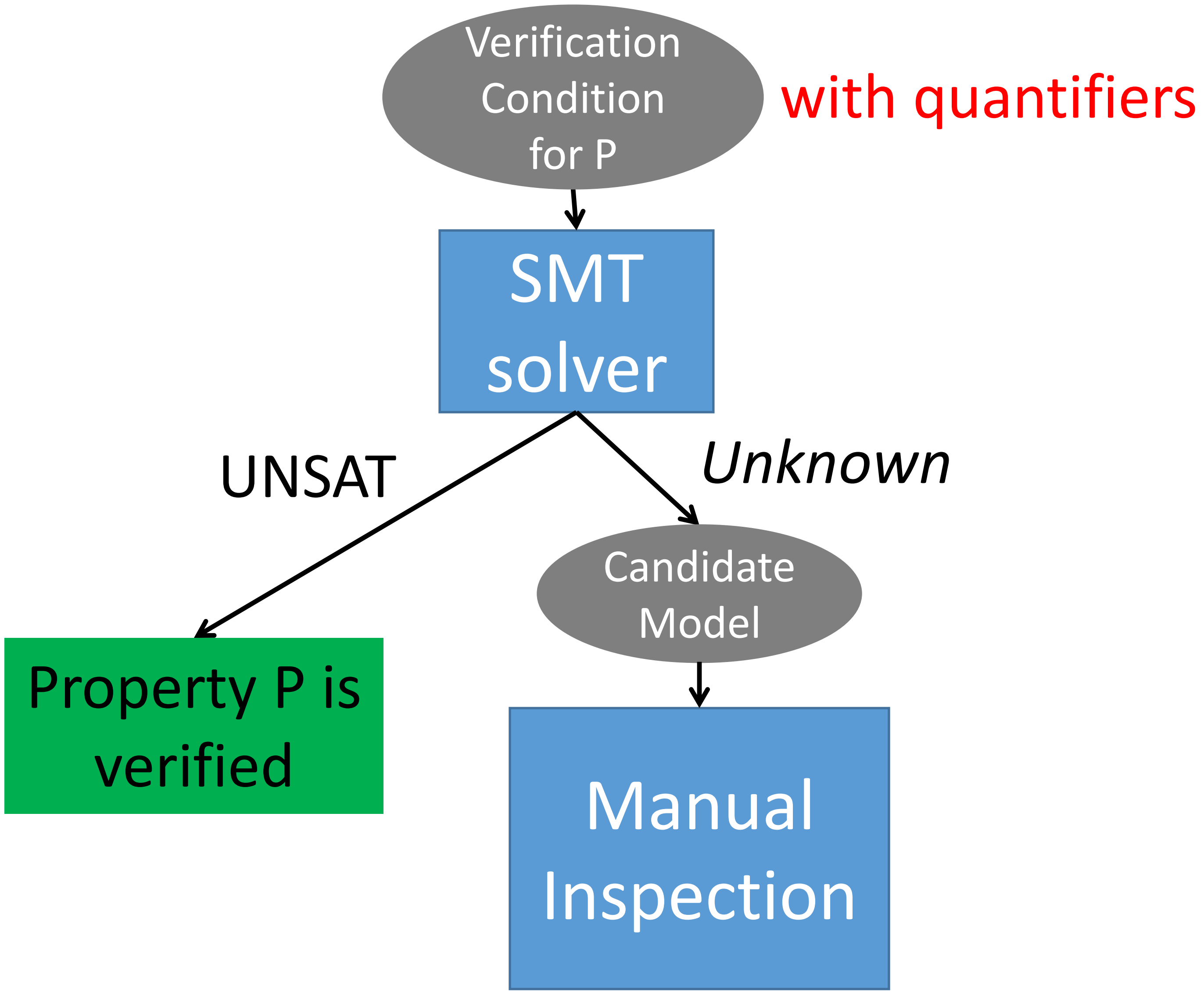
Other techniques for Quantified Formulas

- Advanced techniques in CVC4:
  - Rewrite Rules
  - Automated Induction **[Reynolds/Kuncak VMCAI15]**
  - Finite Model Finding **[Reynolds et al CADE13]**
  - Synthesis **[Reynolds et al CAV15]**

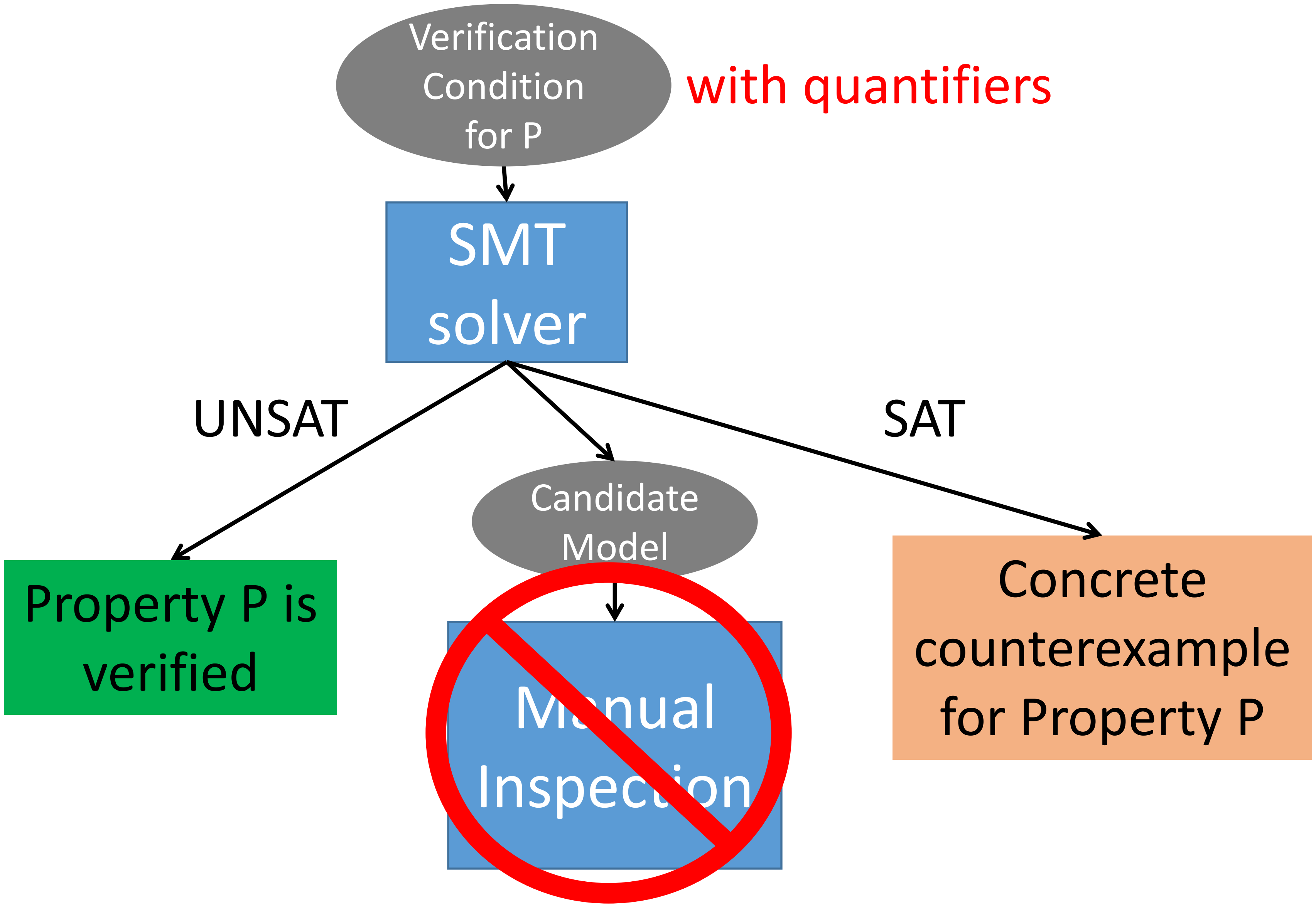
Focus of the remainder

$\Rightarrow$ Each target a particular type of quantified formulas

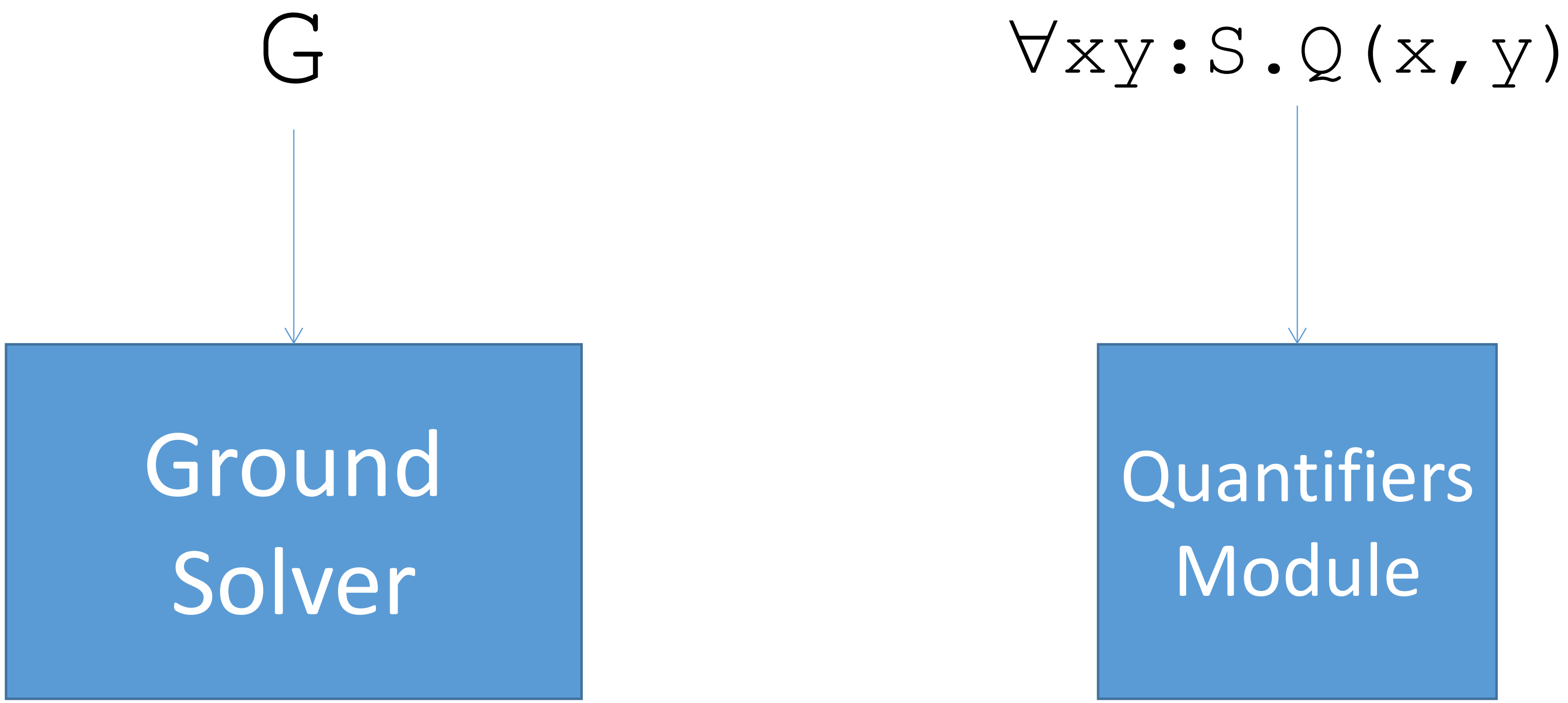
Finite Model Finding : Motivation



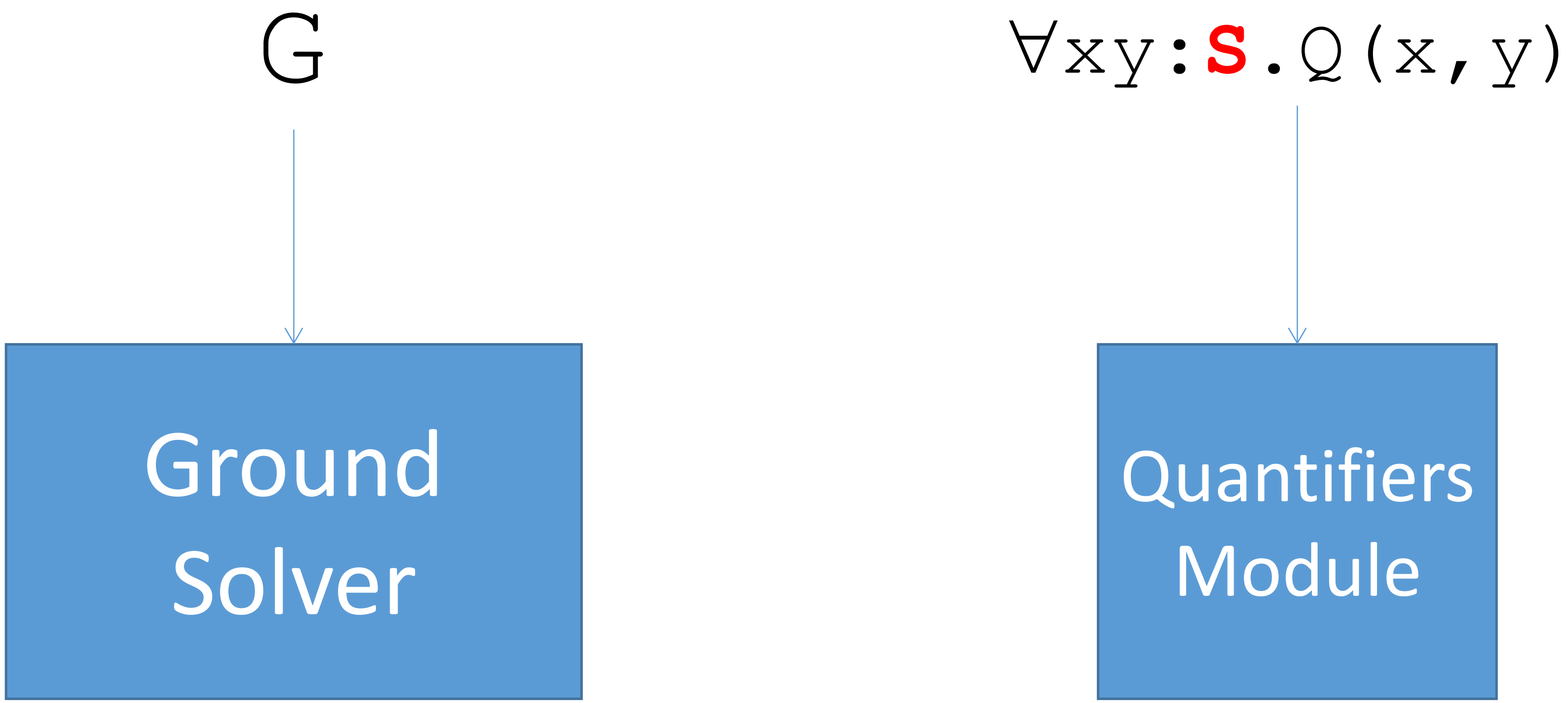
Finite Model Finding : Motivation



Finite Model Finding in SMT



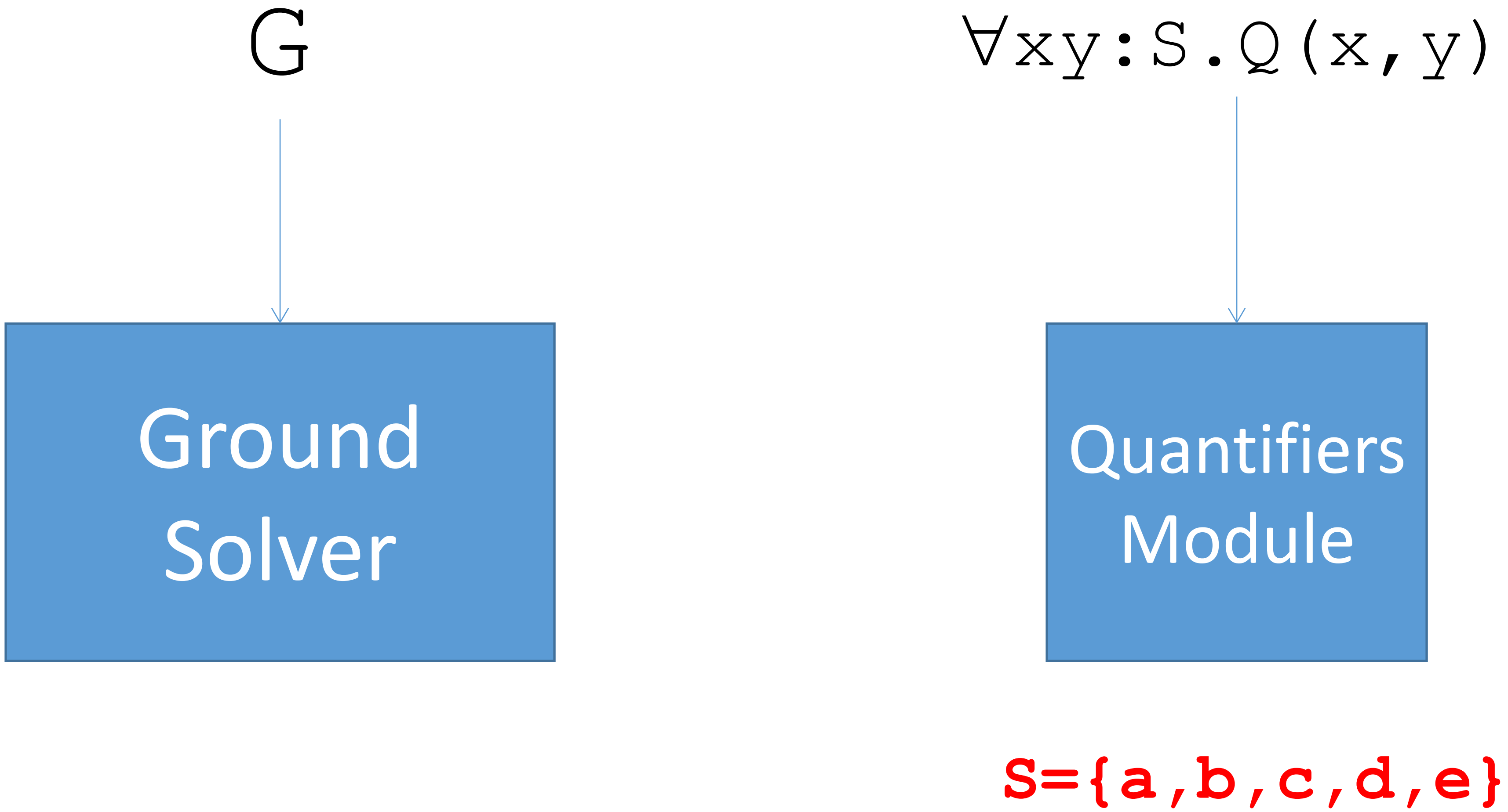
Finite Model Finding in SMT



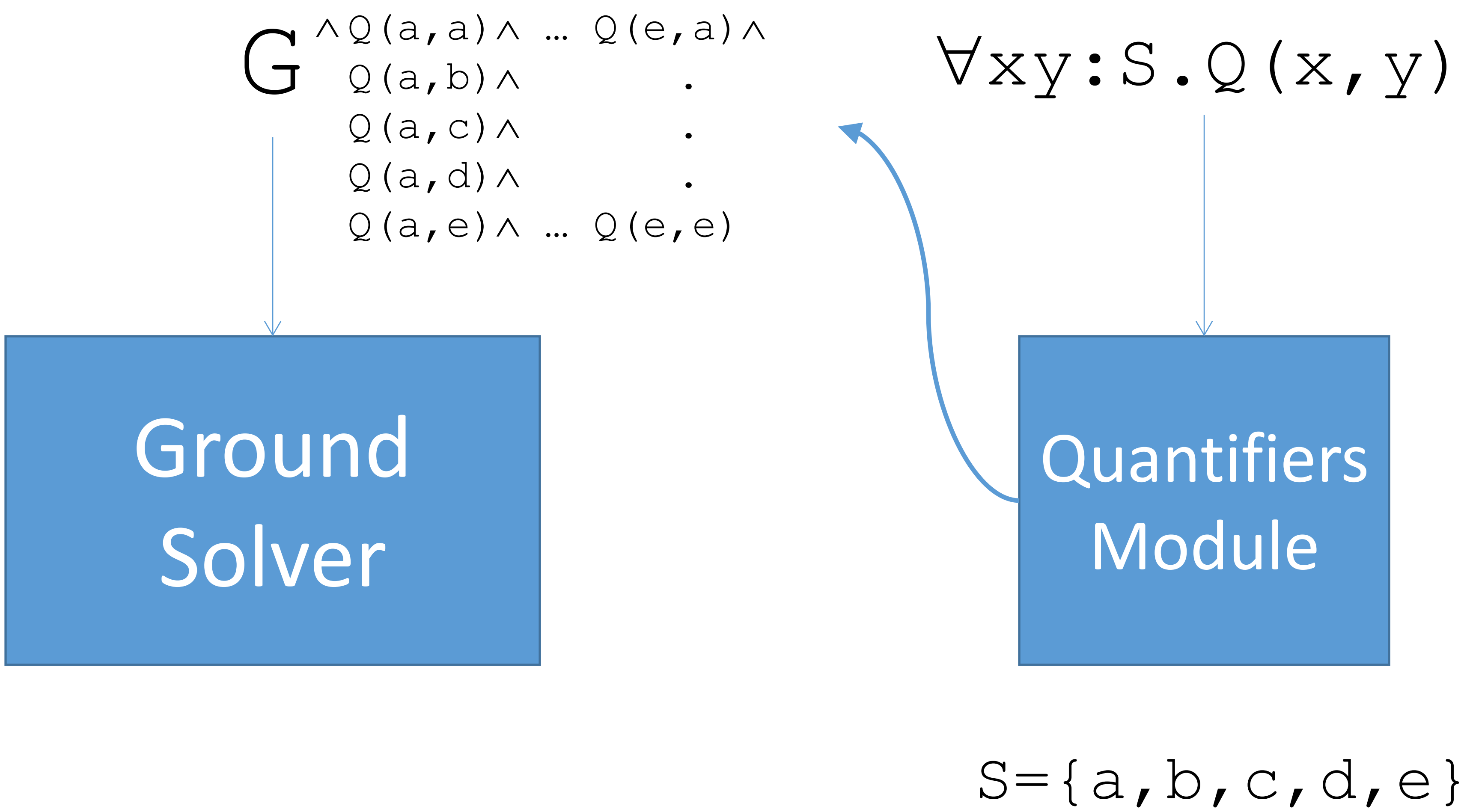
⇒ *If **S** has finite interpretation,*

- *use finite model finding*

Finite Model Finding in SMT

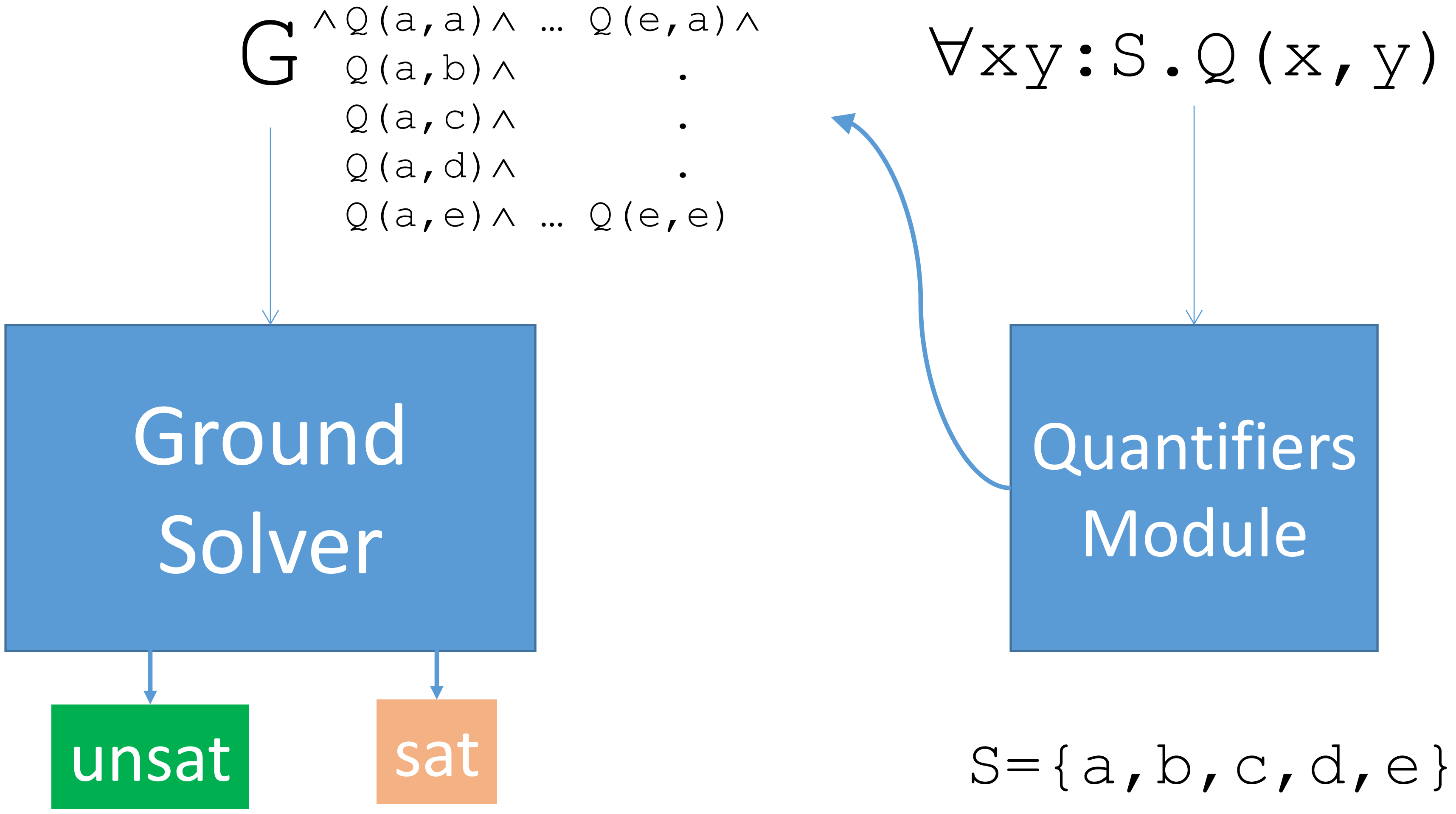


Finite Model Finding in SMT



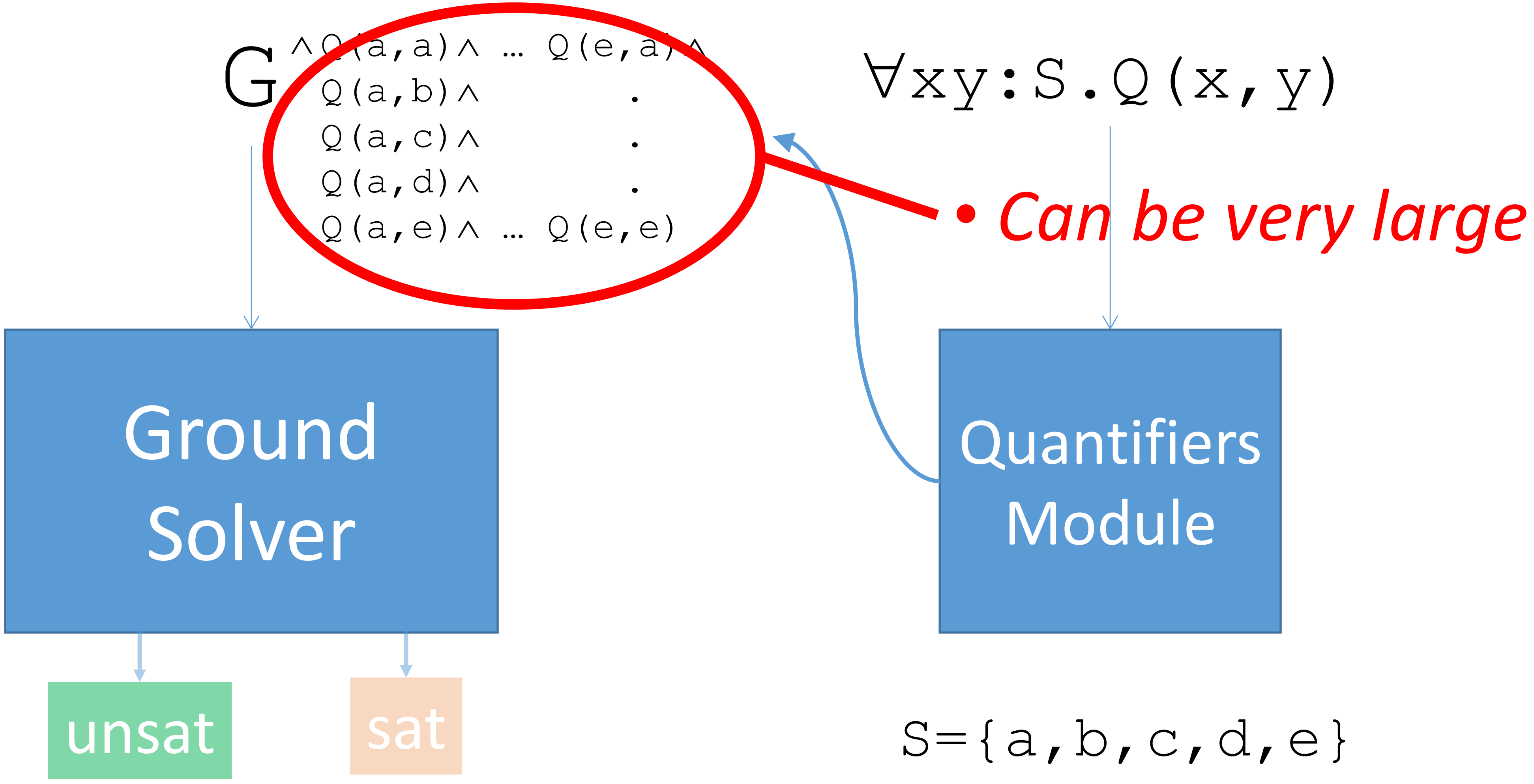
- Reduction of quantified formulas to ground formulas

Finite Model Finding in SMT



⇒ Ability to answer SAT, assuming decision procedure for G∧Q(a,a)∧…∧Q(e,e)

Finite Model Finding in SMT



Finite Model Finding: Example

$a,b,c,d,e : S$

$P,R : (S,S) \rightarrow Bool$

$a \neq b,\ b \neq c,\ c \neq d,\ d \neq e,\ e \neq a$

$\neg P(a,b),\ \neg R(a,c)$

$\forall xy. P(x,y) \vee R(x,y)$

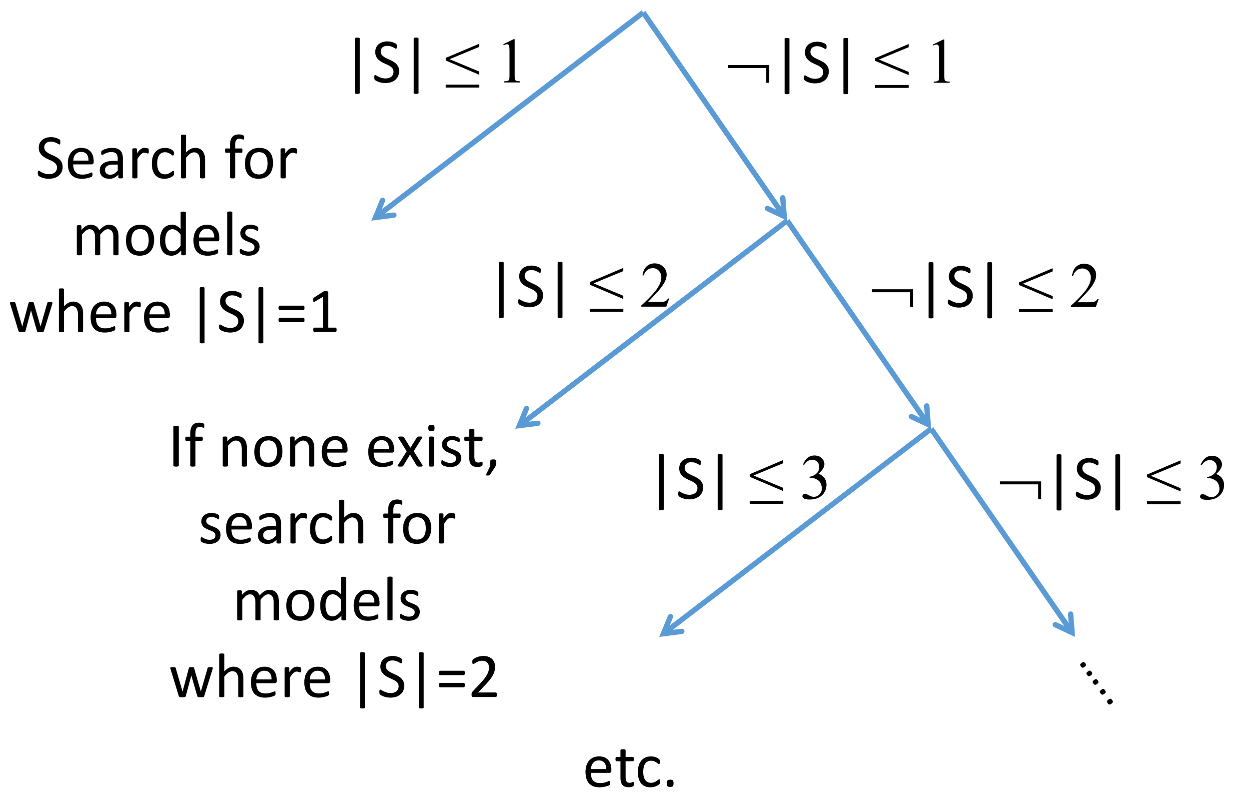
EXAMPLE…

Finite Model Finding in SMT

- Address large # instantiations by:
  1. Minimizing model sizes **[Reynolds et al CAV13]**
     - Find interpretation that minimizes the #elements in S
  2. Only add instantiations that refine model **[Reynolds et al CADE13]**
     - Model-based quantifier instantiation [Ge/deMoura CAV 2009]

Finite Model Finding : Minimizing Model Sizes

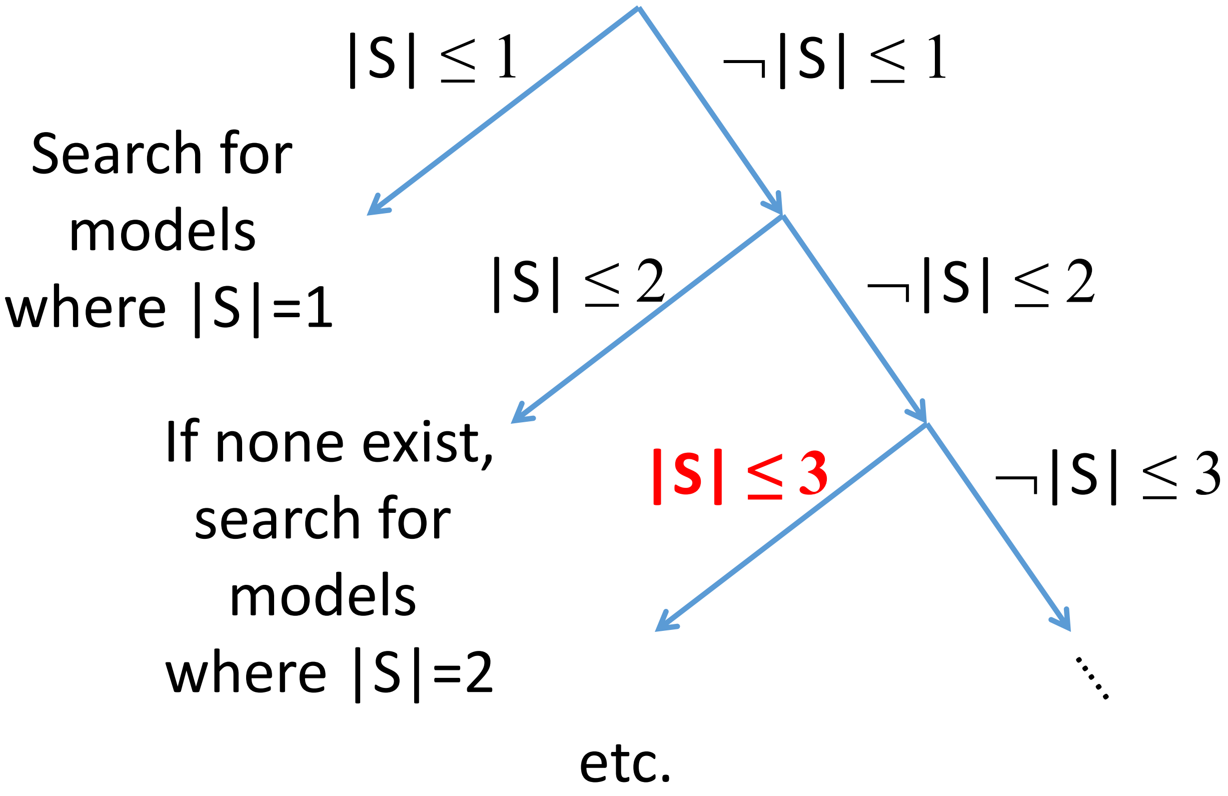
- Minimize model sizes using a theory solver for cardinality constraints



[Reynolds/Tinelli/Goel/Krstic CAV13]

Finite Model Finding : Minimizing Model Sizes

- Minimize model sizes using a theory solver for cardinality constraints



$\Rightarrow$ If model exists where $|S| \leq 3$, only need 3*3=9 instances instead of 5*5=25 instances

[Reynolds/Tinelli/Goel/Krstic CAV13]

FMF: Example

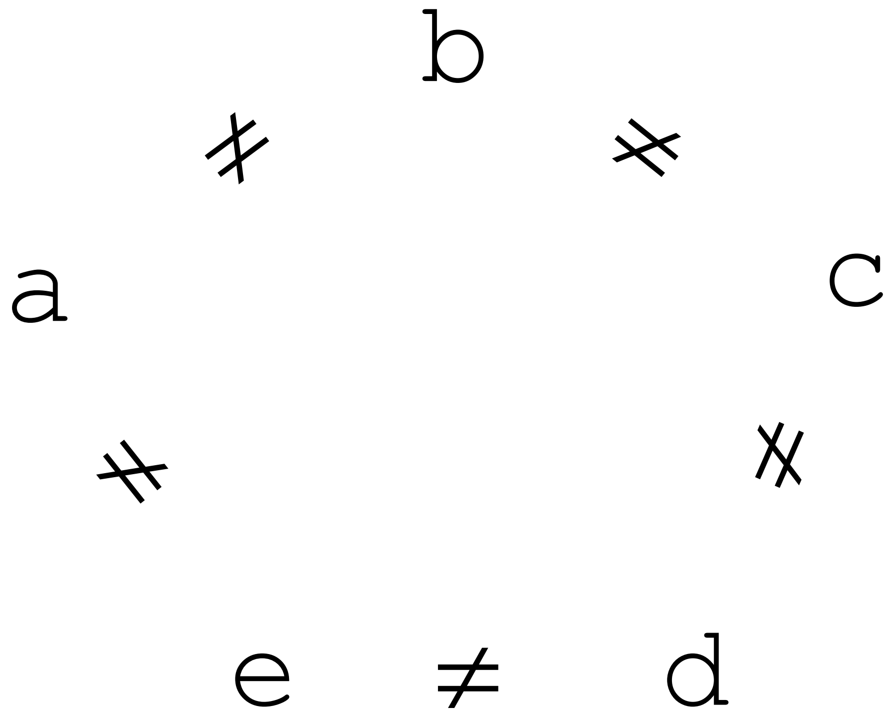
a,b,c,d,e:S

P,R:(S,S)→Bool

a≠b, b≠c, c≠d, d≠e, e≠a

¬P(a,b), ¬R(a,c)

∀xy.P(x,y)∨R(x,y)



FMF: Example

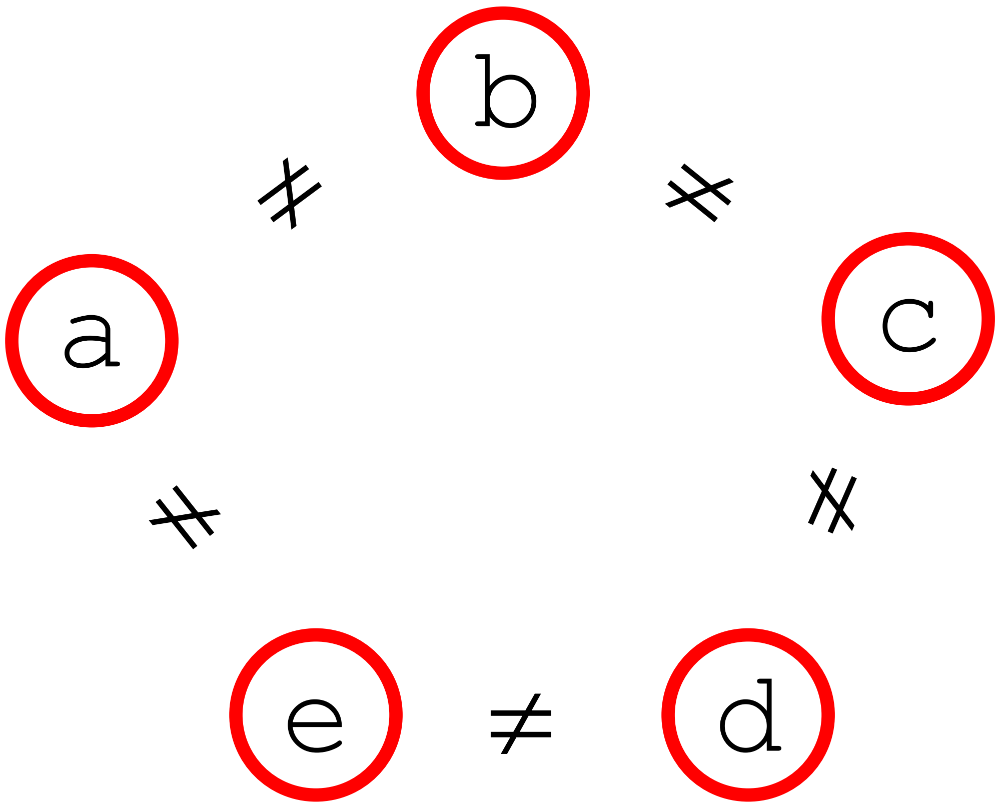
$a,b,c,d,e:S$

$P,R:(S,S)\rightarrow Bool$

$a\neq b,\ \ b\neq c,\ \ c\neq d,\ \ d\neq e,\ \ e\neq a$

$\neg P(a,b),\ \ \neg R(a,c)$

$\forall xy.P(x,y)\vee R(x,y)$



S={a,b,c,d,e}

FMF: Example

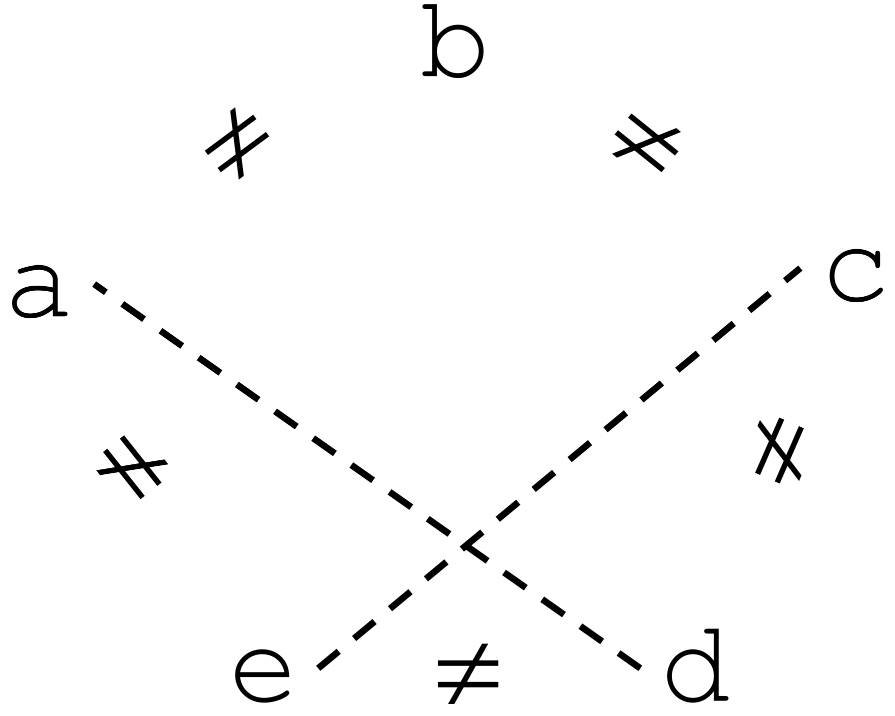
a,b,c,d,e:S

P,R:(S,S)→Bool

a≠b, b≠c, c≠d, d≠e, e≠a

¬P(a,b), ¬R(a,c)

∀xy.P(x,y)∨R(x,y)



FMF: Example

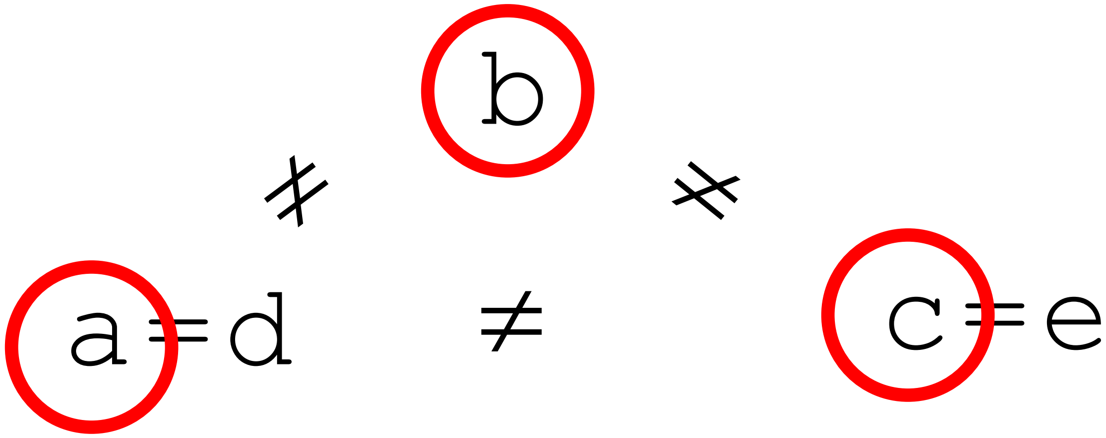
a,b,c,d,e:S

P,R:(S,S)→Bool

a≠b, b≠c, c≠d, d≠e, e≠a

¬P(a,b), ¬R(a,c), **a=d, c=e**

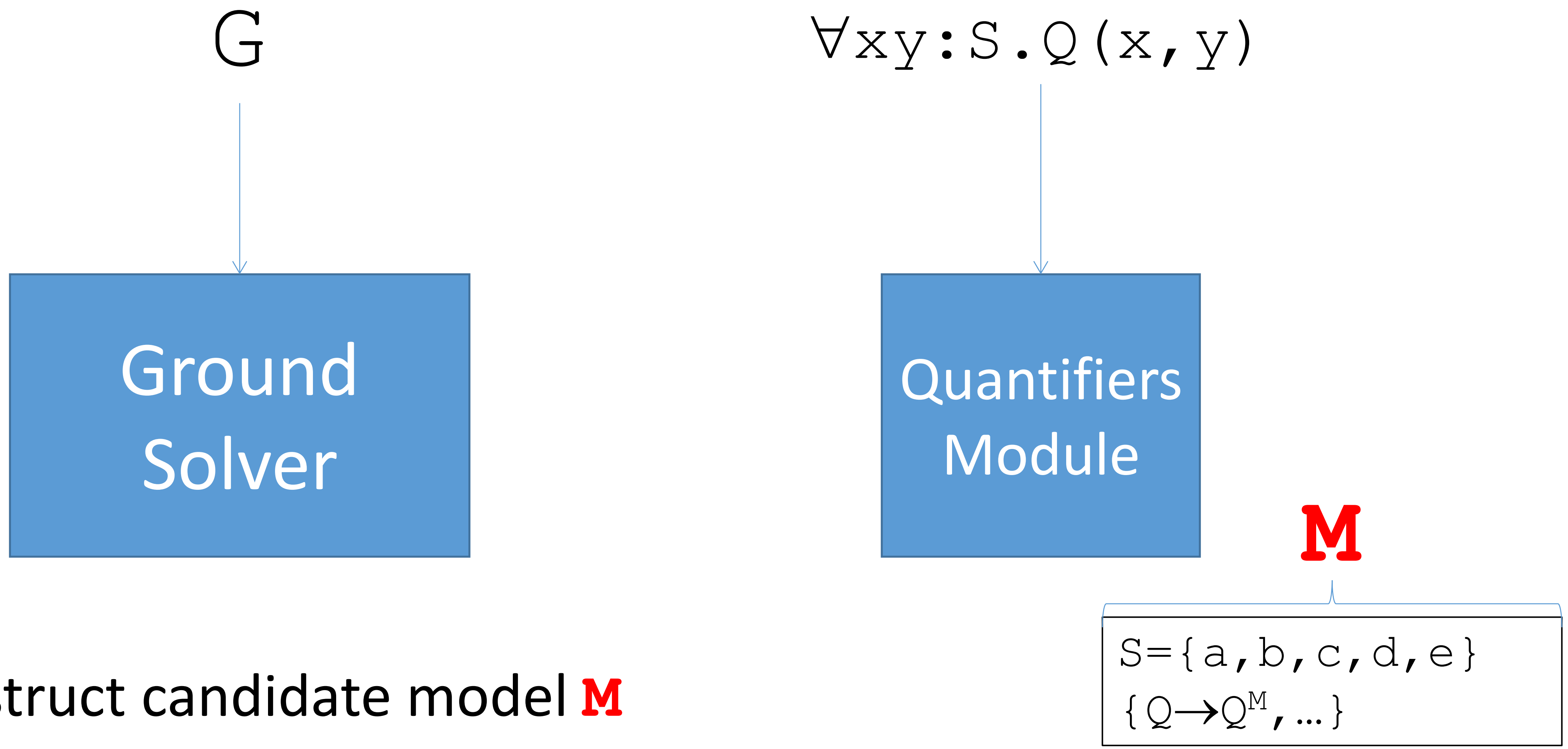
∀xy.P(x,y)∨R(x,y)



S={a,b,c}

EXAMPLE…

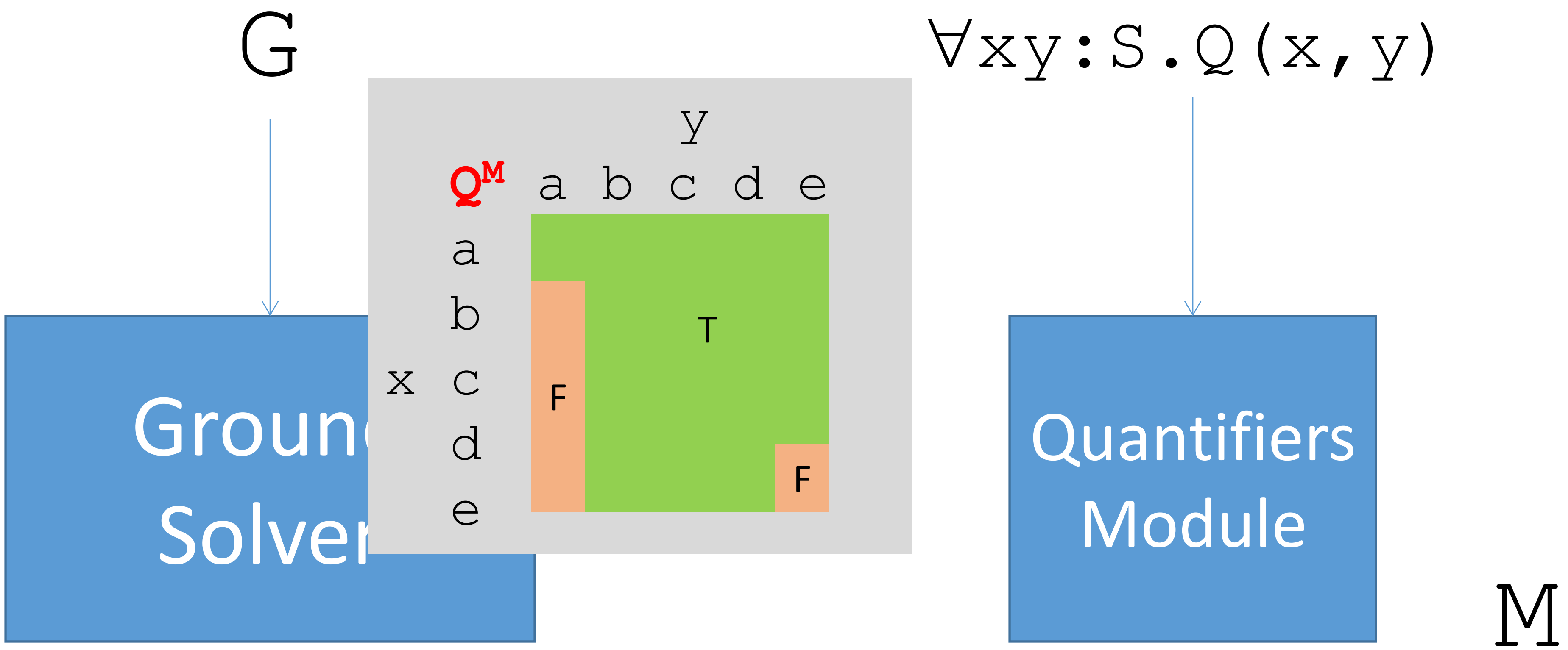
Finite Model Finding : Model-Based Instantiation



- Construct candidate model **M**

[Reynolds/Tinelli/Goel/Krstic/Barrett/Deters CADE13]

Finite Model Finding : Model-Based Instantiation

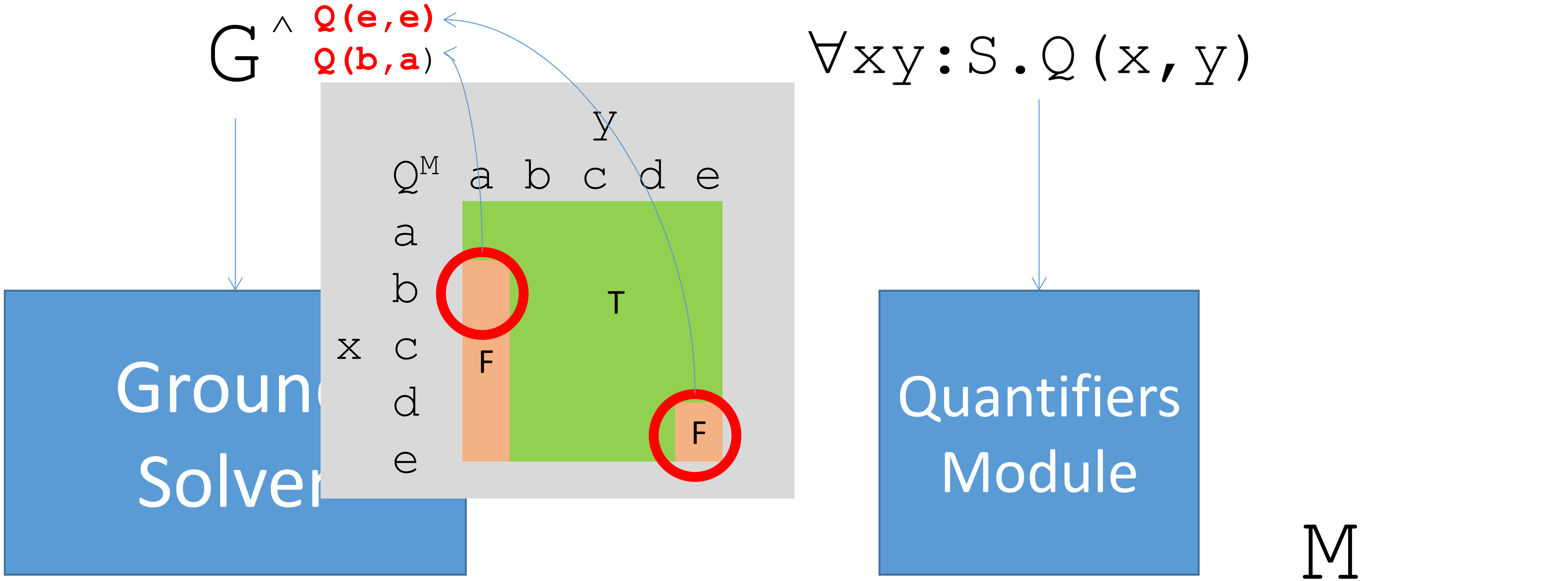


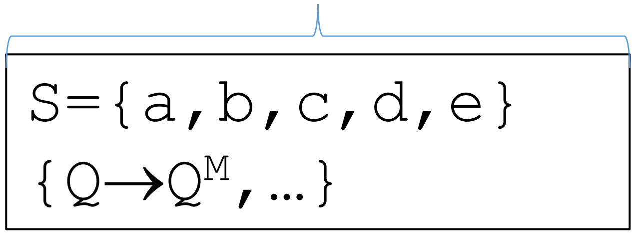
$S=\{a,b,c,d,e\}$
$\{Q \rightarrow Q^M, \ldots\}$

- Evaluate quantified formulas based on Q^M

[Reynolds/Tinelli/Goel/Krstic/Barrett/Deters CADE13]

Finite Model Finding : Model-Based Instantiation





- Only add instances that evaluate to F in Q^M

  $\Rightarrow$ Significantly increased scalability

[Reynolds/Tinelli/Goel/Krstic/Barrett/Deters CADE13]

FMF: Example

```
a,b,c,d,e:S
P,R:(S,S)→Bool
a≠b, b≠c, c≠d, d≠e, e≠a
¬P(a,b), ¬R(a,c)
∀xy.P(x,y)∨R(x,y)
```

FMF: Example

a,b,c,d,e:S

P,R:(S,S)→Bool

$a \neq b,\ b \neq c,\ c \neq d,\ d \neq e,\ e \neq a$

$\neg P(a,b),\ \neg R(a,c)$

$\forall xy.\, P(x,y) \vee R(x,y)$

$P := \lambda xy.(x \neq a \vee y \neq b)$

$R := \lambda xy.(x \neq a \vee y \neq c)$

				y		
	P^M	a	b	c	d	e
	a		F			
	b					
x	c			T		
	d					
	e					

				y		
	R^M	a	b	c	d	e
	a			F		
	b					
x	c			T		
	d					
	e					

FMF: Example

$a,b,c,d,e : S$

$P,R : (S,S) \to Bool$

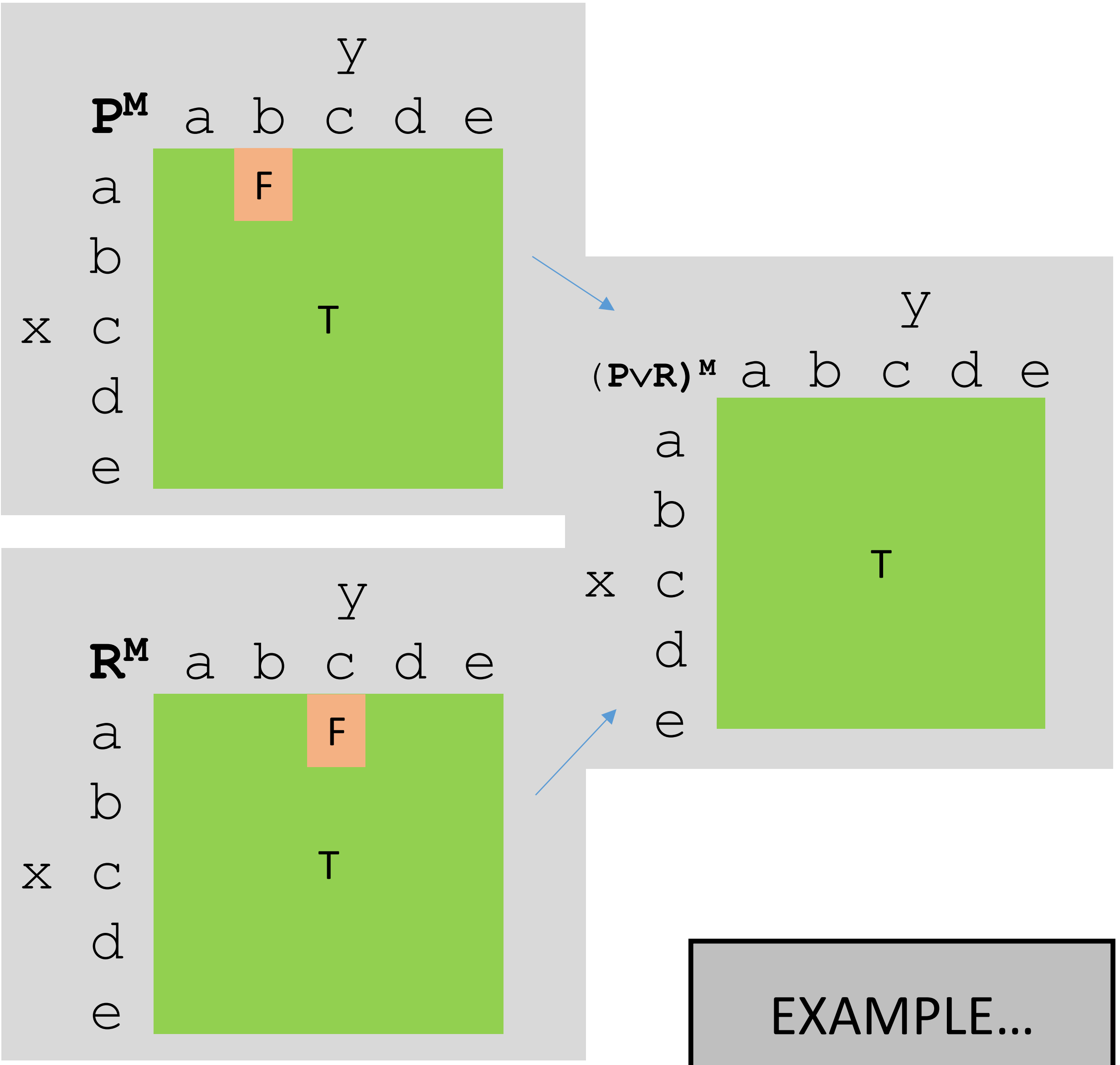
$a \neq b,\ \ b \neq c,\ \ c \neq d,\ \ d \neq e,\ \ e \neq a$

$\neg P(a,b),\ \ \neg R(a,c)$

$\forall xy.\, P(x,y) \vee R(x,y)$

$P := \lambda xy.\,(x \neq a \vee y \neq b)$

$R := \lambda xy.\,(x \neq a \vee y \neq c)$



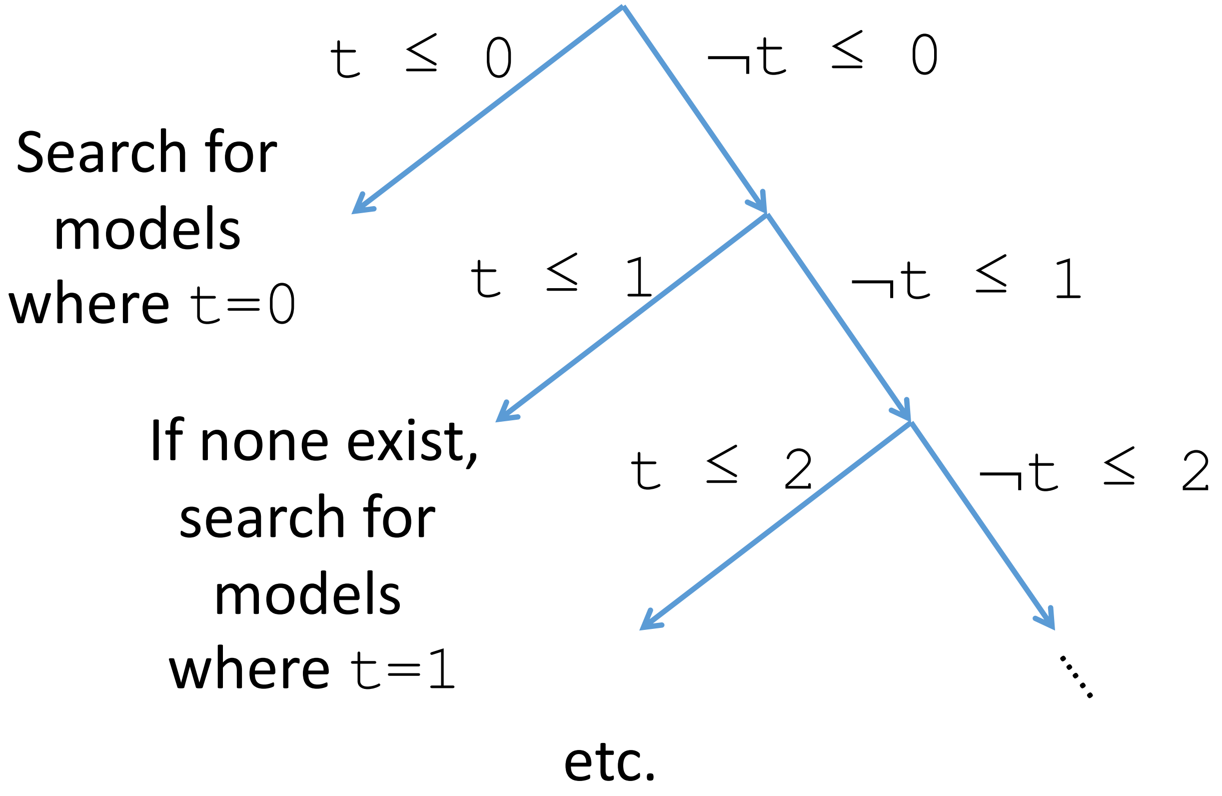
EXAMPLE…

Finite Model Finding in CVC4

- **Sound** for both “sat” and “unsat”
- **Finite-model complete**
  - If there is a finite model, CVC4 will eventually find it
    (when all quantification is over sorts that are interpreted as finite)
- Refutationally **incomplete** in general
  - But regardless, is often able to answer “unsat”

Extension: Bounded Integer Quantification

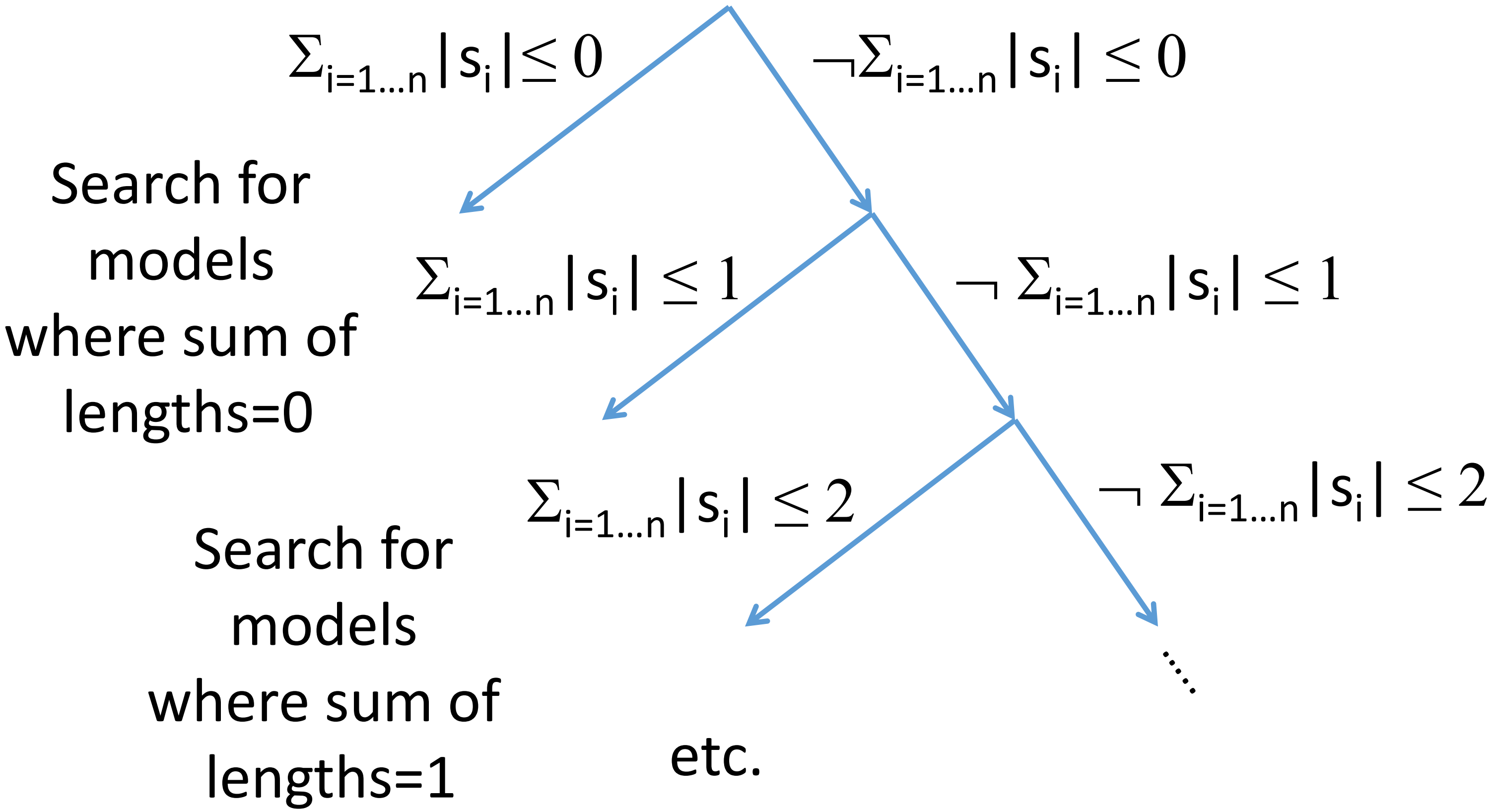
- $\forall$x:Int. 0 ≤ x < t ⇒P(x)



EXAMPLE...

Extension: Bounded Length Strings

- Given input $F[s_1,\dots,s_n]$ for strings $s_1 \dots s_n$:



EXAMPLE…

Synthesis: Motivation

- Synthesis Problem : $\underbrace{\exists \mathtt{f}}_{\text{There exists a function f}} . \underbrace{\forall \mathtt{x} . }_{\text{such that for all } \mathtt{x},\ \mathtt{P(f,x)}} \mathtt{P(f,x)}$

- Most existing approaches for synthesis
  - Rely on specialized solver that makes subcalls to an SMT Solver
- *CVC4 has approach for synthesis, which is entirely inside SMT solver*

Example : Max of Two Integers

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

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

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

How does an SMT solver handle Max example?

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

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

How does an SMT solver handle Max example?

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

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

- 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?

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

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?

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

Convert to first-order

$$\forall xy.\ \exists g. (g \geq x \land g \geq y \land (g = x \lor g = y))$$

[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

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

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

    $\Rightarrow$ CVC4 has specialized instantiation procedure

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

Max Example

$$\forall xy.\ \exists g.(g \geq x \land g \geq y \land (g = x \lor g = y))$$

Ground Solver

Quantifiers Module

Max Example

`∀xy. ∃g.isMax(g,x,y)`

Ground Solver

Quantifiers Module

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

Max Example

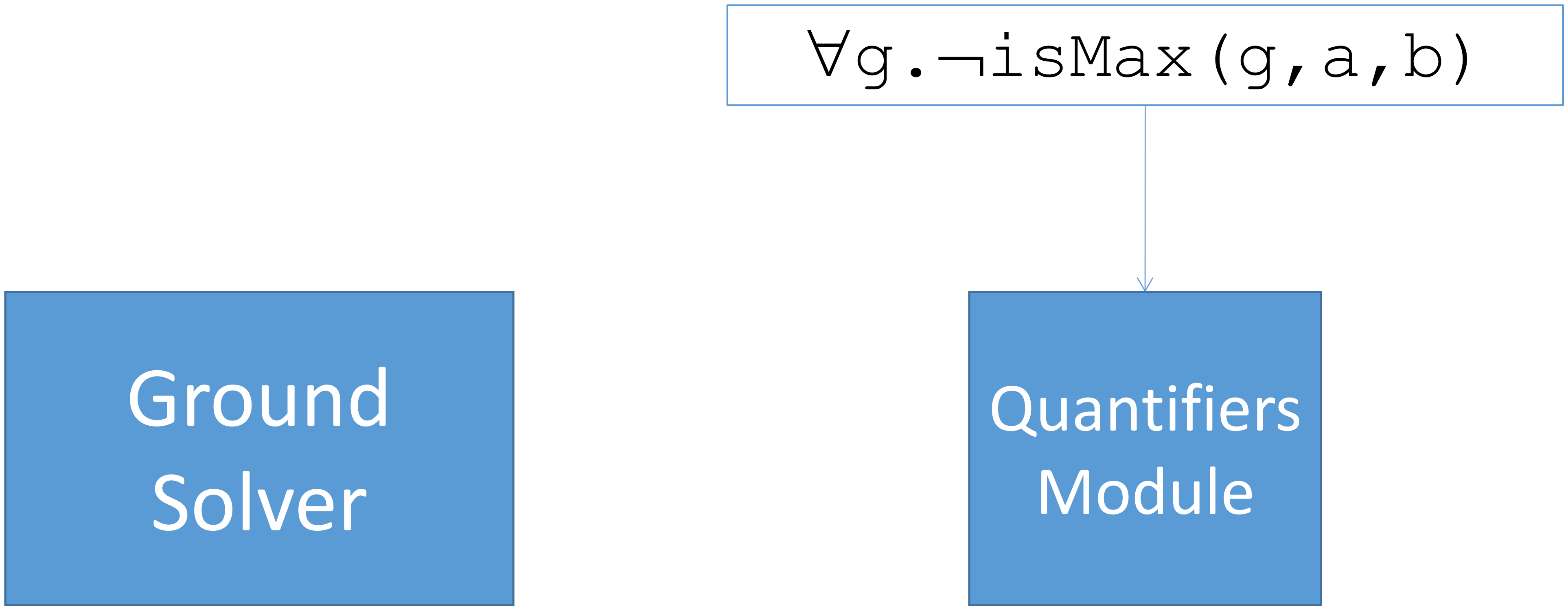
$\forall g. \neg \mathrm{isMax}(g, \mathbf{a}, \mathbf{b})$

Ground Solver

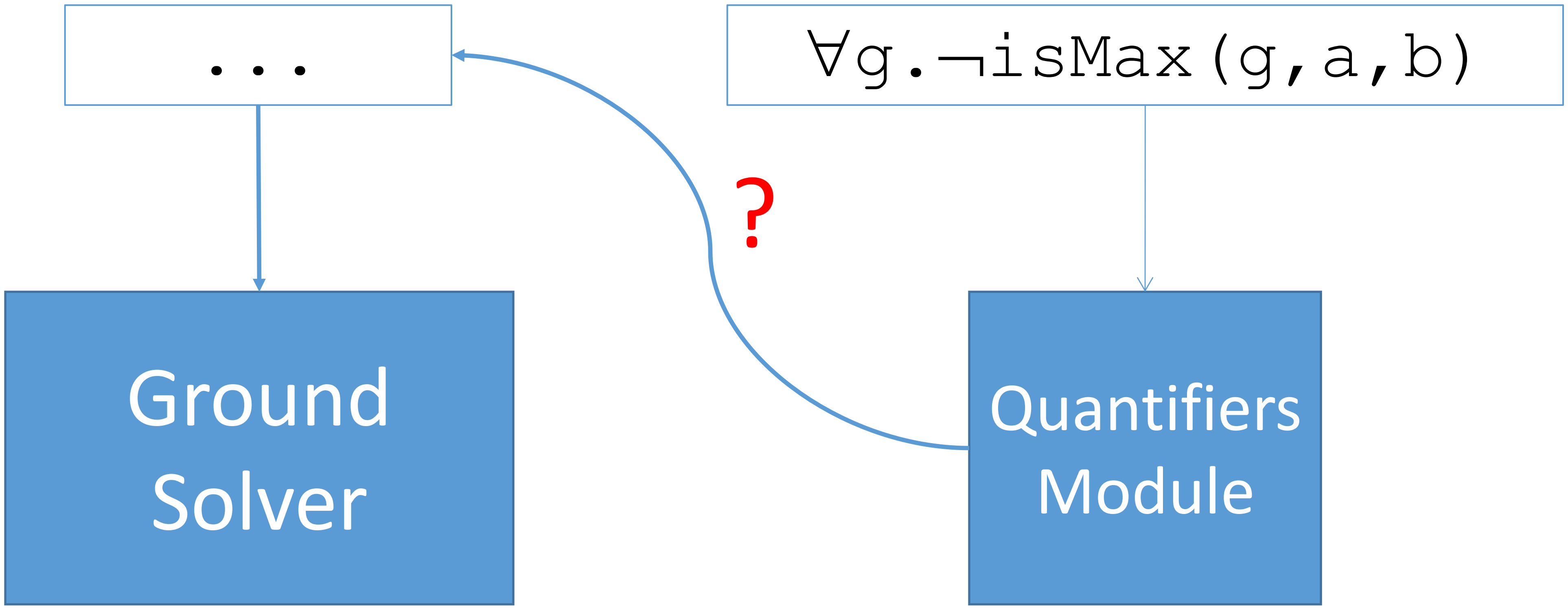
Quantifiers Module

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

Max Example

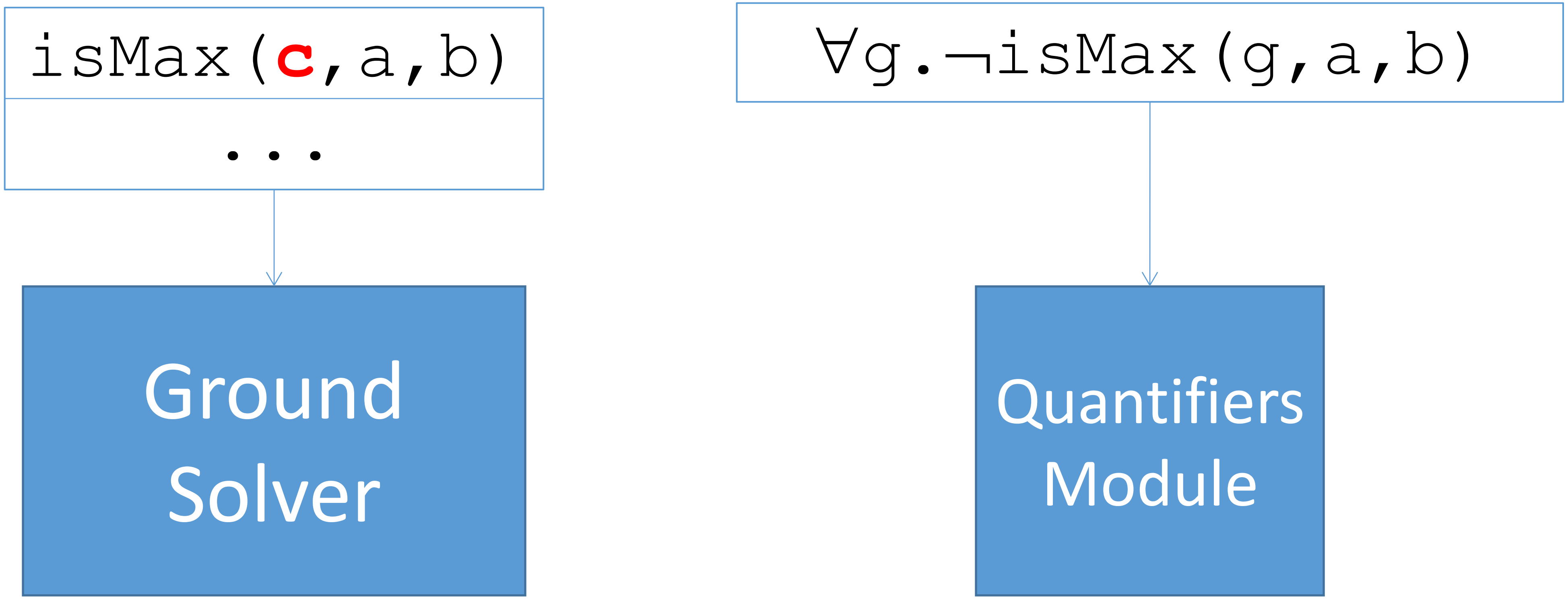


Max Example



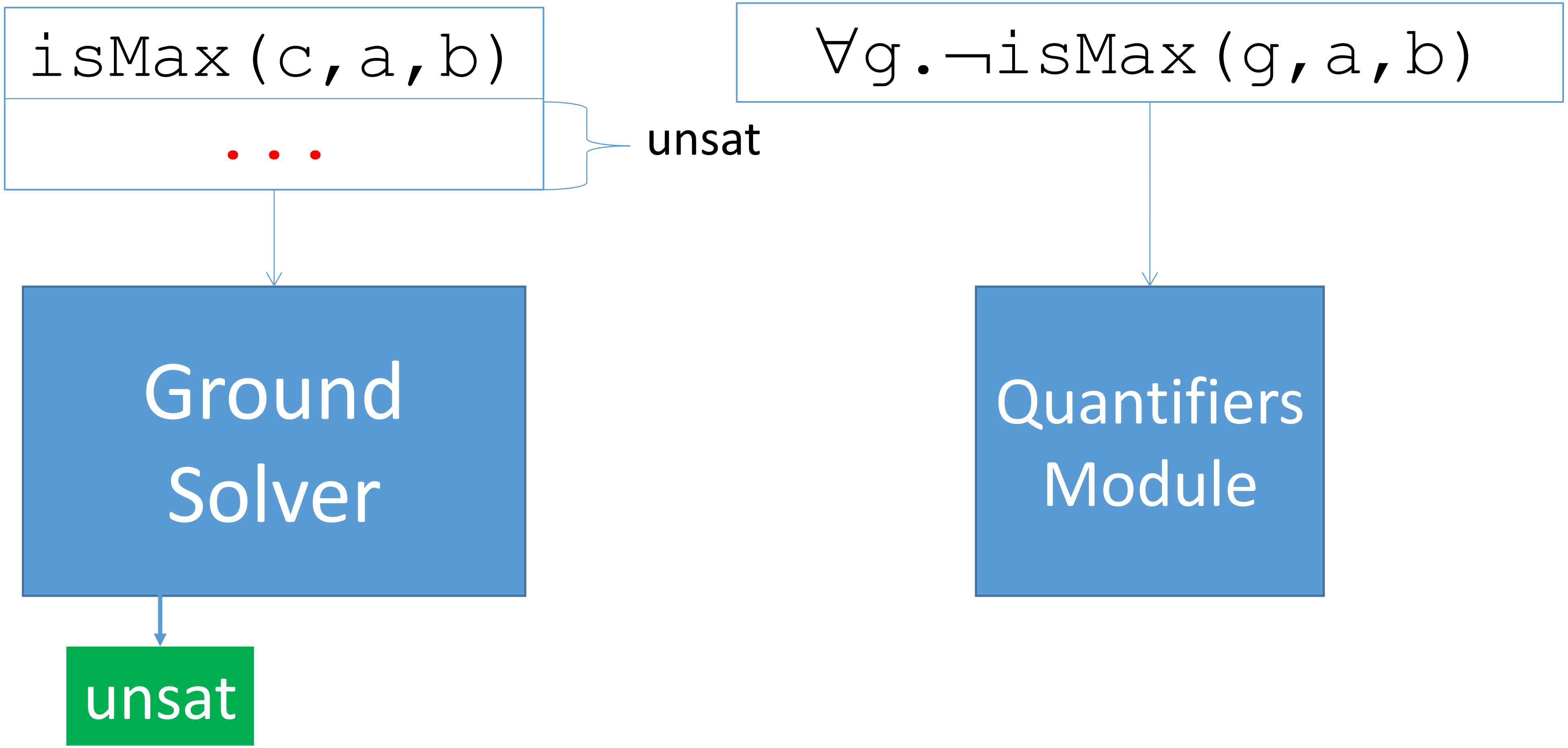
- Which instances of $\forall g.\neg \texttt{isMax(g,a,b)}$ do we consider?

Counterexample-Guided Instantiation



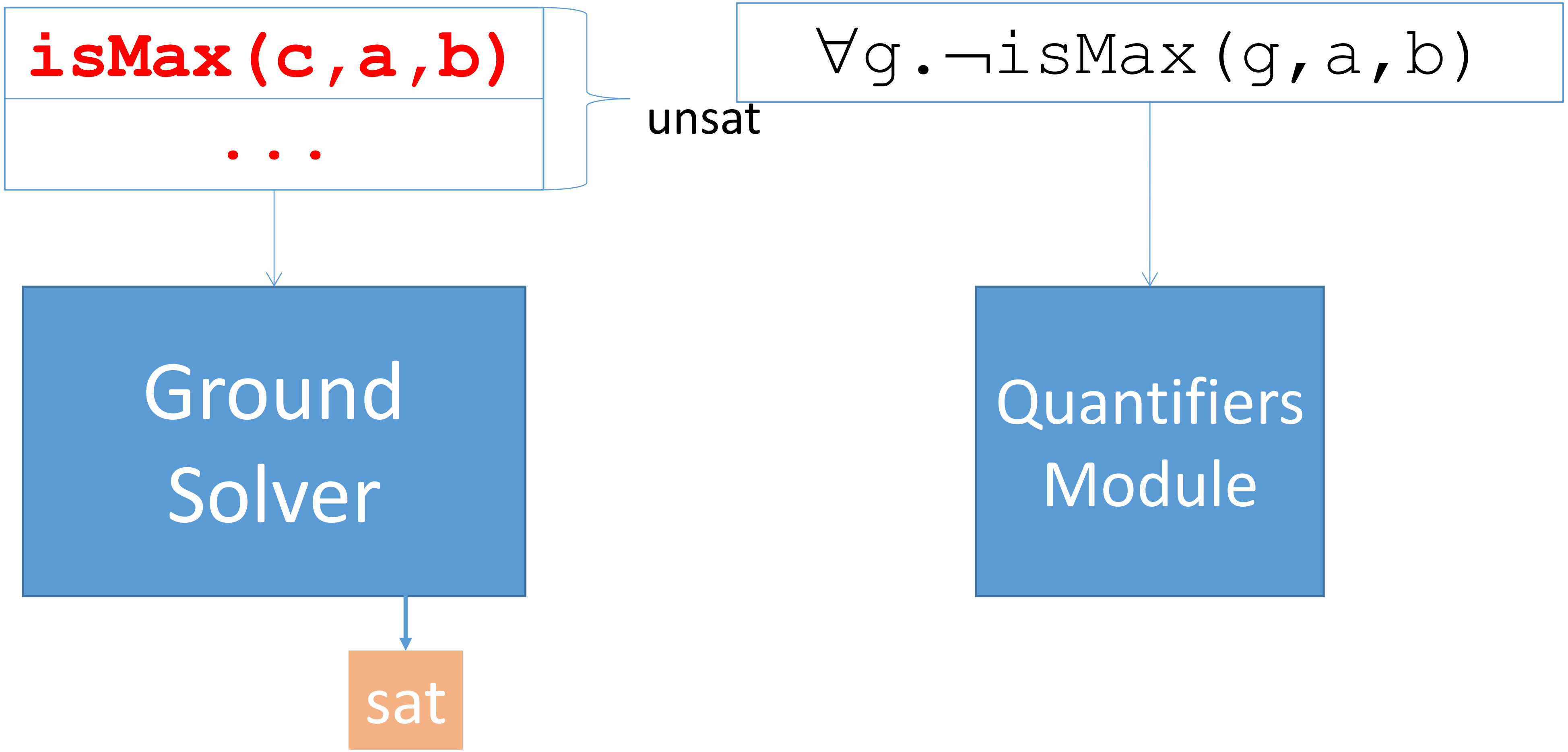
- **Idea**: choose instances of $\forall$`g.`$\neg$`isMax(g,a,b)` based on models for "counterexample" fresh constant **`c`**

Counterexample-Guided Instantiation



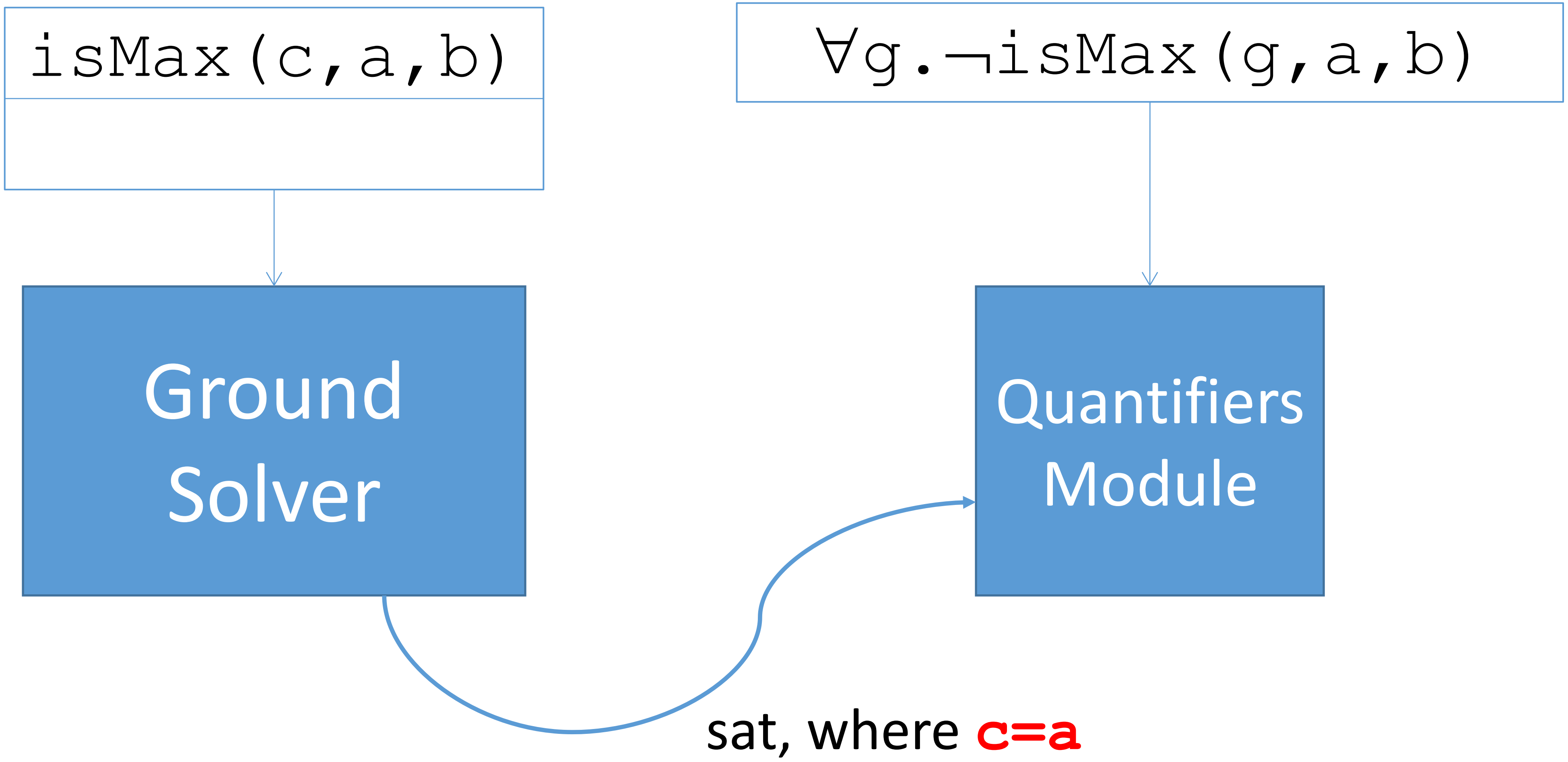
- If ground constraints without CE is unsat, answer “unsat”

Counterexample-Guided Instantiation

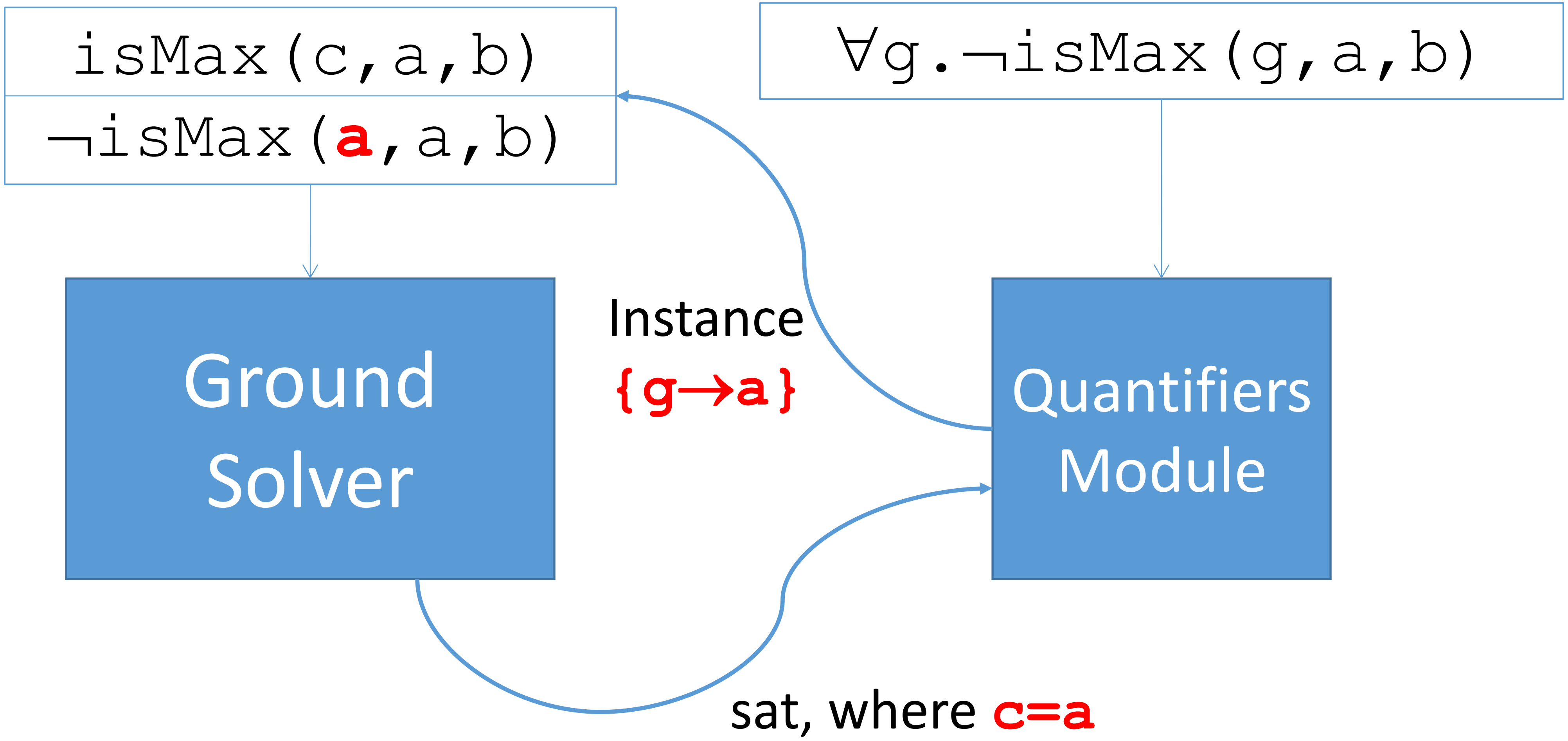


- Else, if ground constraints with CE is unsat, answer "sat"

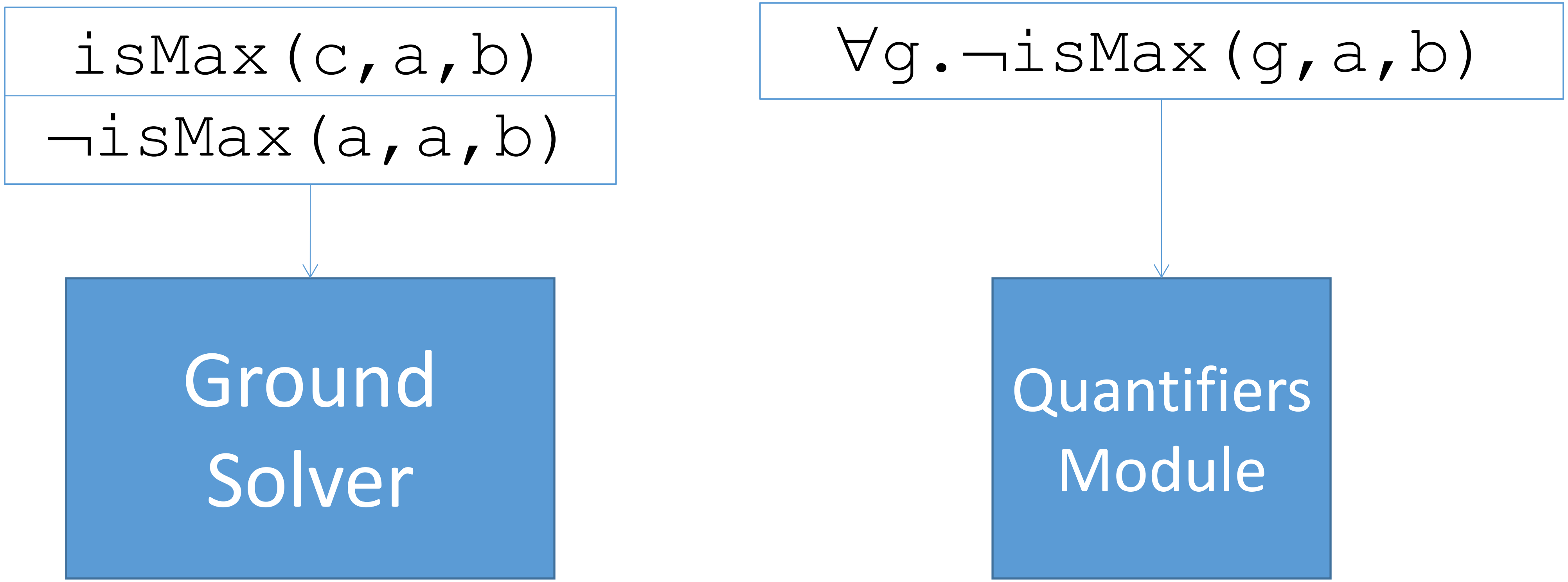
Counterexample-Guided Instantiation



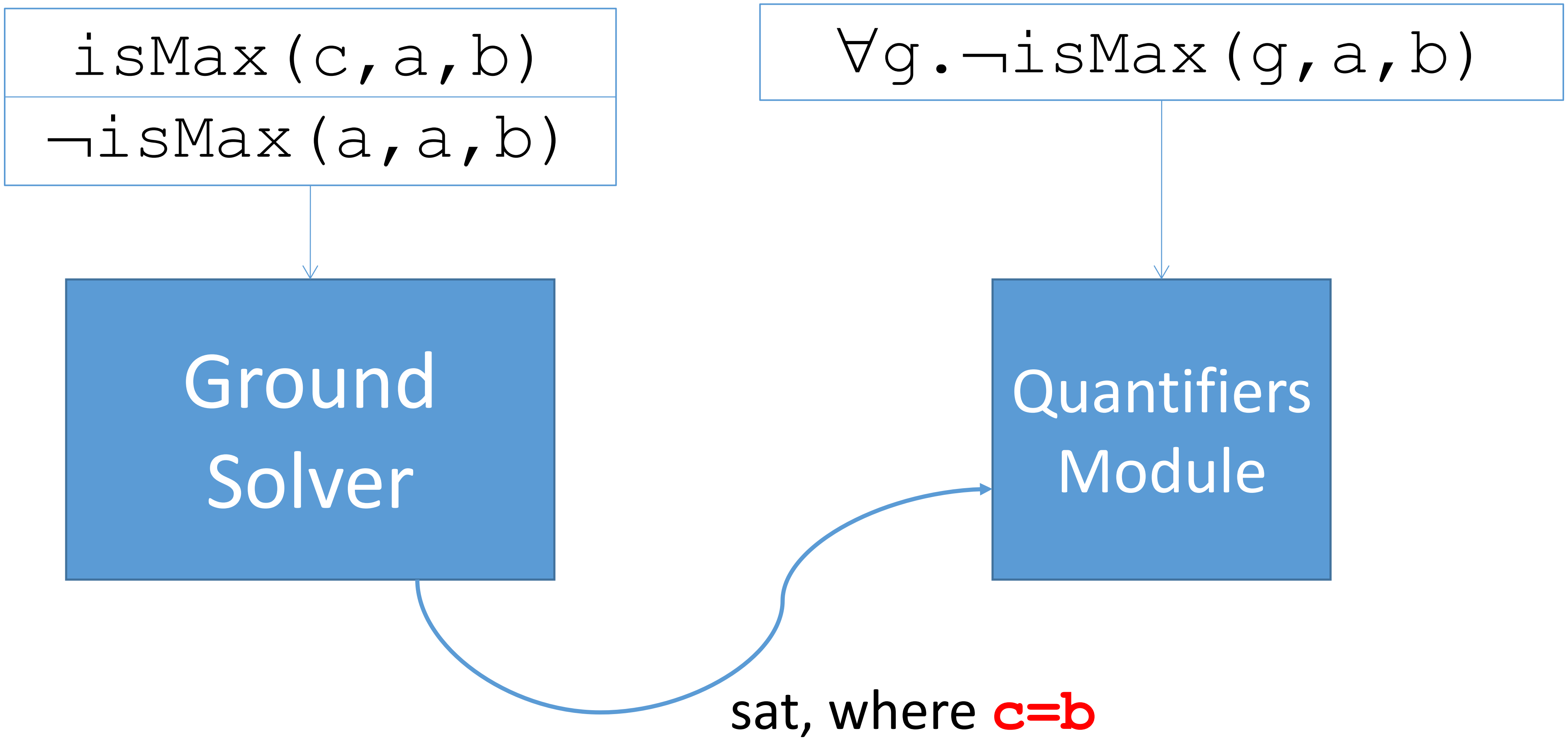
Counterexample-Guided Instantiation



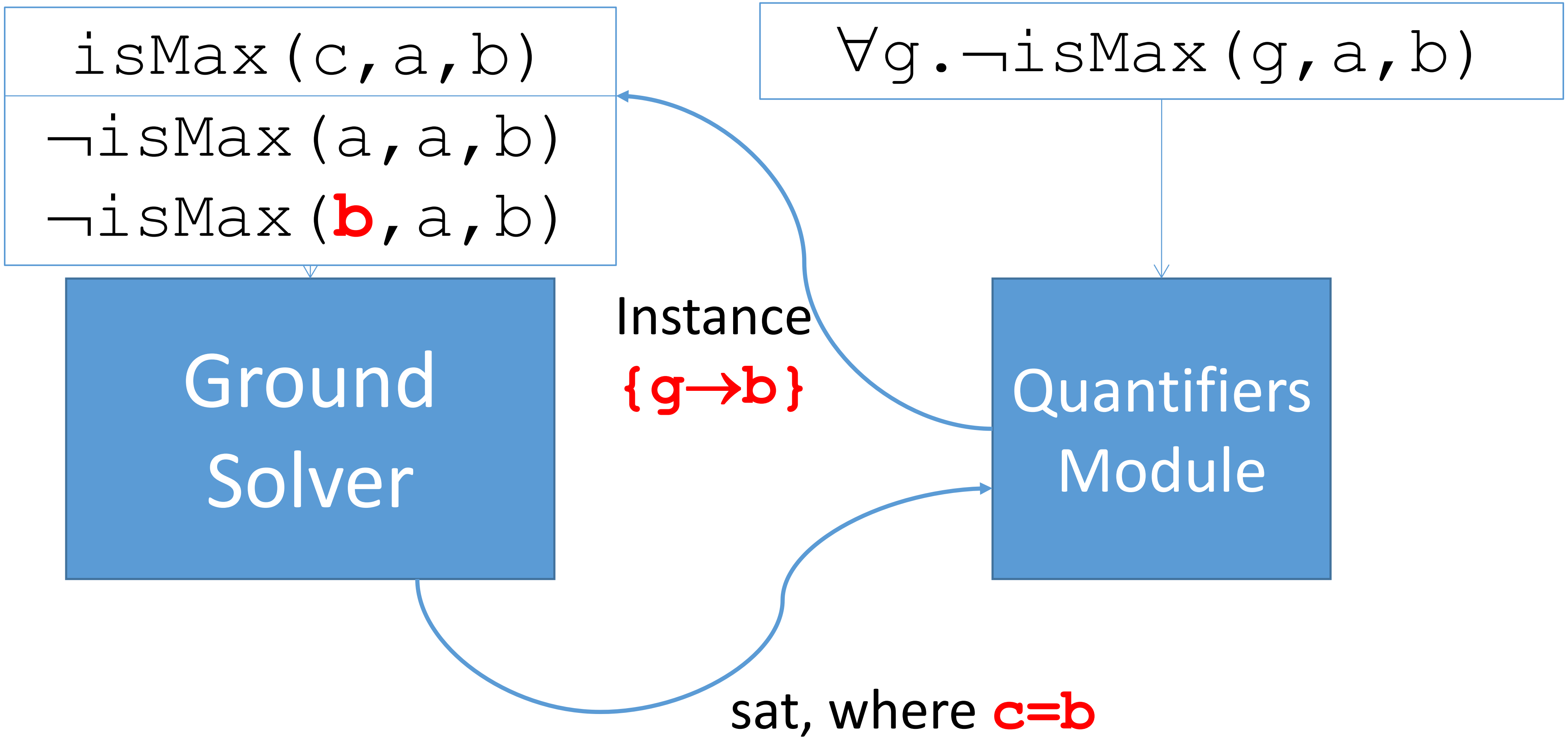
Counterexample-Guided Instantiation



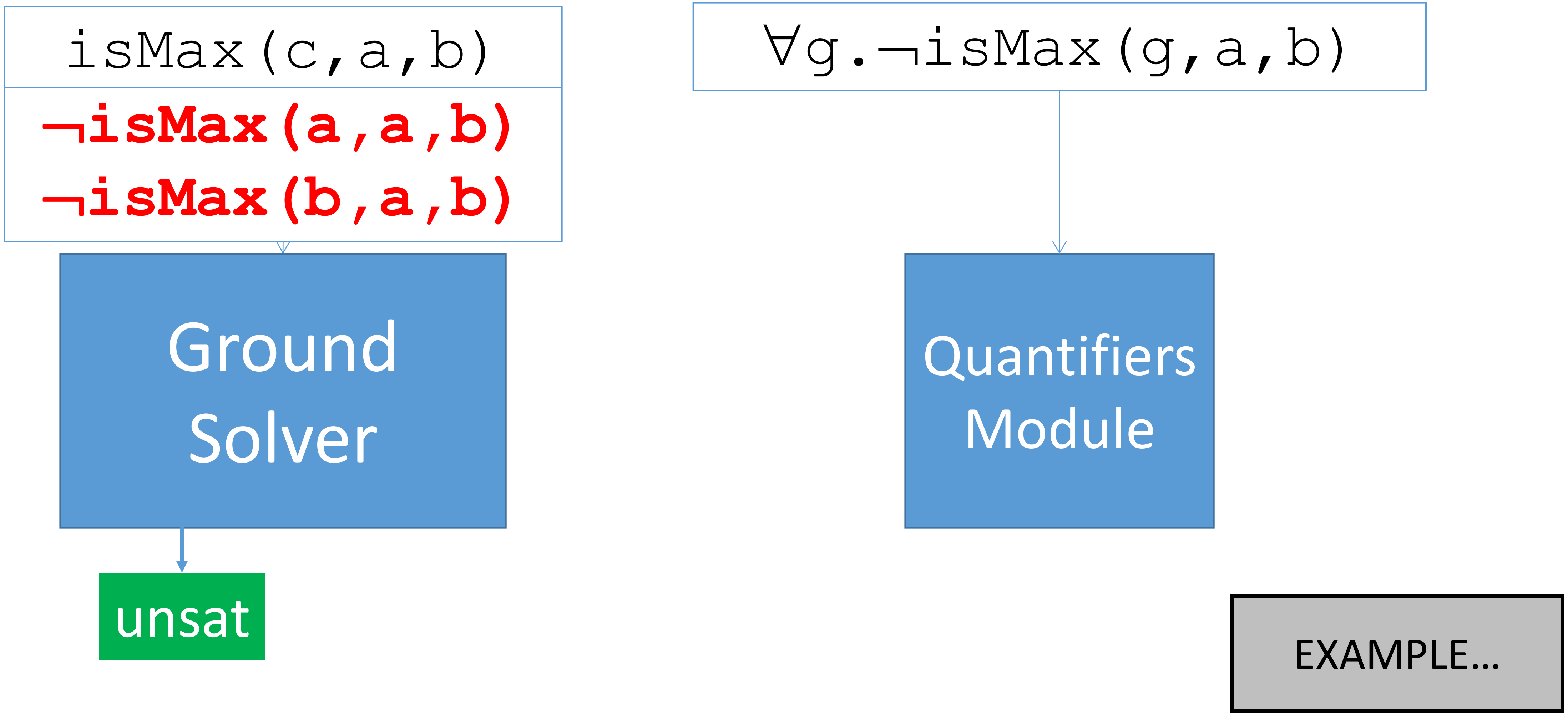
Counterexample-Guided Instantiation



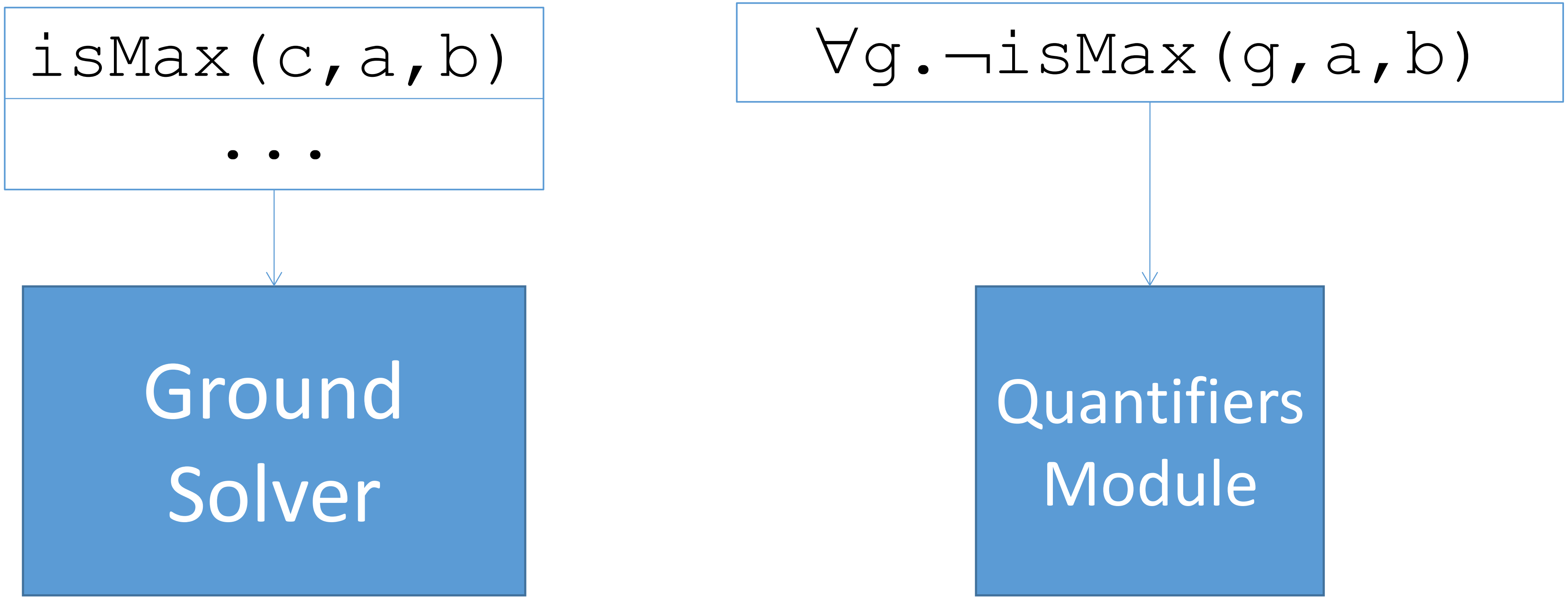
Counterexample-Guided Instantiation



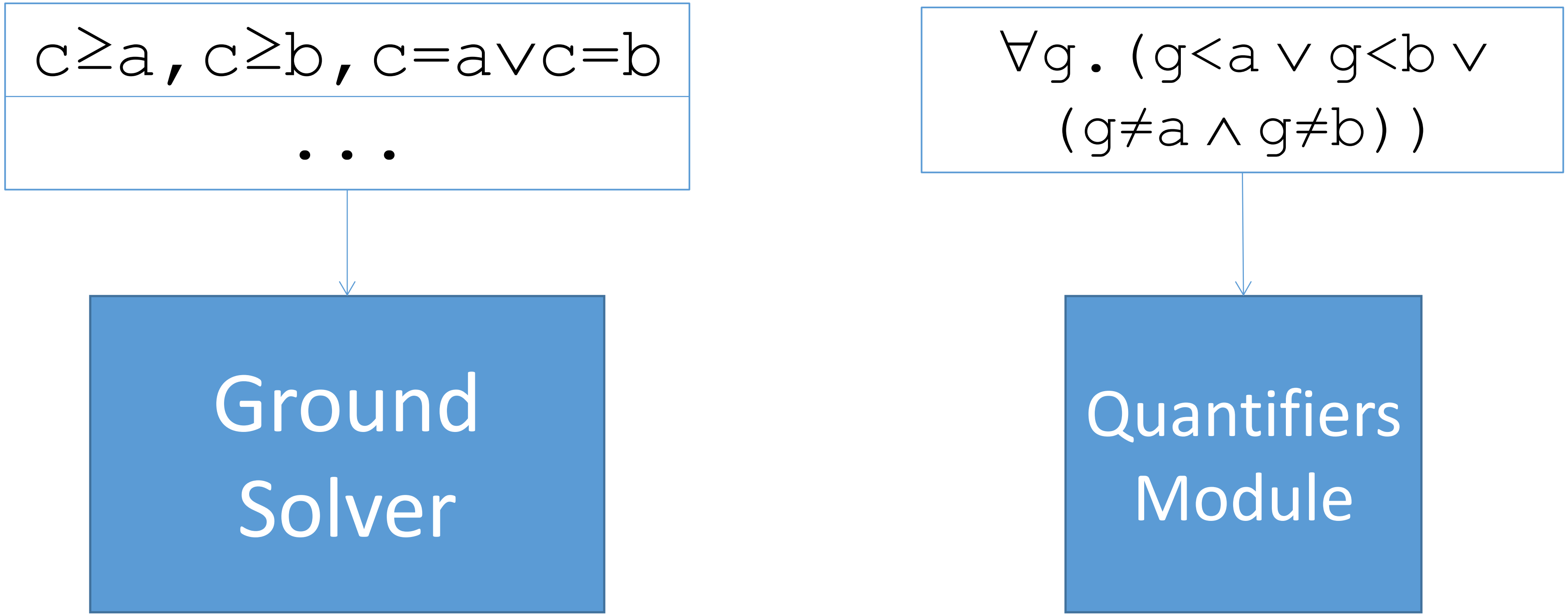
Counterexample-Guided Instantiation



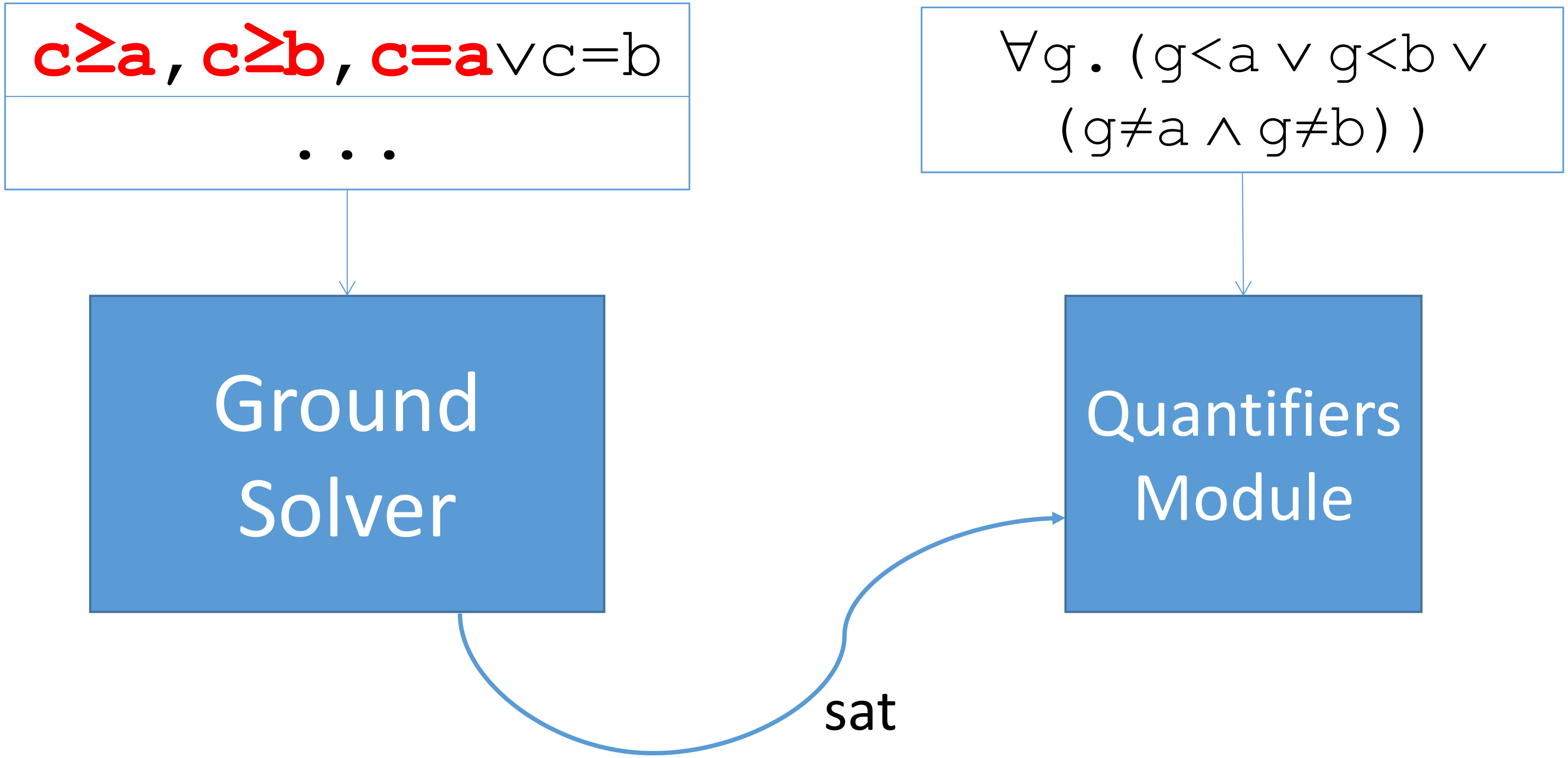
Counterexample-Guided Instantiation



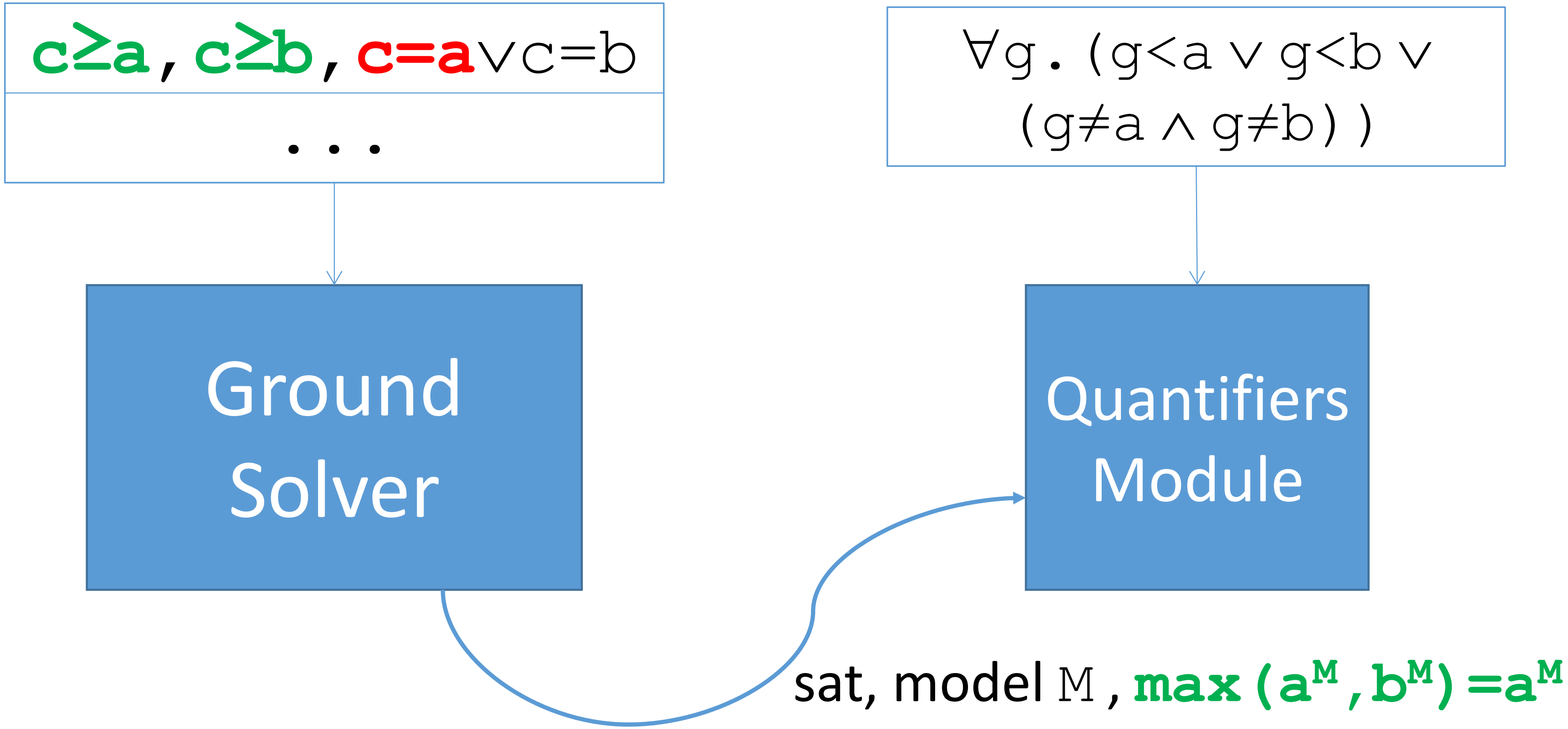
Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

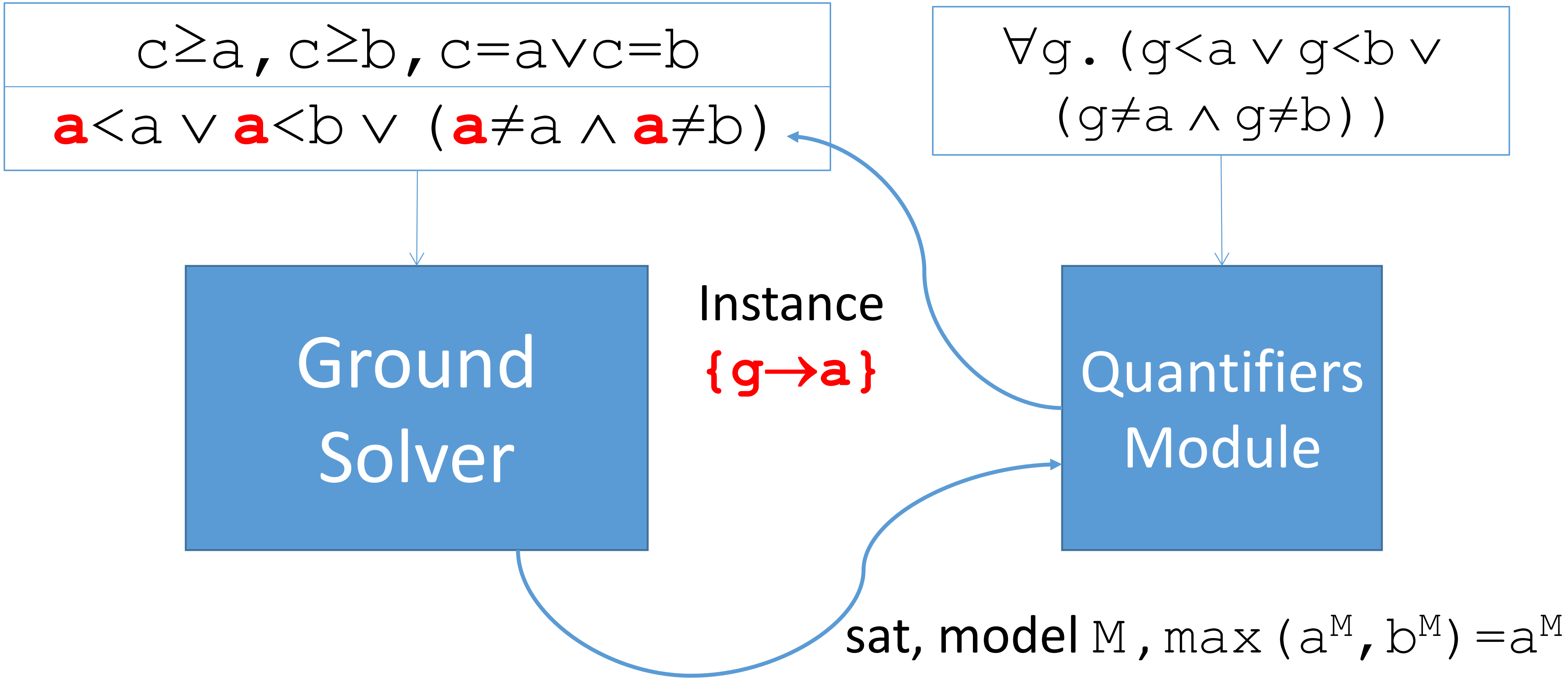


Counterexample-Guided Instantiation

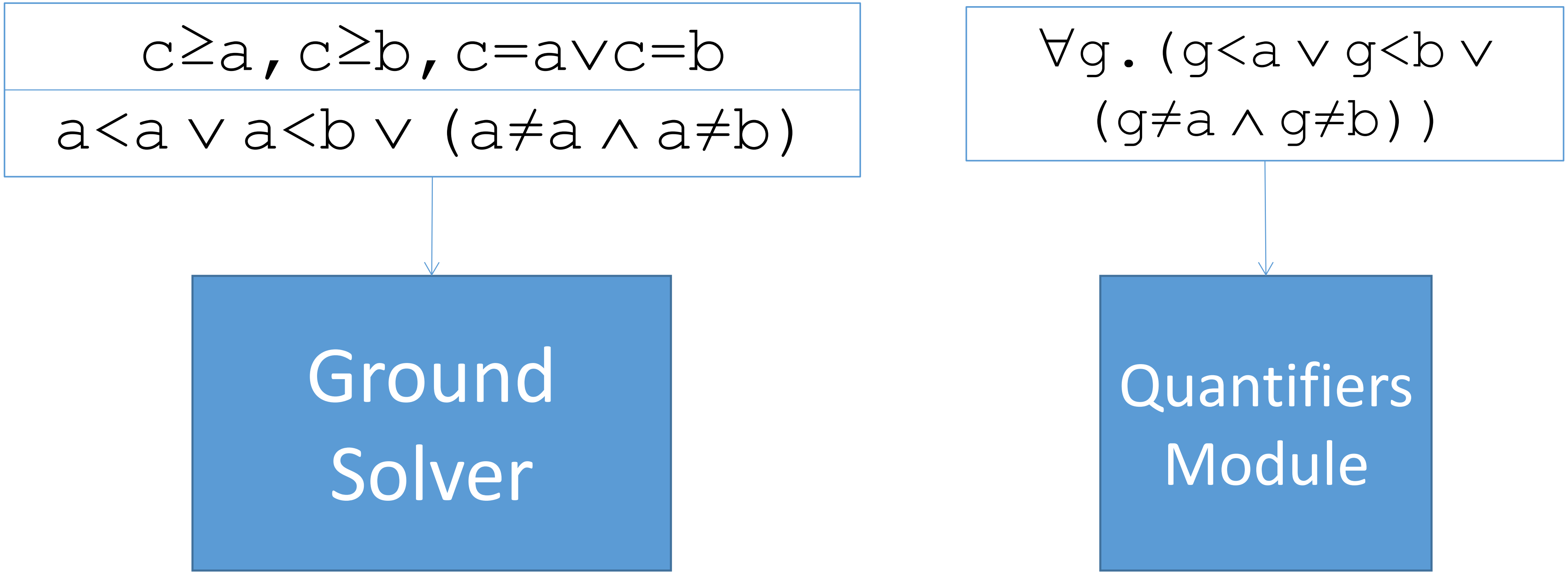


- Take maximal lower bound for c in model M

Counterexample-Guided Instantiation



Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

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

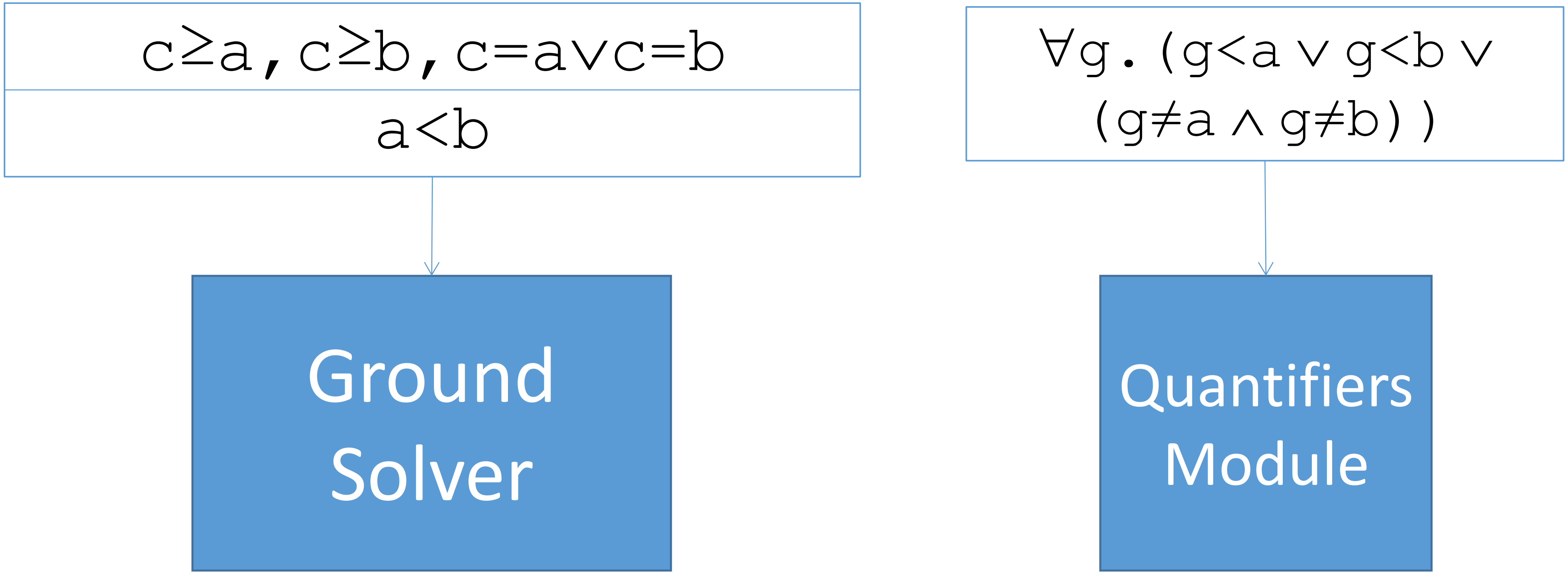
$\cancel{a < a} \vee a < b \vee (\cancel{a \neq a \wedge a \neq b})$

Ground Solver

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

Quantifiers Module

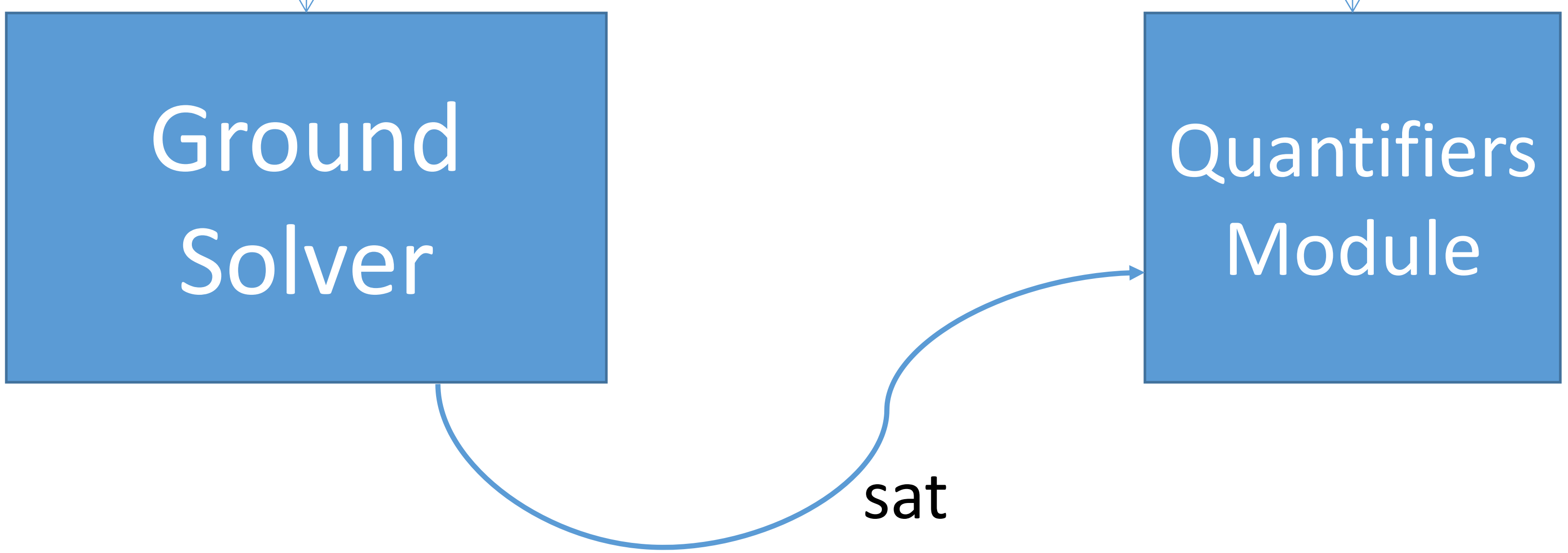
Counterexample-Guided Instantiation



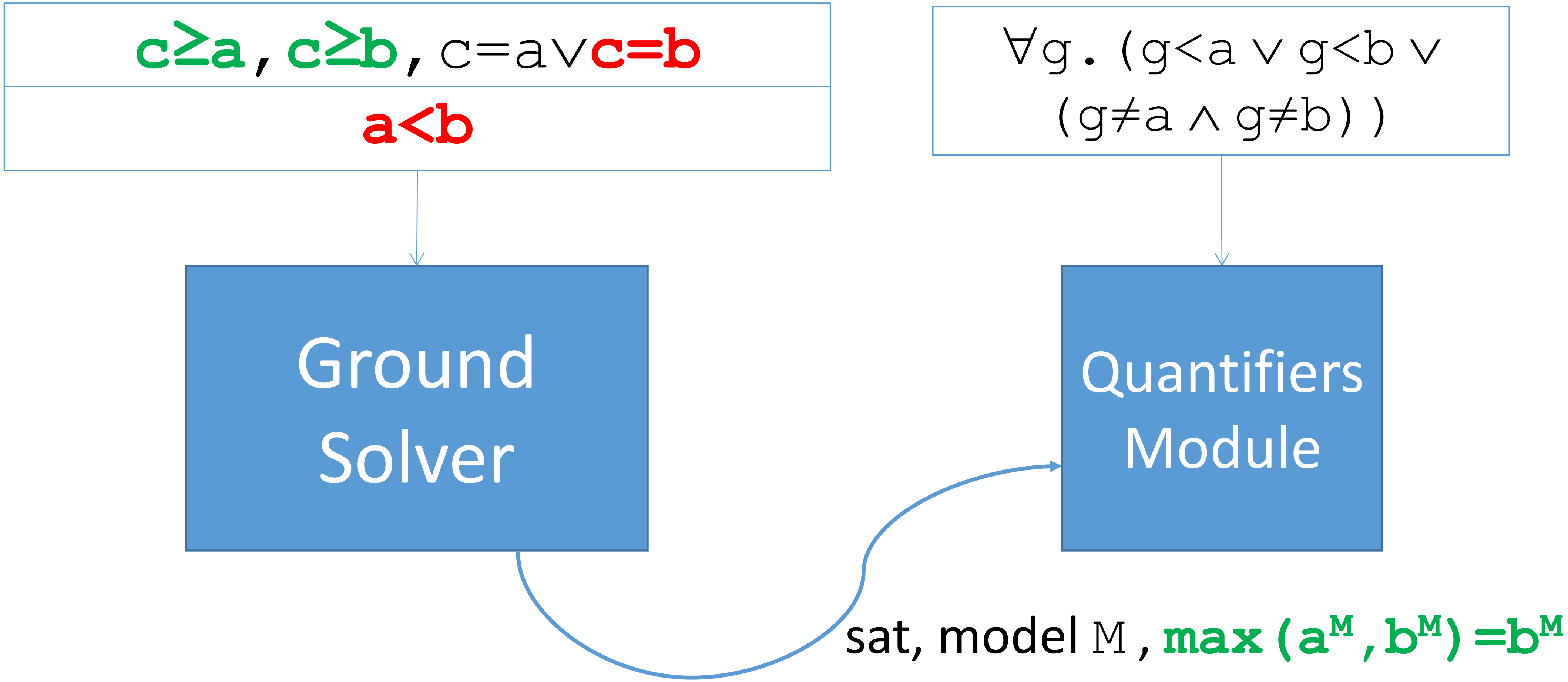
Counterexample-Guided Instantiation

$$\frac{c \geq a,\ c \geq b,\ c = a \vee c = b}{a < b}$$

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

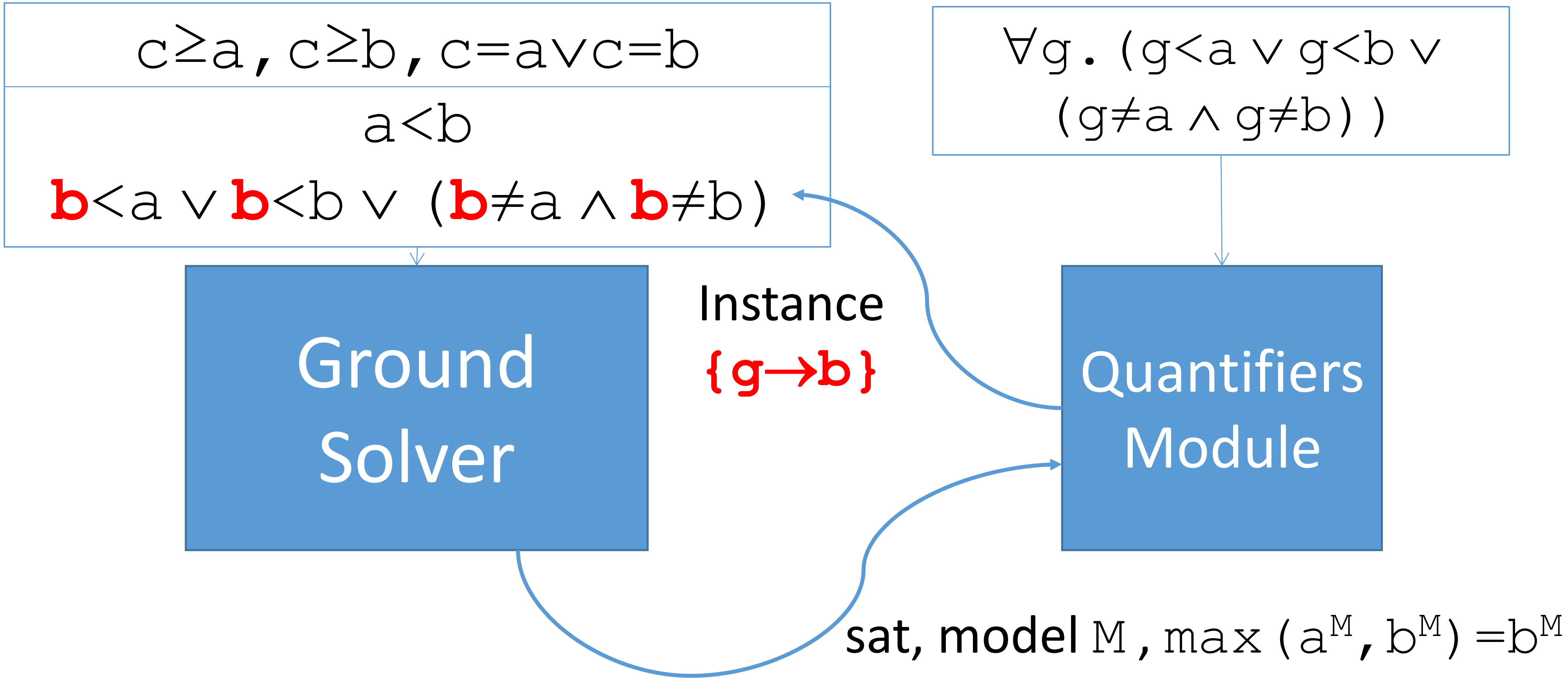


Counterexample-Guided Instantiation



- Take maximal lower bound for c in model M

Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

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

$\mathbf{a < b}$

$\mathbf{b < a}$

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

Ground Solver

Quantifiers Module

unsat

Synthesis: Solutions

`∃f. ∀xy.isMax(f(x,y),x,y)`

Ground Solver

Quantifiers Module

Synthesis: Solutions

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

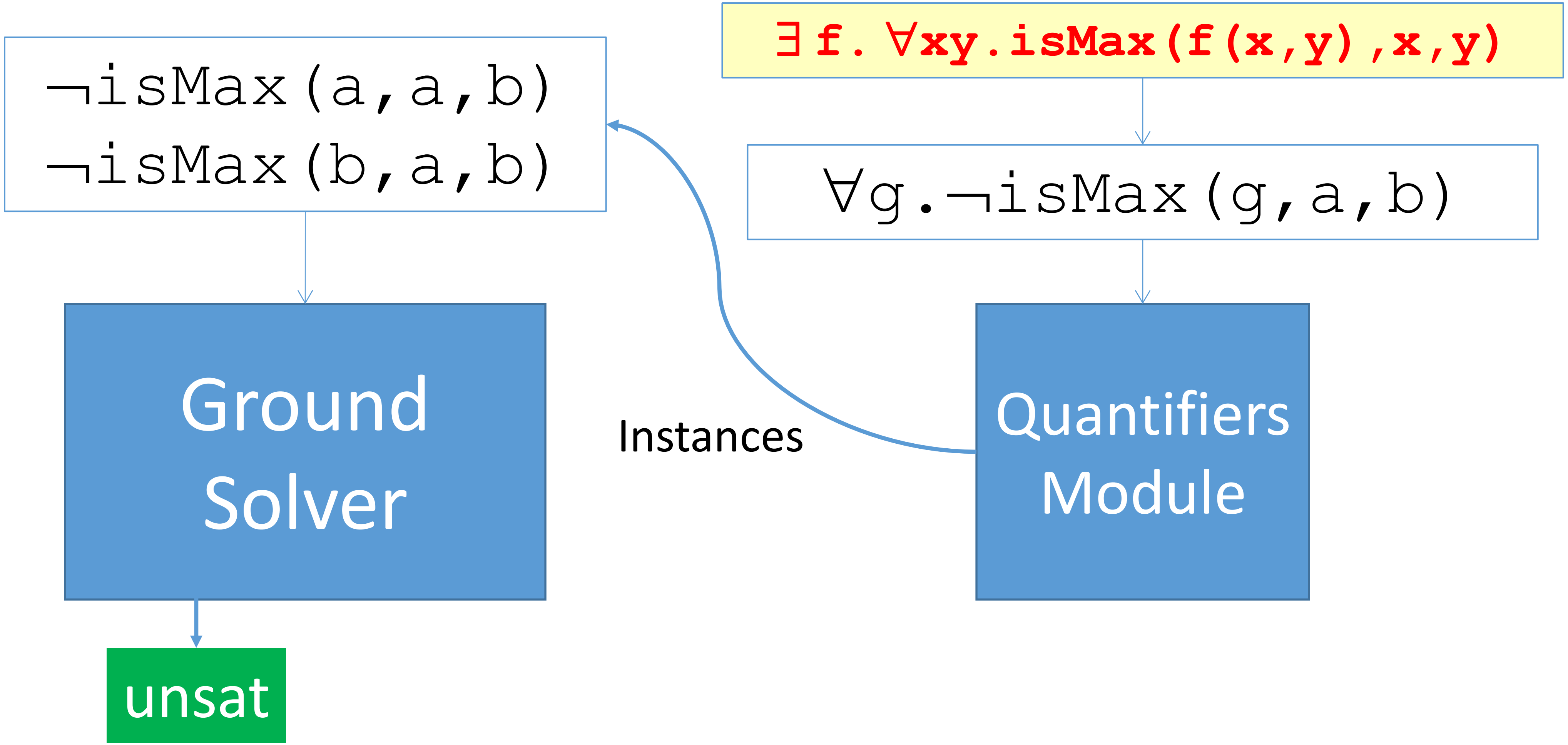
Negate, convert to FO

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

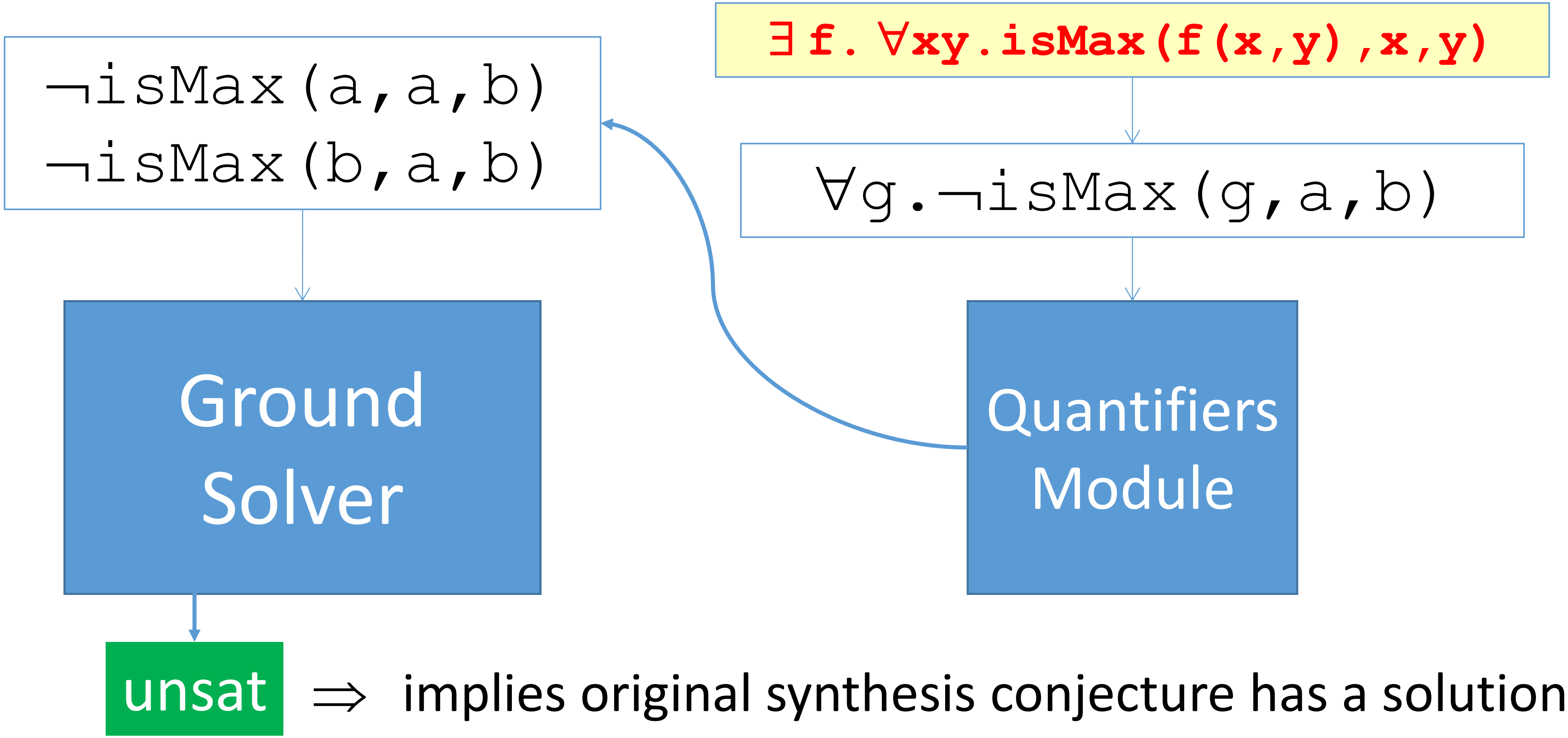
Ground Solver

Quantifiers Module

Synthesis: Solutions

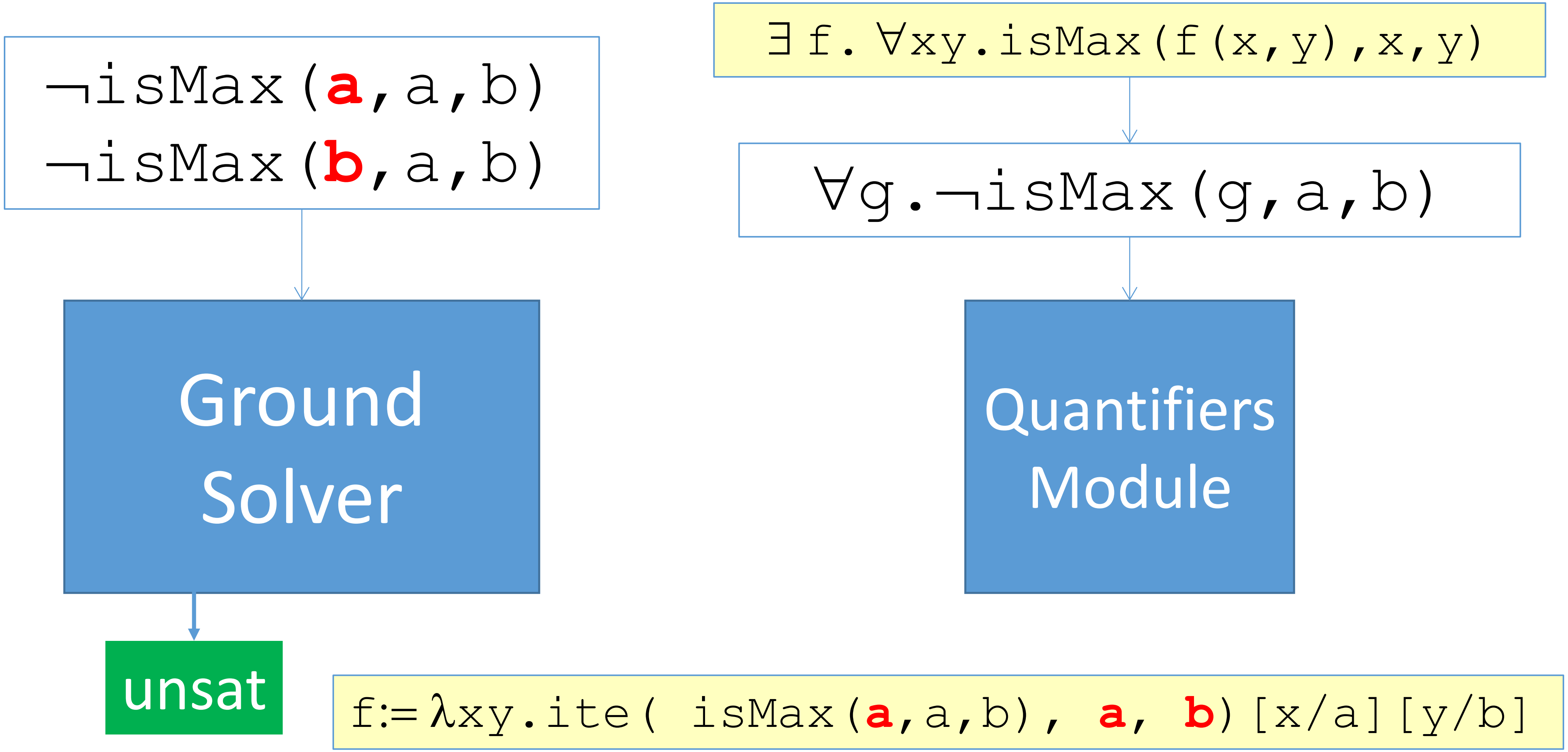


Synthesis: Solutions



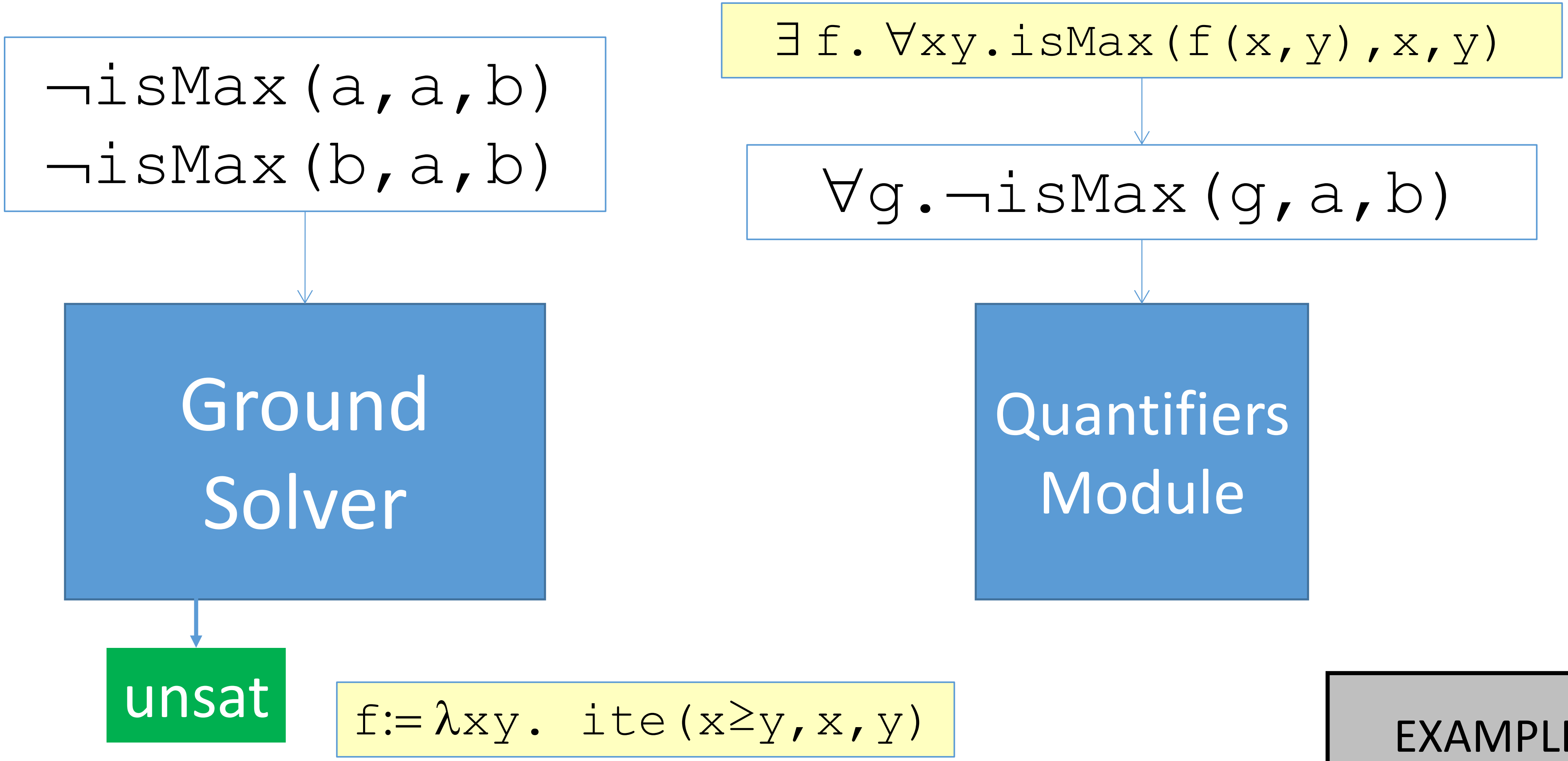
$\Rightarrow$ implies original synthesis conjecture has a solution

Synthesis: Solutions



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

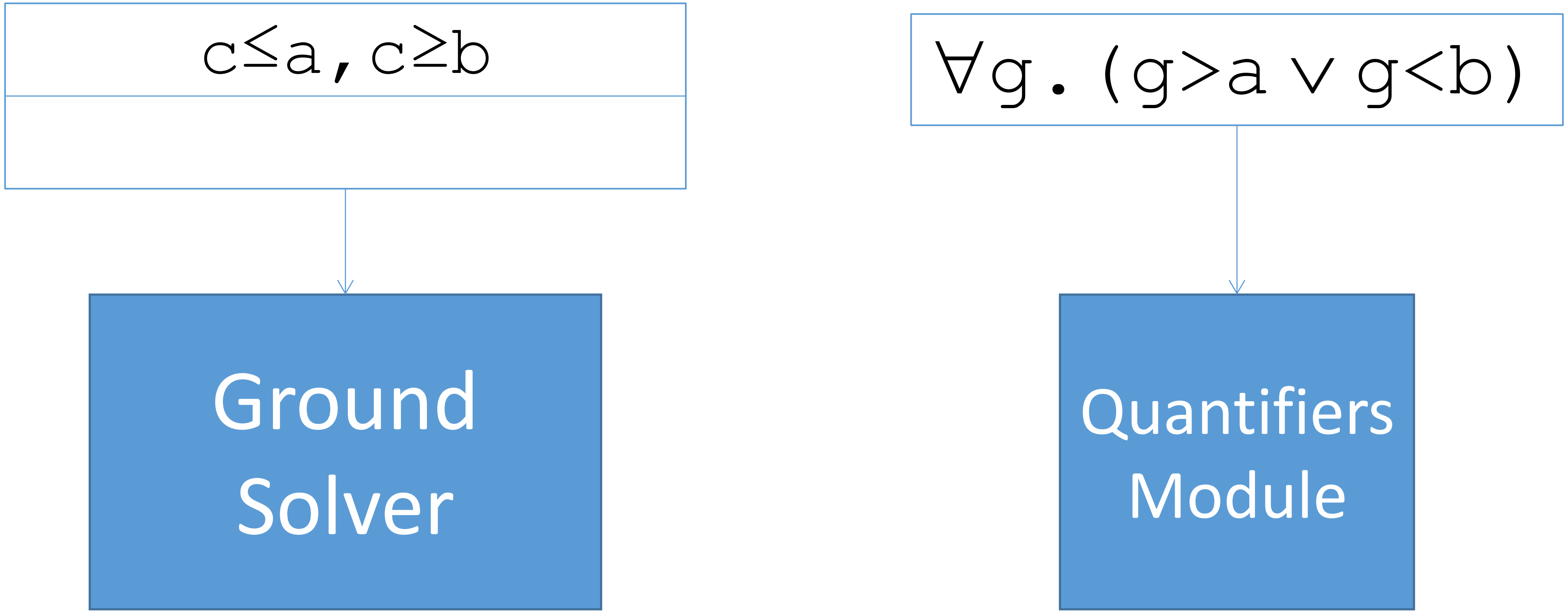
Synthesis: Solutions



EXAMPLE…

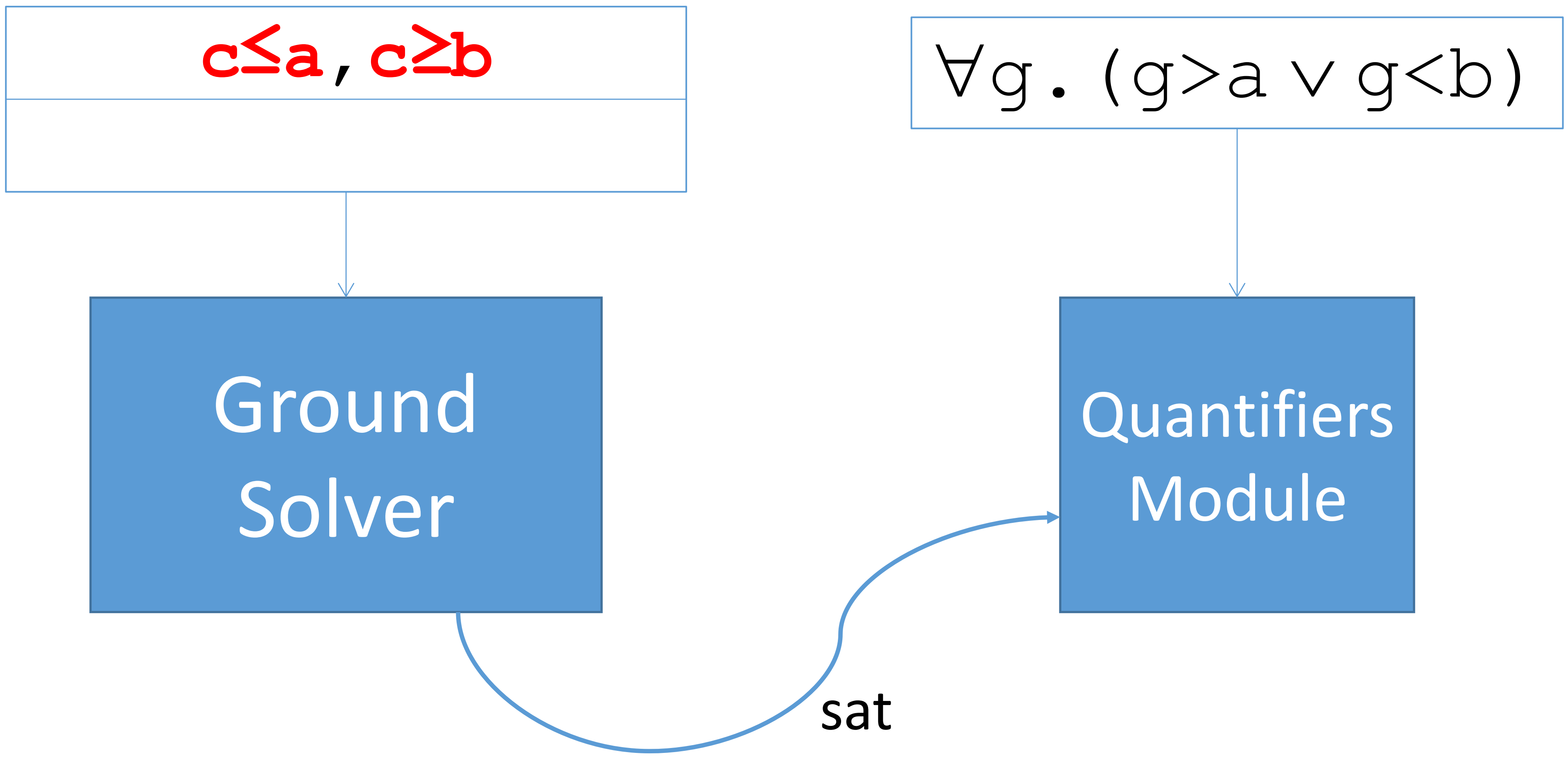
⇒ Desired function, after simplification

Counterexample-Guided Instantiation

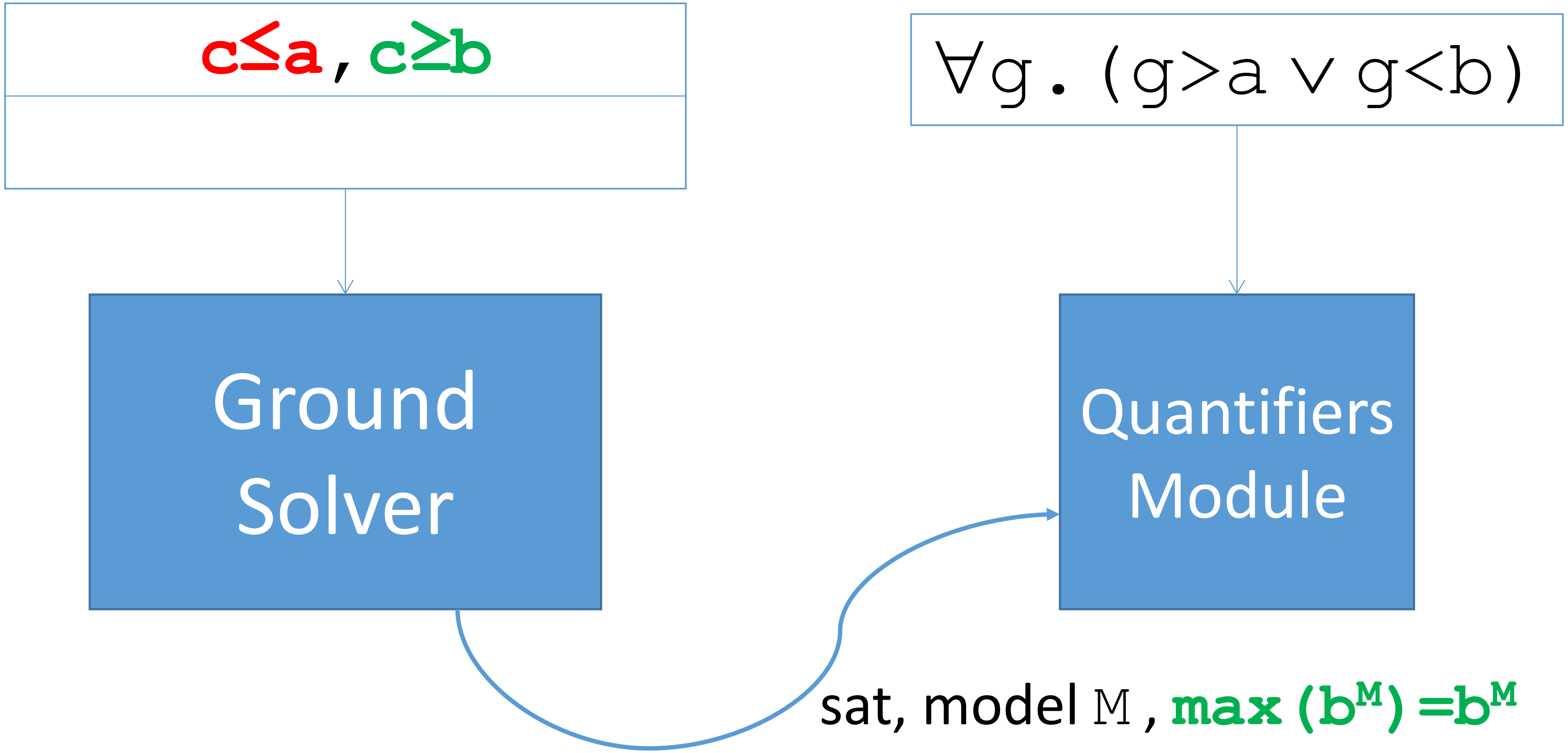


- Consider example: $\forall g.\ (g>a \lor g<b)$

Counterexample-Guided Instantiation

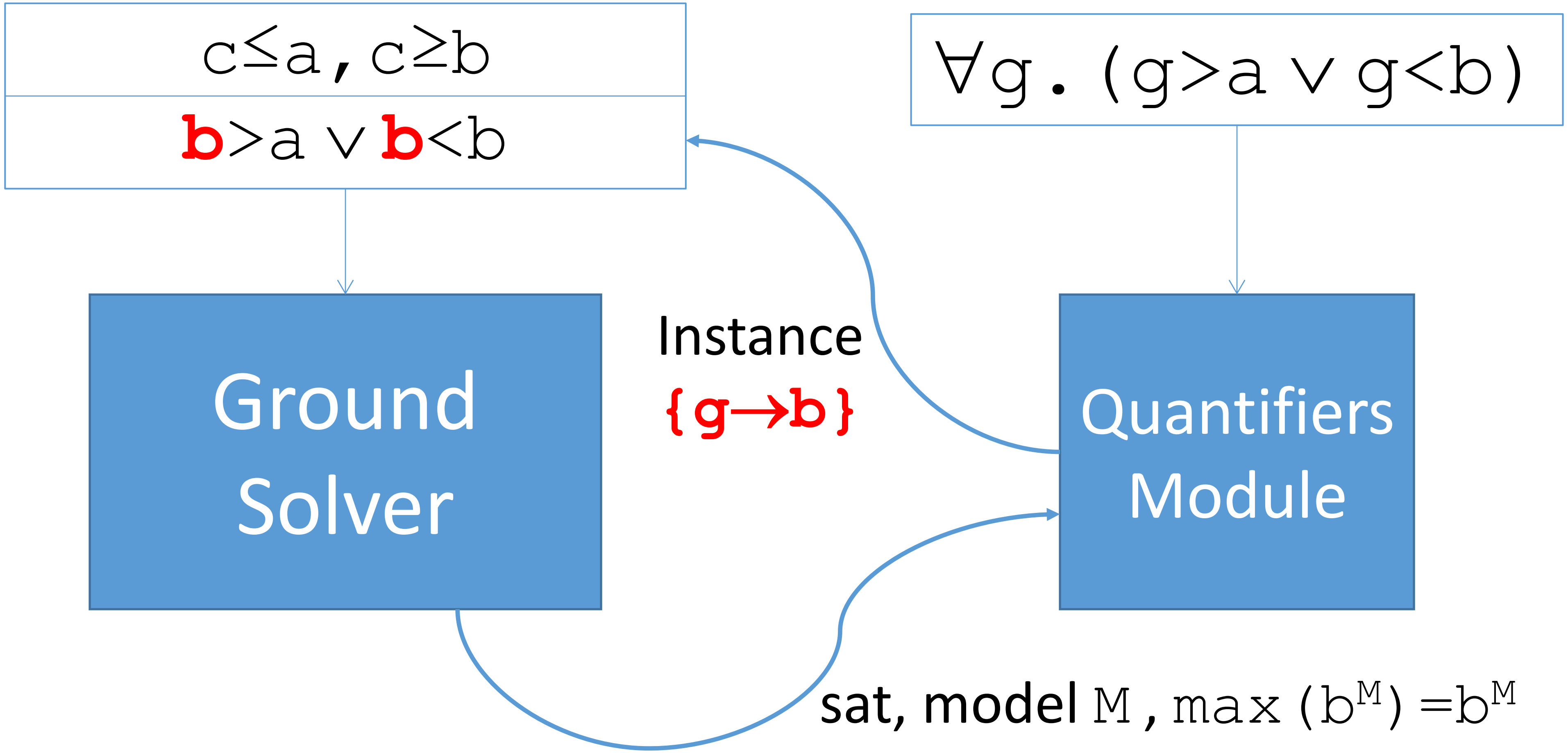


Counterexample-Guided Instantiation

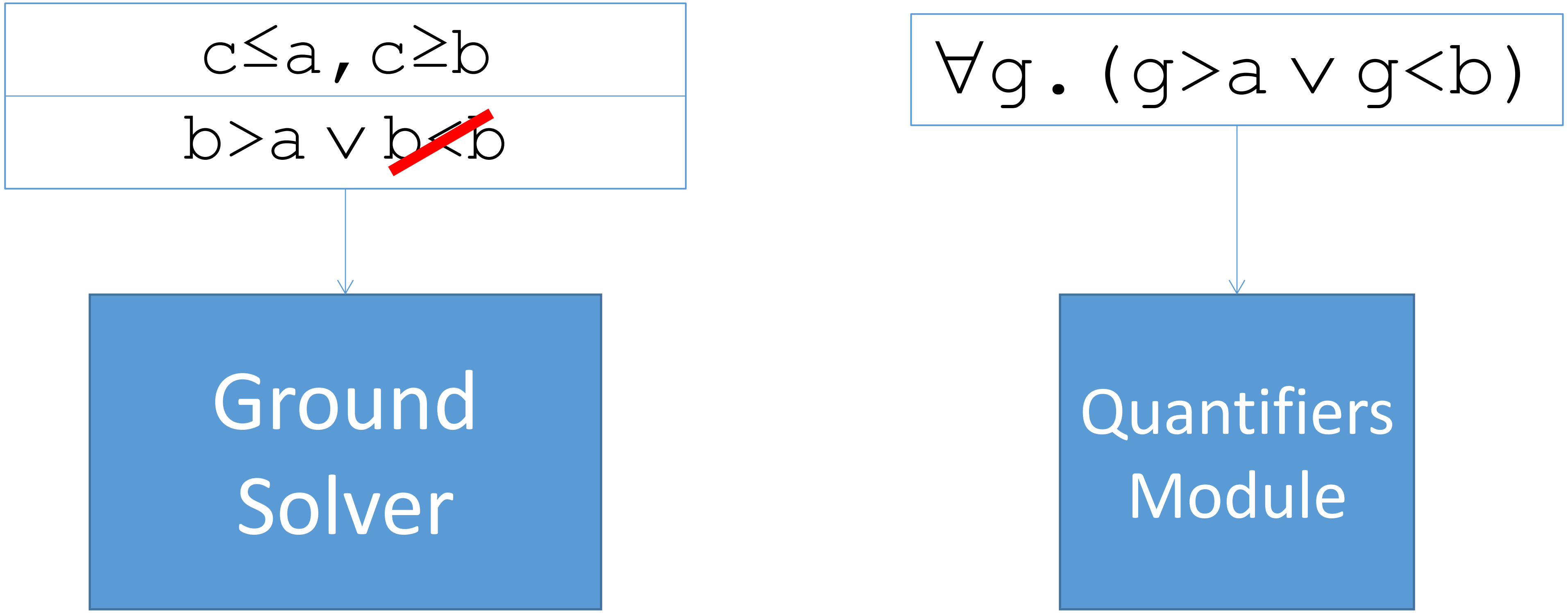


- Take maximal lower bound for c in model M

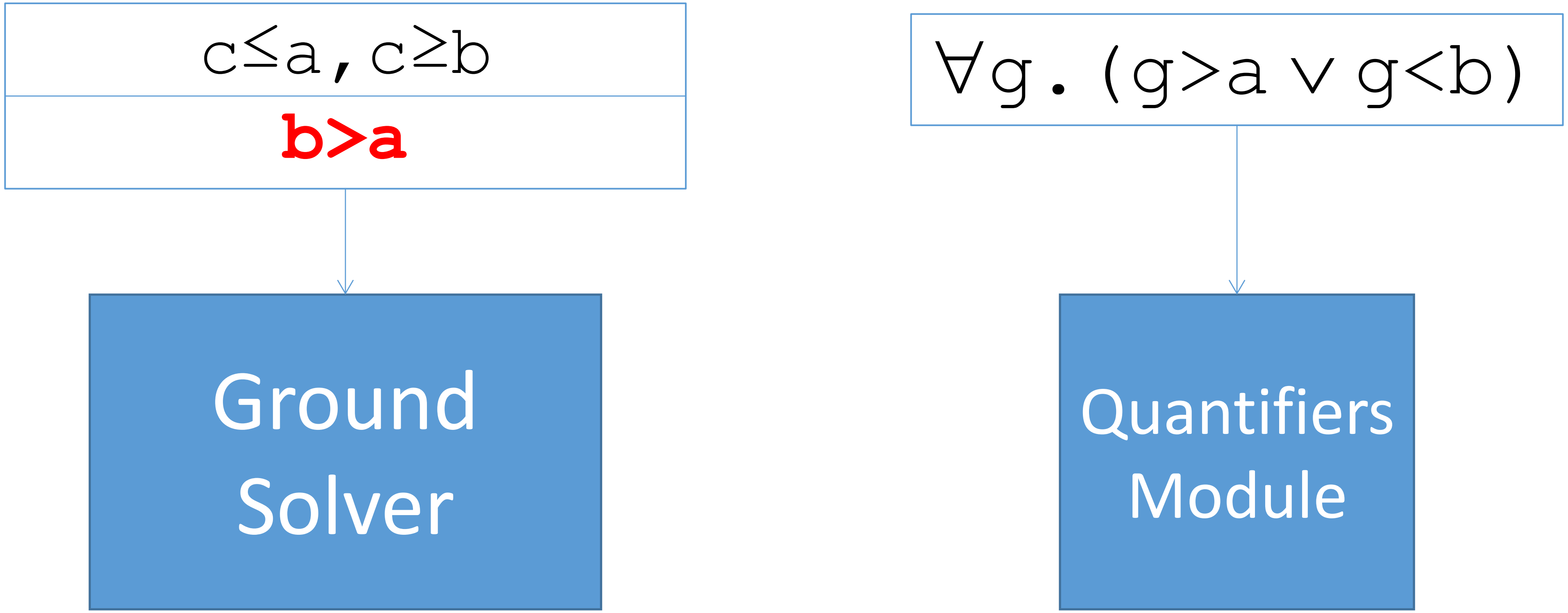
Counterexample-Guided Instantiation



Counterexample-Guided Instantiation

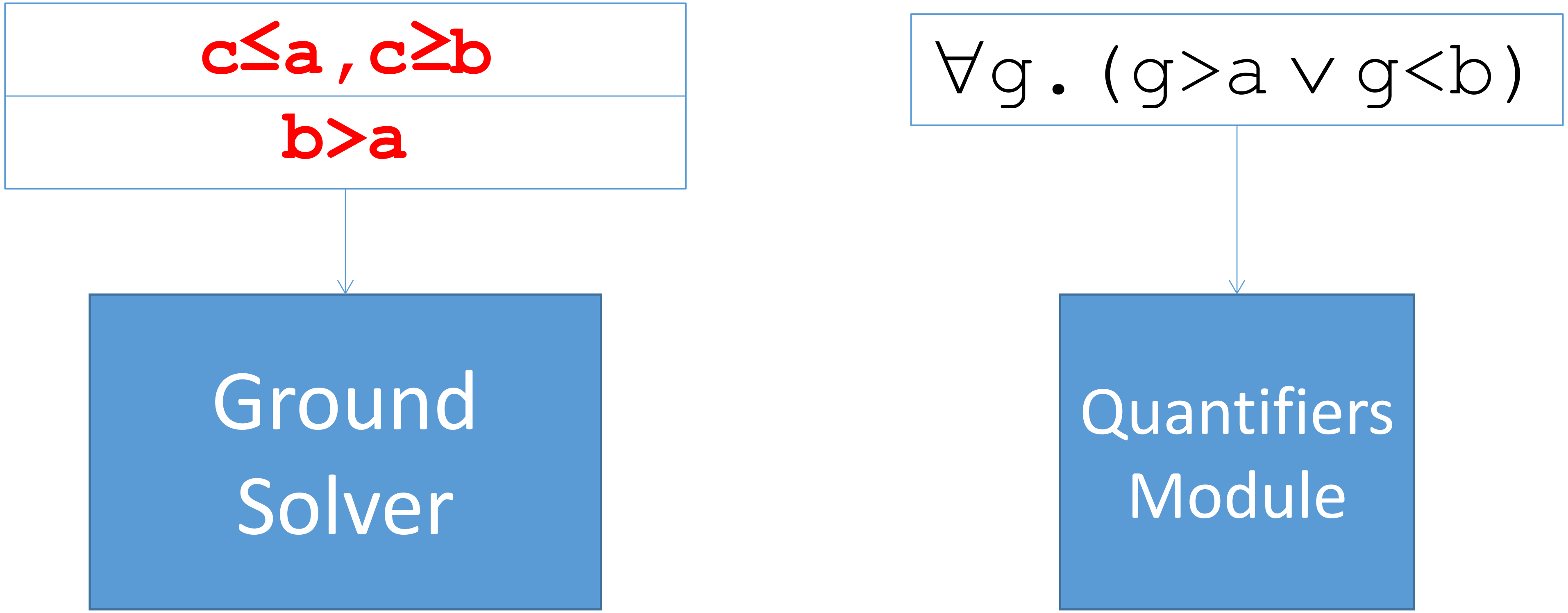


Counterexample-Guided Instantiation



- {b>a} is sat

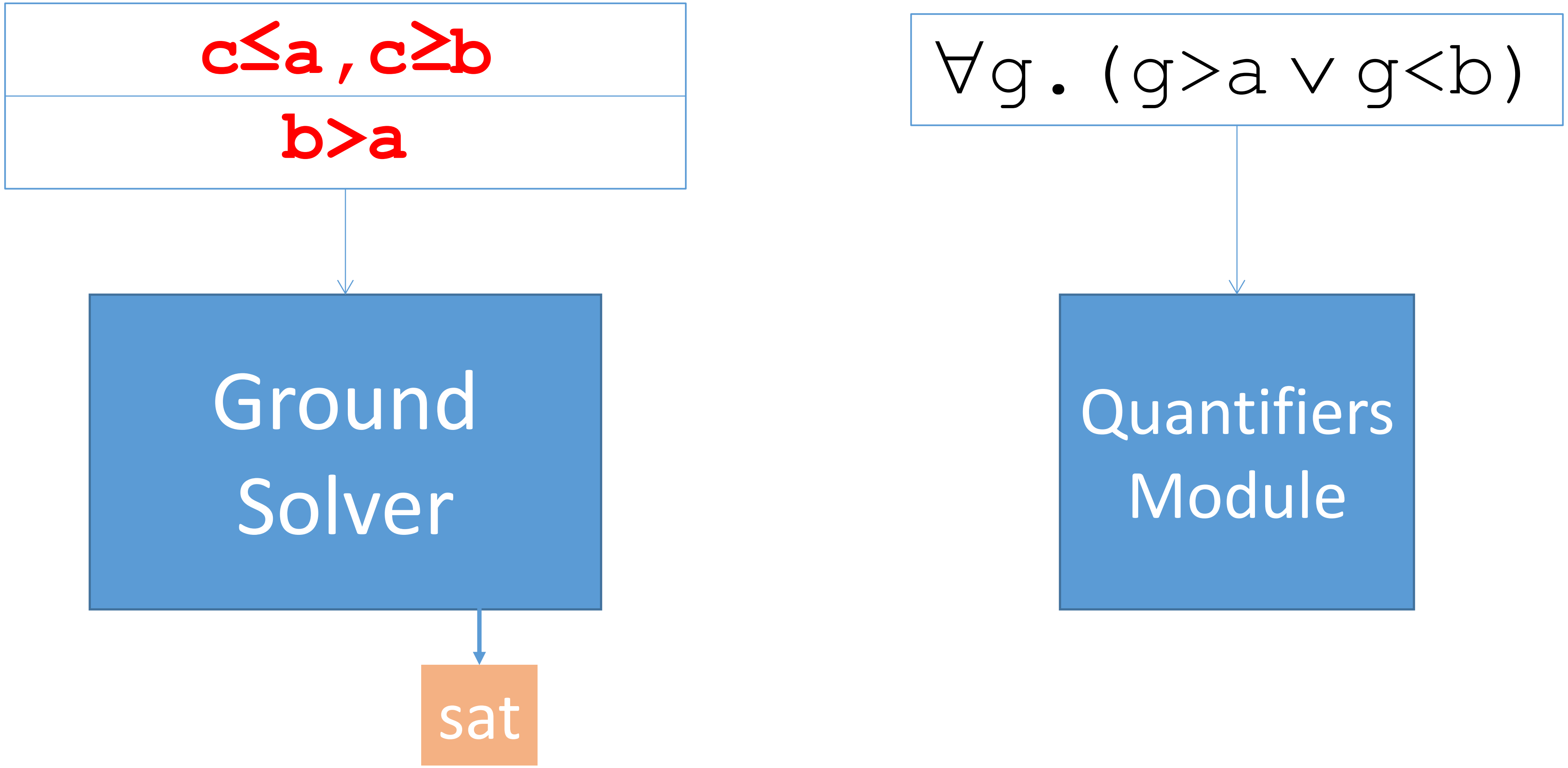
Counterexample-Guided Instantiation



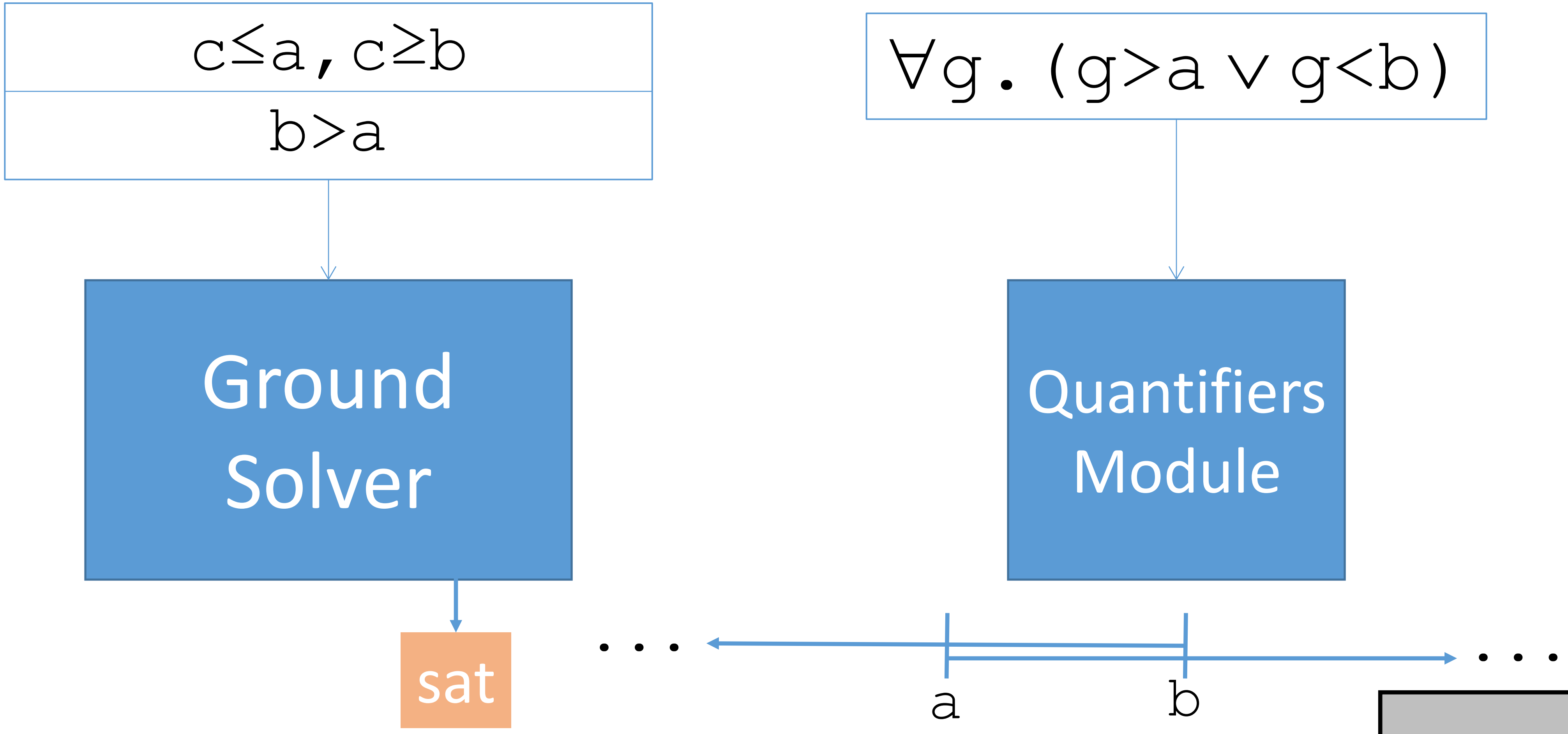
- $\{b>a\}$ is sat
- ...but $\{c\leq a, c\geq b, b>a\}$ is unsat

  $\Rightarrow$ 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)`

EXAMPLE...

Counterexample-Guided Instantiation

- For linear real and integer arithmetic:
  - With one quantifier alternation:
    - **Sound** and **complete** (terminating) **[Reynolds/King/Kuncak, draft 2015]**
  - With arbitrary quantifier alternations:
    - Effective in practice, for both “sat” and “unsat”

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
  - 2nd place, first-order theorems division of CASC 25
  - **Won**, first-order non-theorems division of CASC 25

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

Conclusion

- CVC4 + quantified formulas can be used for:
  - Theorem proving and verification
  - Finite model finding (`--finite-model-find`)
  - Function synthesis (`--cegqi`, on *.sl)
  - …and more:
    - Inductive Theorem Proving (`--quant-ind`)
    - Model finding for recursive functions (`--fmf-fun`)
    - …

$\Rightarrow$ *All techniques work in combination with the wide array of ground theories CVC4 supports*

Thanks!

- CVC4 is publicly available at:

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