Synthesis by Quantifier Instantiation in CVC4

Andrew Reynolds

May 4, 2015

Overview

- SMT solvers : how they work
- Synthesis Problem : $\exists$ f. $\forall$ x. P(f, x)

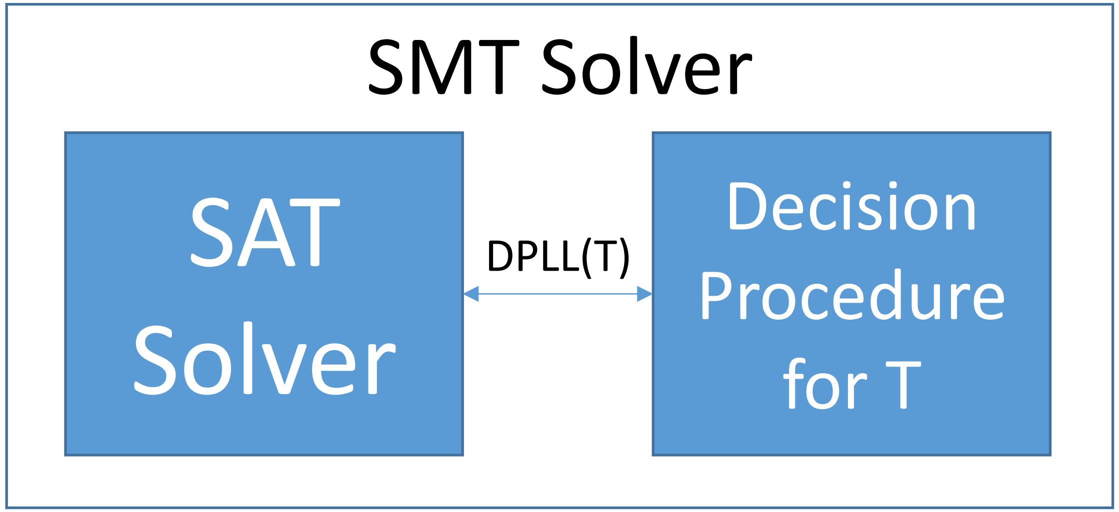
  There exists a function f such that for all x, P(f, x)

- New approaches for synthesis problems in an SMT solver [CAV 15]
  - Implemented in the SMT solver CVC4
- Evaluation

SMT solvers

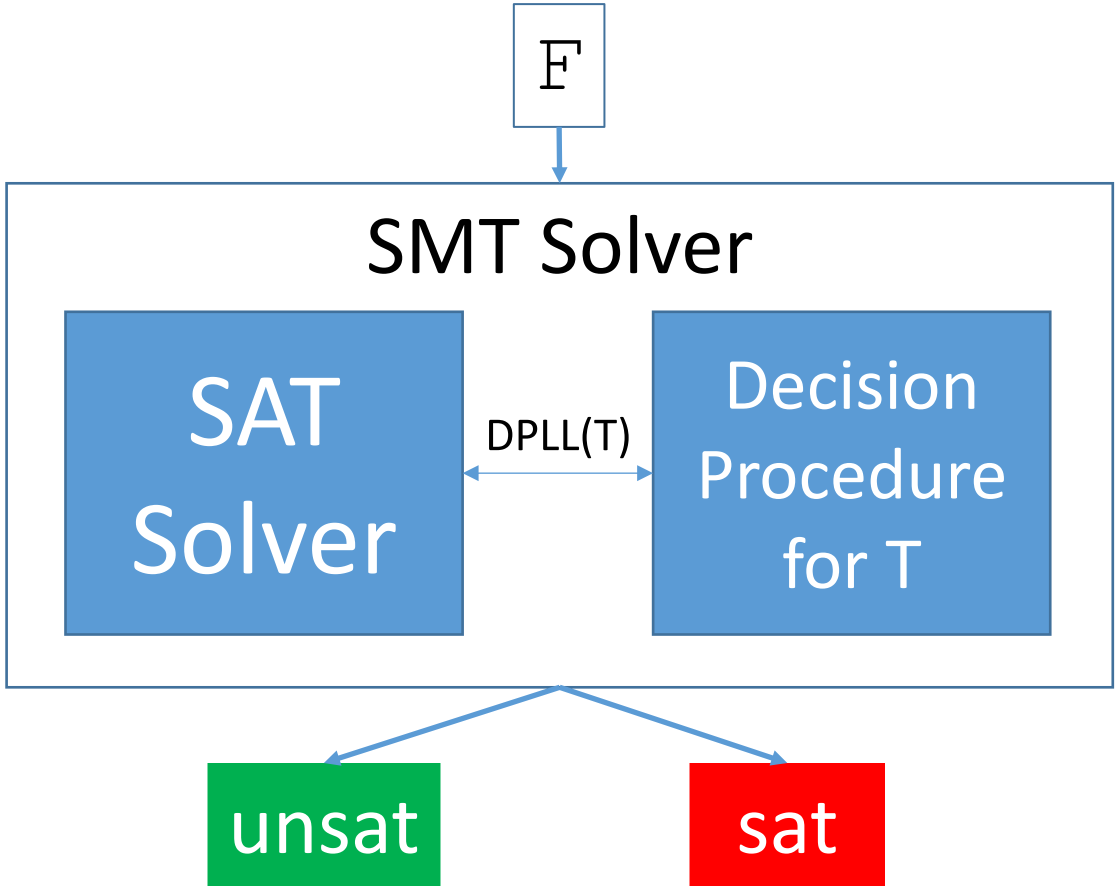
- Are powerful tools used in many formal methods applications:
  - Software and Hardware verification
  - Automated Theorem Proving
  - Scheduling and Planning
  - Software synthesis
- Reason about Boolean combinations of *theory* constraints:
  - Linear arithmetic : `2*a+1>0`
  - Bitvectors : `bvsgt(a,#bin0001)`
  - Arrays : `select(store(a,5,b),c)=5`
  - Datatypes : `tail(cons(a,b))=b`
  - ….

SMT Solver for Theory T



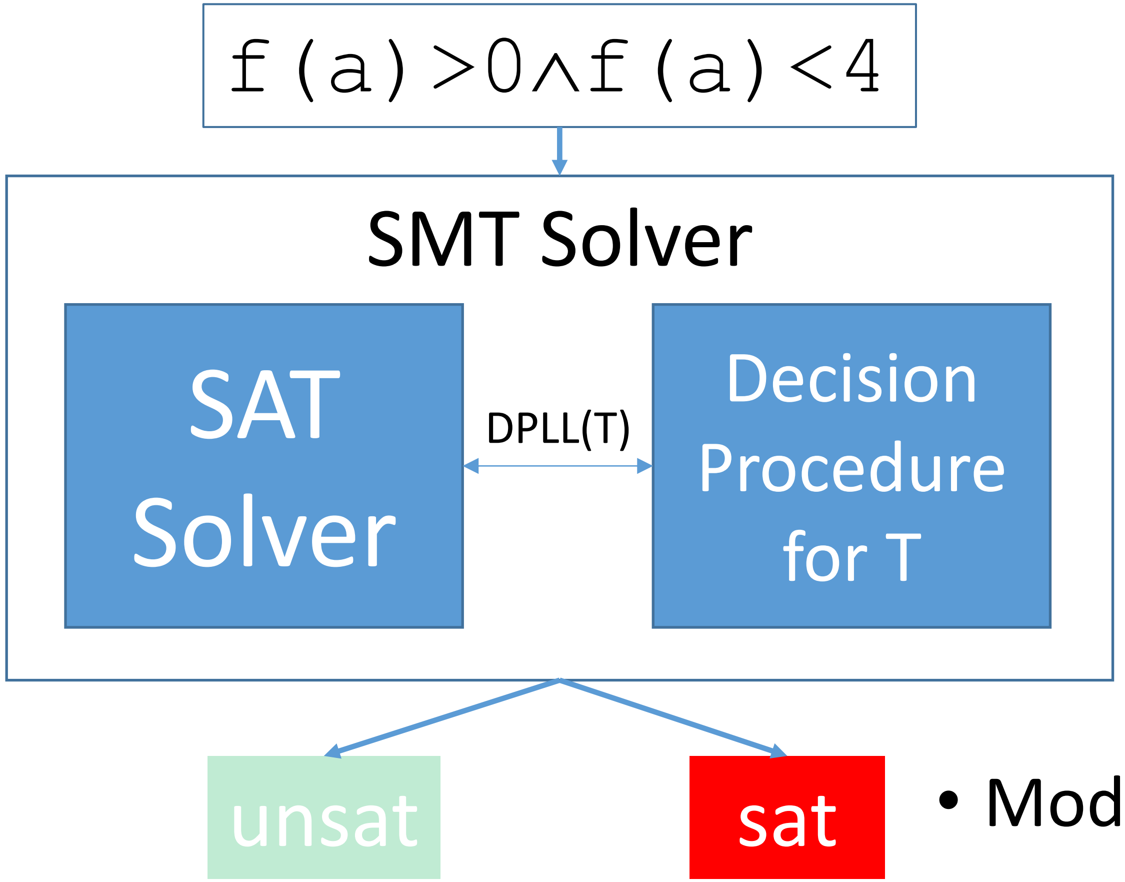
- Combines:
  - Off the shelf SAT solver
  - (Possibly combined) decision procedure for decidable theory T
- Components communicate via DPLL(T) framework

SMT Solver for Theory T



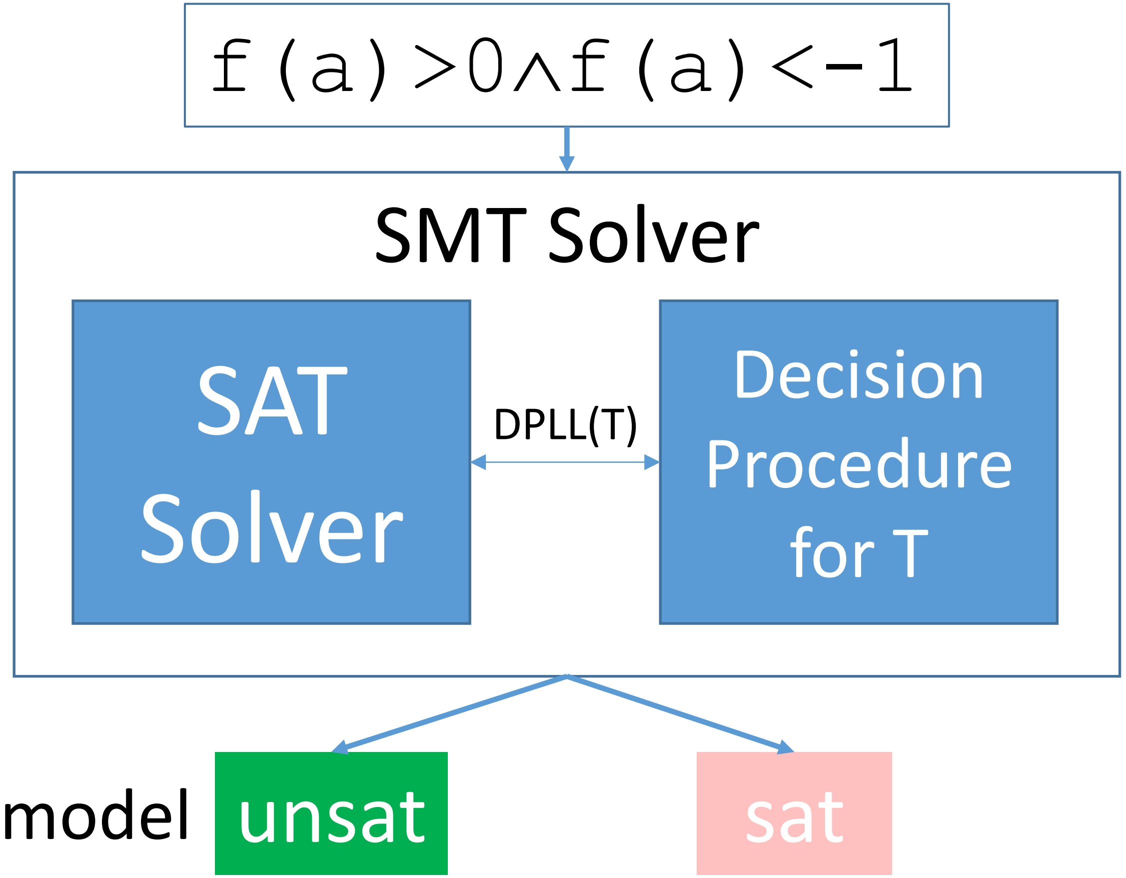
- Determines if set of formulas F is *T-satisfiable*

SMT Solver for Theory T



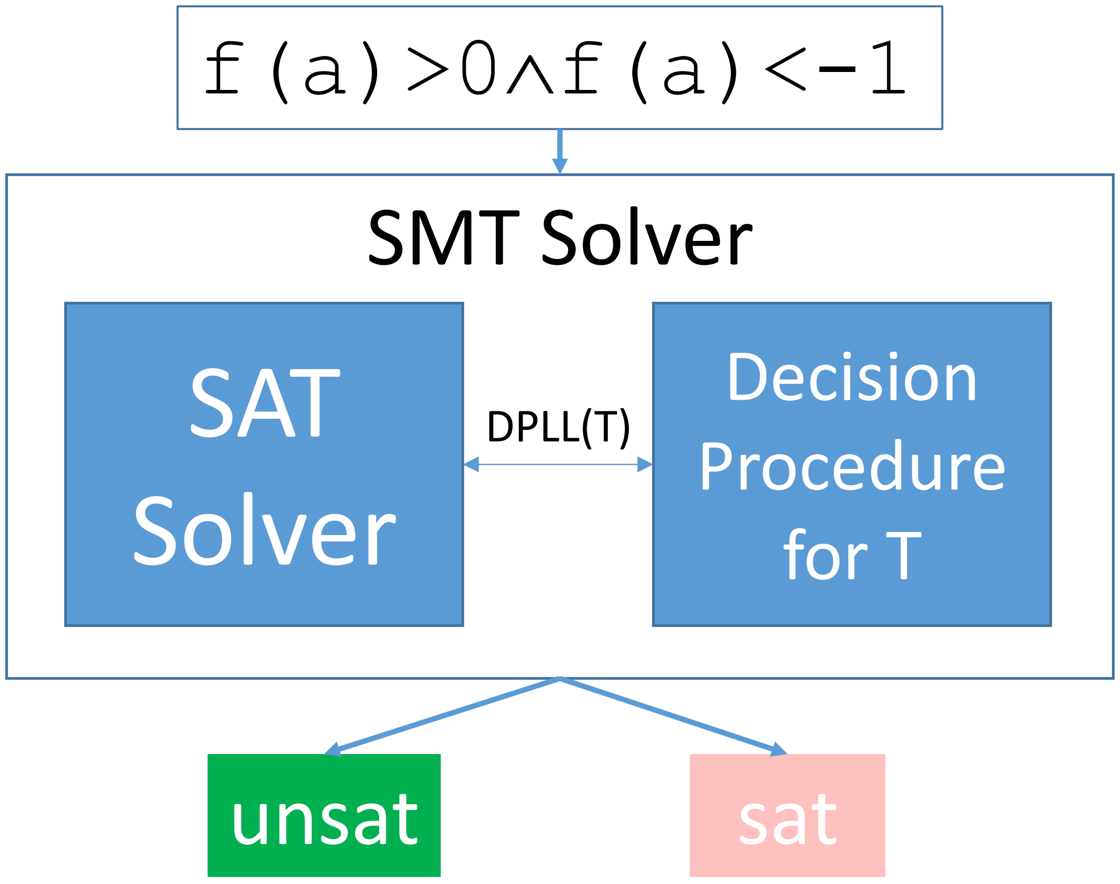
- Model, for example `f(a)=1`

SMT Solver for Theory T



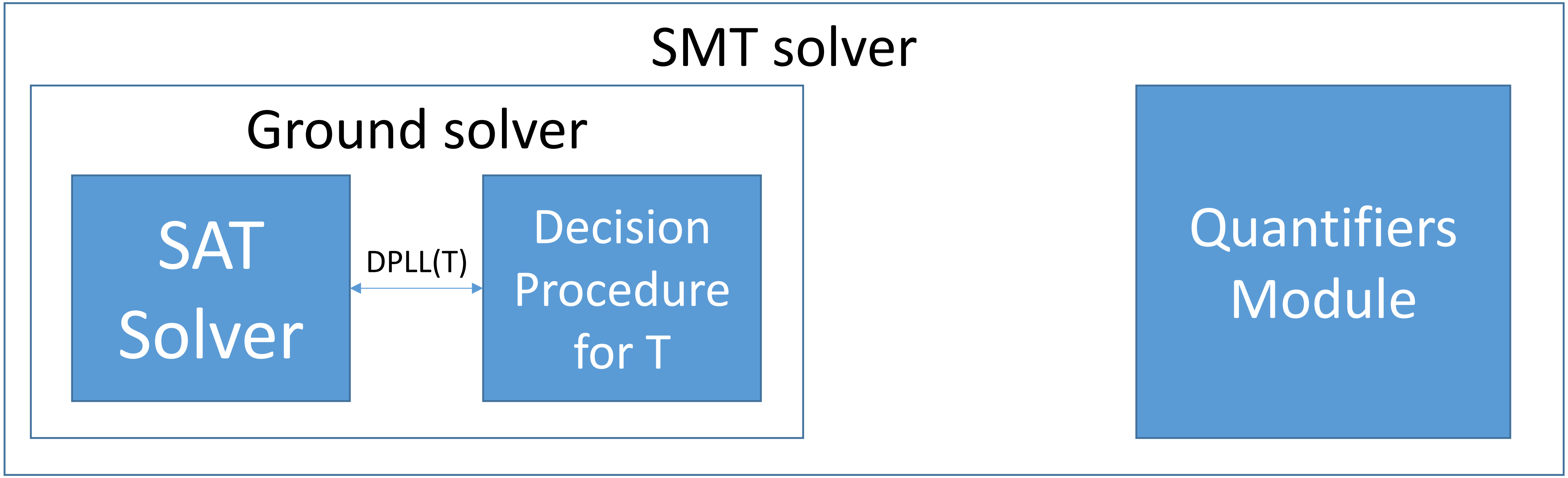
- No model

SMT Solver for Theory T



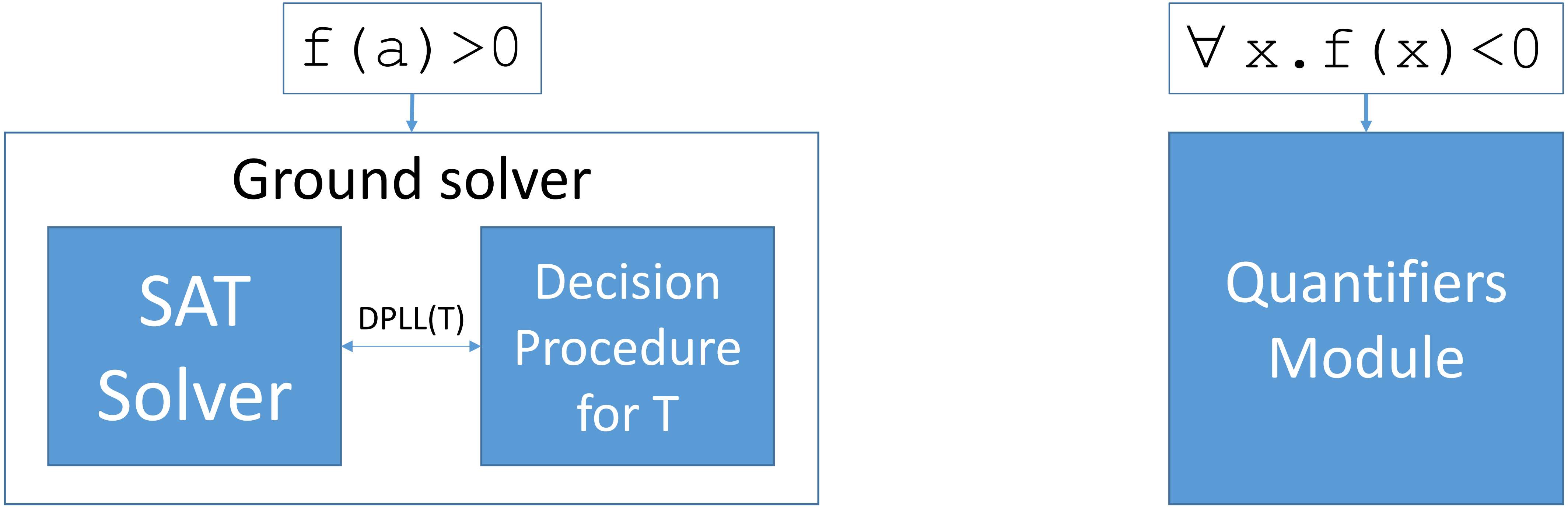
- For decidable theories (e.g. here T is $T_{UF}+T_{LIA}$)
  - Solver is terminating with either "unsat" or "sat"

SMT Solver + Quantified Formulas



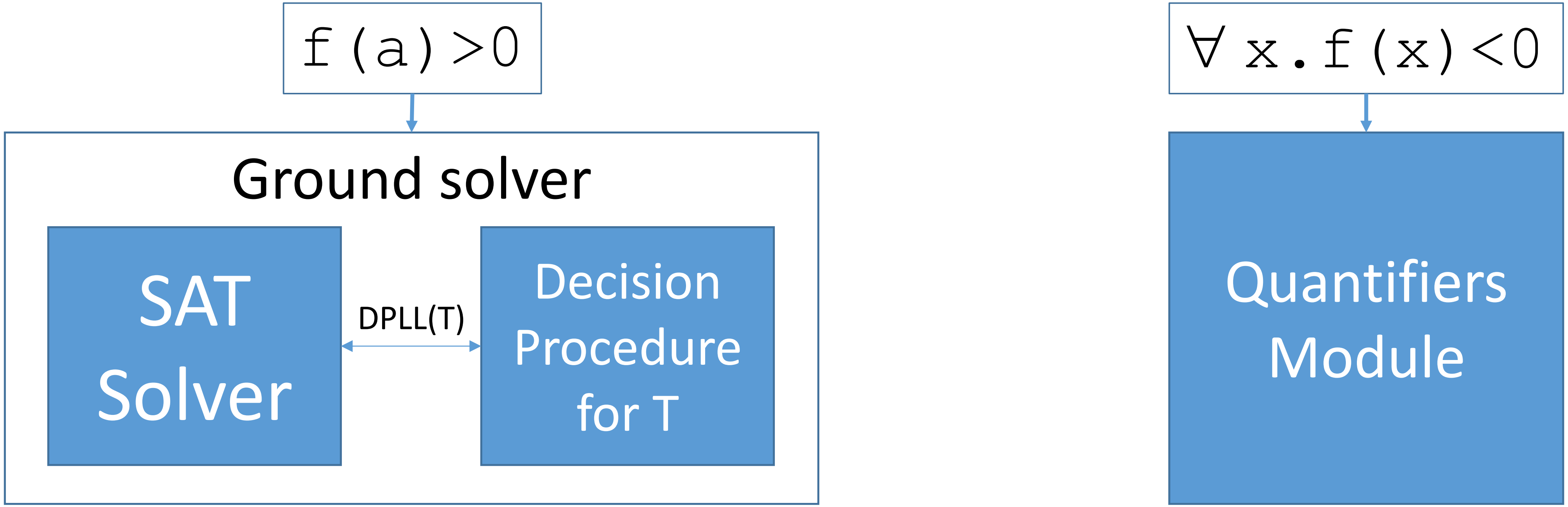
- SMT solvers have limited support for (first-order) quantified formulas ∀

SMT Solver + Quantified Formulas

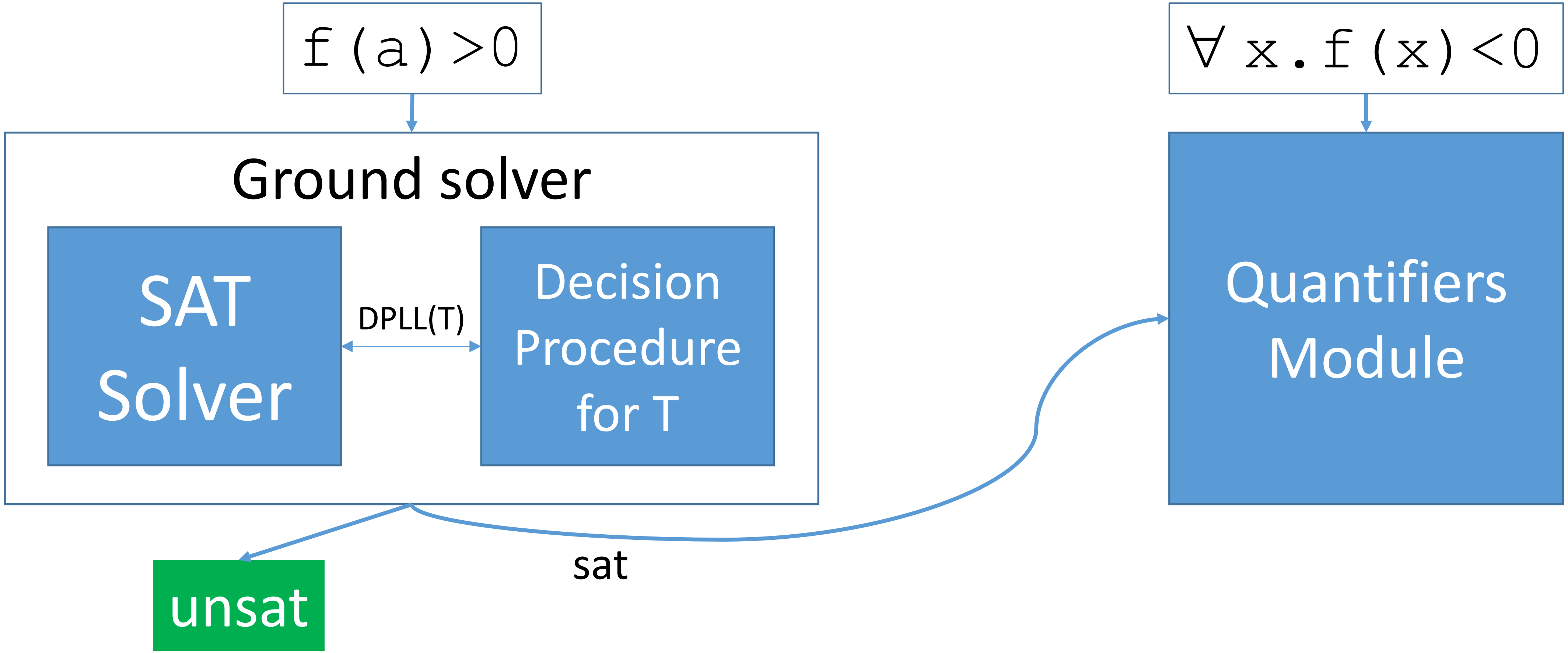


- For input `f(a)>0` ∧ `∀x.f(x)<0`
  - Ground solver maintains a set of ground (variable-free) constraints : `f(a)>0`
  - Quantifiers Module maintains a set of axioms : `∀x.f(x)<0`

SMT Solver + Quantified Formulas

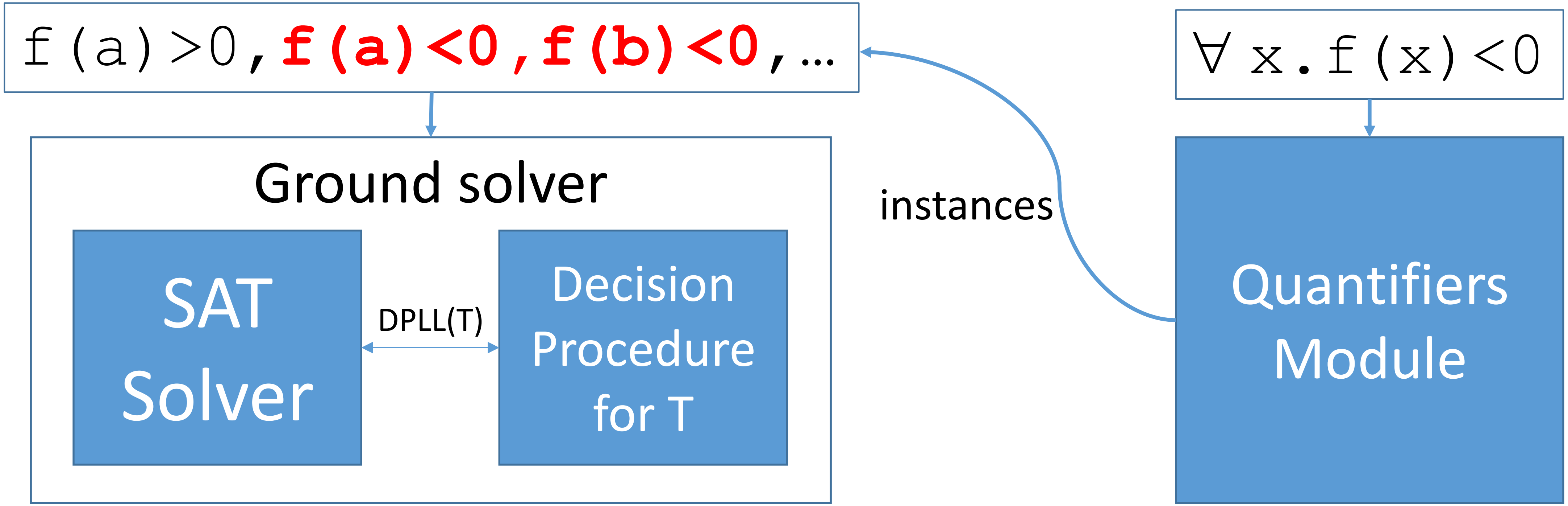


SMT Solver + Quantified Formulas



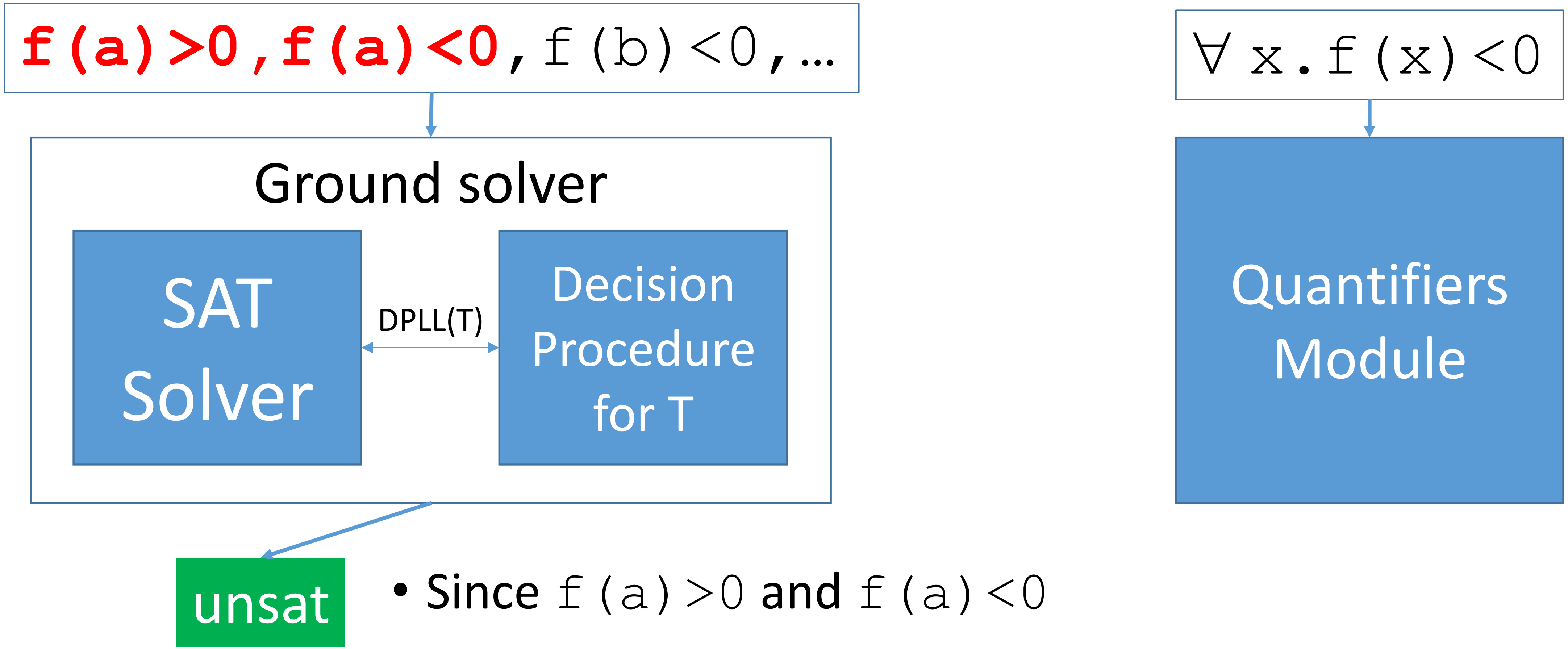
- Ground solver checks T-satisfiability of current set of constraints

SMT Solver + Quantified Formulas



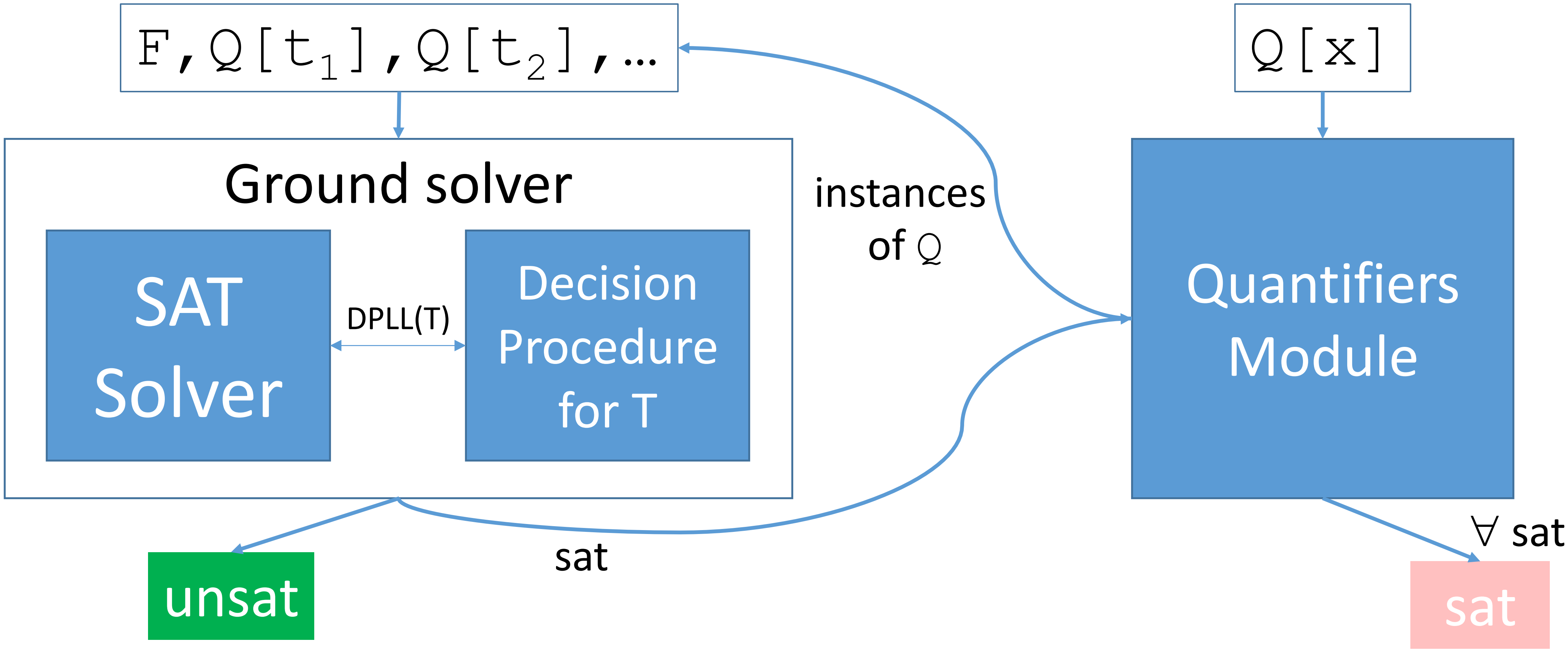
- Quantifiers Module adds instances of axioms
  - Goal : add instances until ground solver can answer "unsat"

SMT Solver + Quantified Formulas



- Since `f(a)>0` and `f(a)<0`

SMT Solver + Quantified Formulas



- Generally, a sound but incomplete procedure
  - Difficult to answer sat (when have we added enough instances of Q[x]?)

Approaches for Quantifiers in SMT

- Heuristic instantiation (good for “unsat”):
  - E-matching [Detlefs et al 2003, Ge et al 2007, de Moura/Bjorner 2007]
- Complete approaches (may answer “sat”):
  - Local theory extensions [Sofronie-Stokkermans 2005]
  - 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

The Synthesis problem

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

There exists a function $\mathtt{f}$ such that for all $\mathbf{x}$, property $\mathtt{P}$ holds

- Most existing approaches for synthesis
  - E.g. [Solar-Lezama et al 2006, Udupa et al 2013, Milicevic et al 2014]
  - Rely on specialized solver that makes subcalls to an SMT Solver
- Approach for synthesis in this talk:
  - *Instrument an approach for synthesis entirely inside SMT solver*

Running 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:

$$f := \lambda xy.ite(x>y,x,y)$$

How does an SMT solver handle Max example?

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

How does an SMT solver handle Max example?

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

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

- Straightforward approach:
  - Treat `f` as an *uninterpreted function*
  - Succeed if SMT solver can find correct interpretation of `f`, answer "sat"

$\Rightarrow$ *However, this is challenging*

  - SMT solvers have limited ability to find models when $\forall$ are present
  - It is difficult to directly synthesize interpretation `λxy.ite(x>y,x,y)`

Refutation-Based Synthesis

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

- Since SMT solvers are limited at answering "sat" when $\forall$ are present,
  - $\Rightarrow$ Can we instead use a *refutation-based* approach for synthesis?

What if we negate the synthesis conjecture?

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

- Negate the synthesis conjecture
- If we are in a *satisfaction-complete* theory T (e.g. linear arithmetic, bitvectors):
  - *F is T-satisfiable if and only if $\neg F$ is T-unsatisfiable*
  - In such cases:
    - If SMT solver can establish $\neg \exists \texttt{f}. \forall \mathbf{x}. \texttt{P}(\texttt{f}, \mathbf{x})$ is *unsatisfiable*
    - Then we know that $\exists \texttt{f}. \forall \mathbf{x}. \texttt{P}(\texttt{f}, \mathbf{x})$ is satisfiable ($\texttt{f}$ has a solution)

Challenge: Second-Order Quantification

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

negate

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

- Want to show negated formula is unsatisfiable
- Challenge: outermost quantification $\forall f$ 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

Challenge: Second-Order Quantification

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

negate

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

- Want to show negated formula is unsatisfiable
- Challenge: outermost quantification $\forall f$ 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 $\Leftarrow$ *Focus of this talk*
  2. When f is given syntactic restrictions

Single Invocation Property : Max Example

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

Single Invocation Property : Max Example

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

- *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)

- Examples of properties that are not single invocation:
  - $\forall c. \exists xy. c(x,y) = c(y,x)$, e.g. c is commutative

Single Invocation Property : Max Example

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

Push quantification downwards

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

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

Single Invocation Property : Max Example

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

Push quantification downwards

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

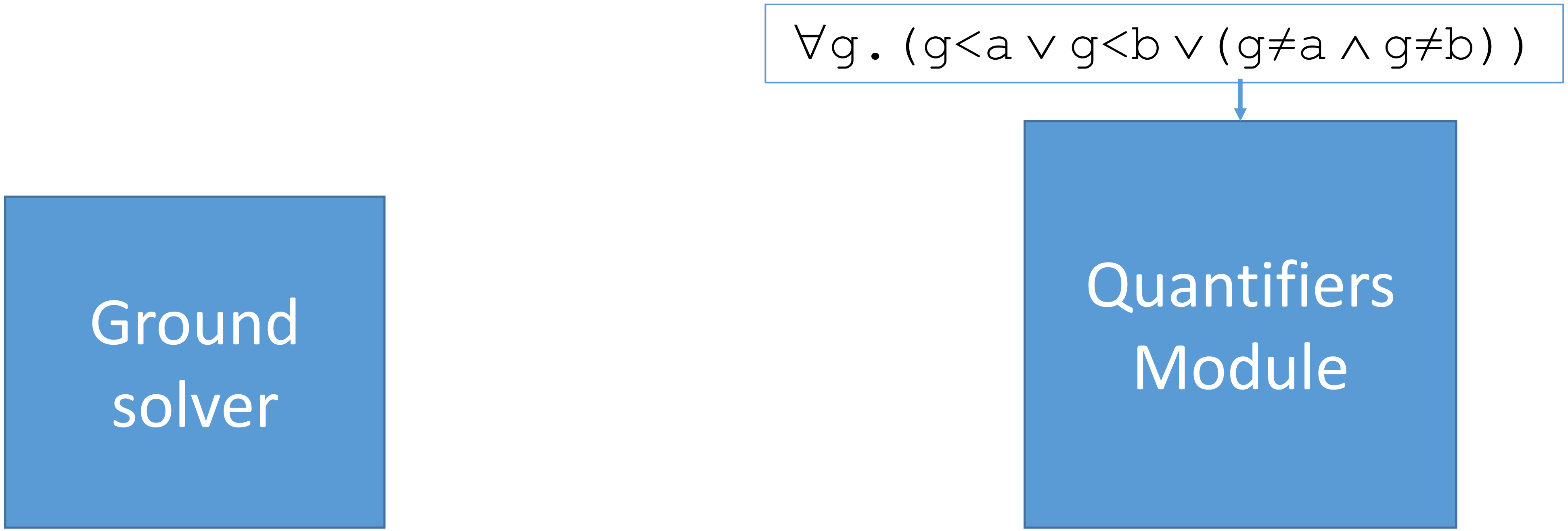
Skolemize, for fresh a and b

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

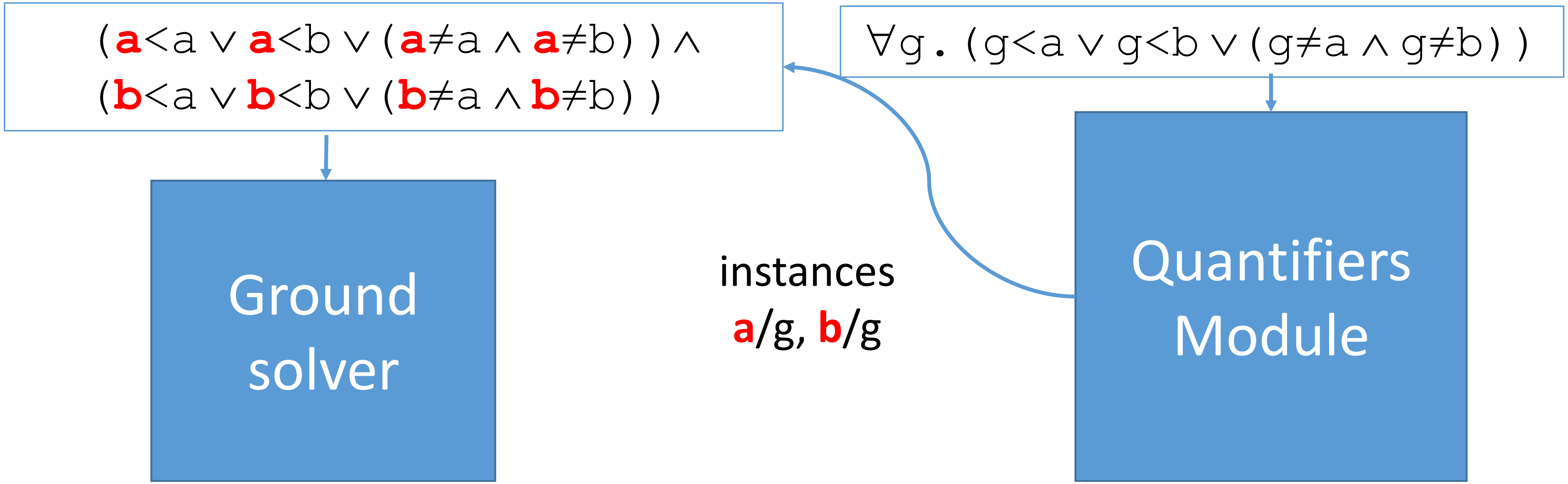
Solving Max Example

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

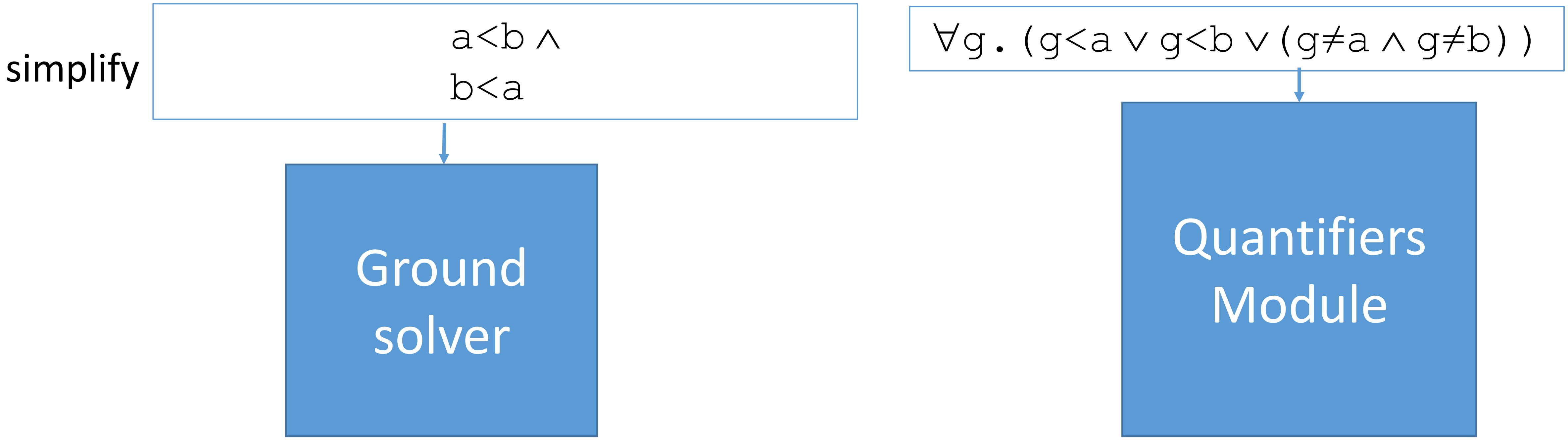
Solving Max Example



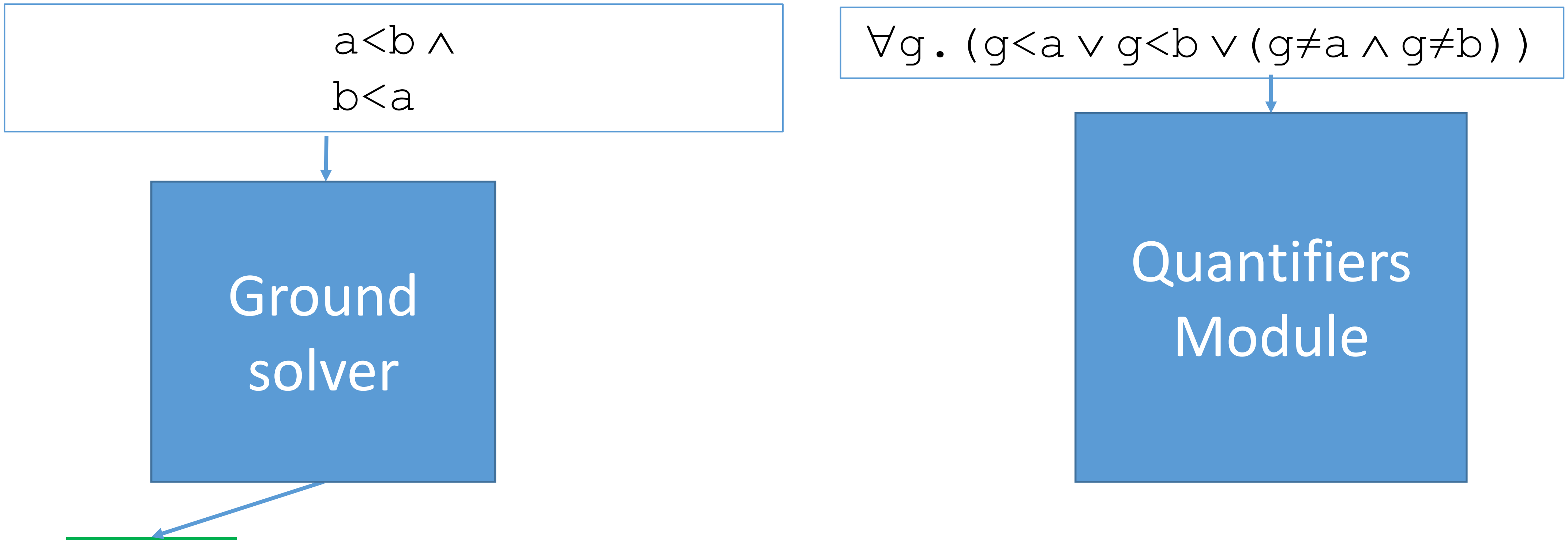
Solving Max Example



Solving Max Example



Solving Max Example



unsat ⇒ $\forall g.\,(g<a \lor g<b \lor (g\neq a \land g\neq b))$ is unsatisfiable,
implies original synthesis conjecture has a solution

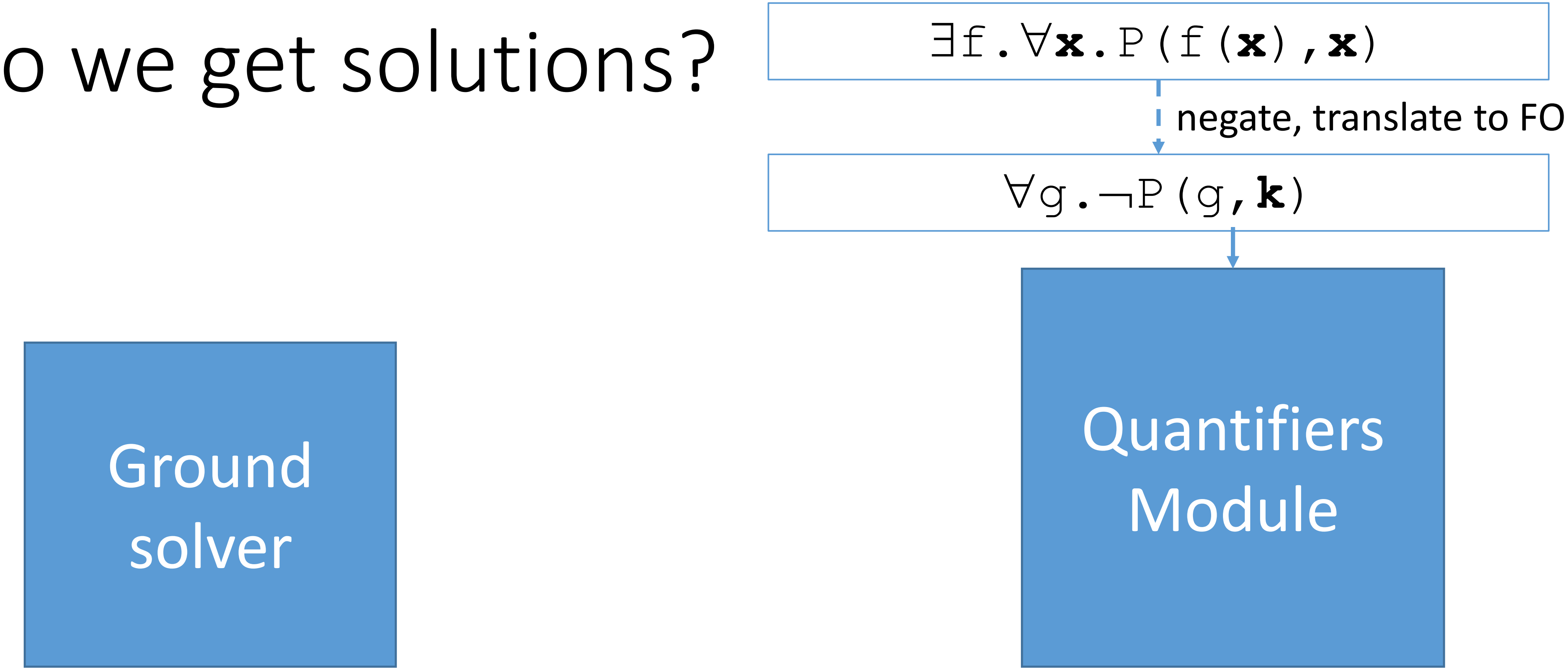
How do we get solutions?

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

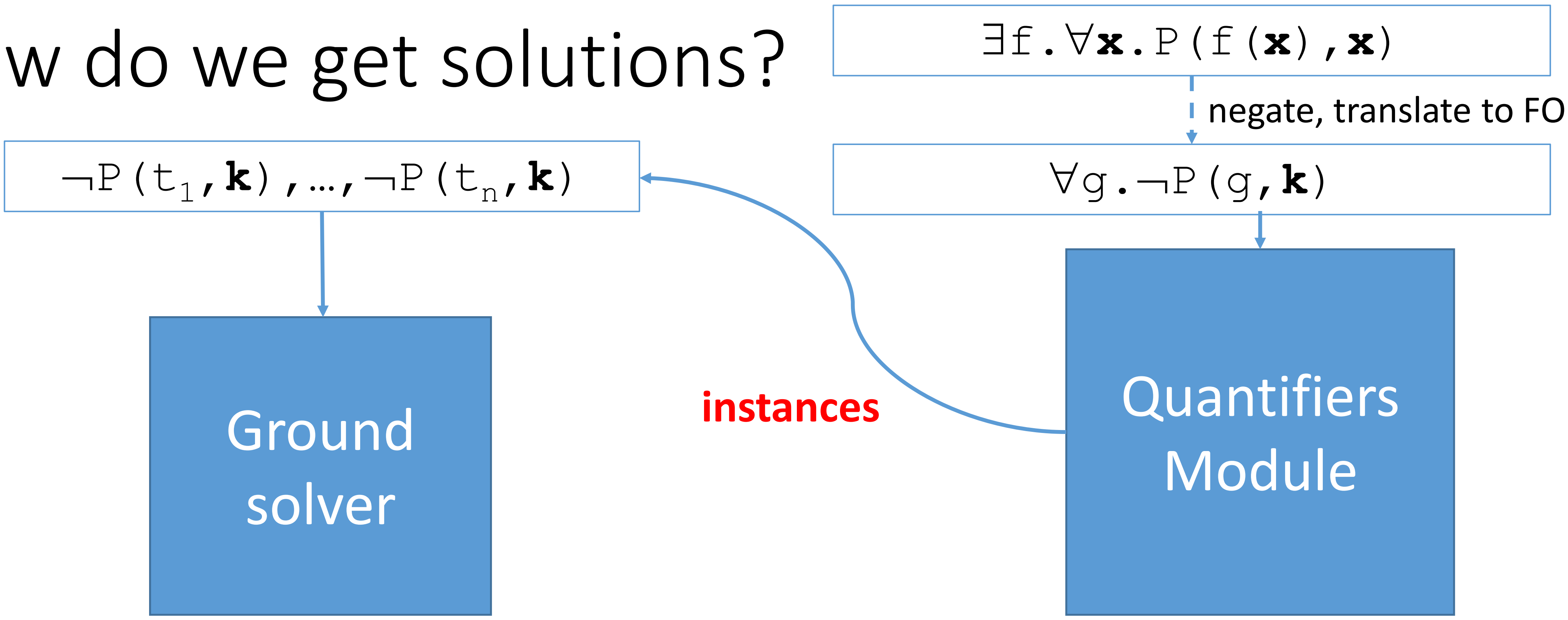
- Given refutation-based approach for synthesis conjecture $\exists f. \forall \mathbf{x}. P(f(\mathbf{x}), \mathbf{x})$

  $\Rightarrow$ Solution for f can be extracted from unsatisfiable core of instantiations

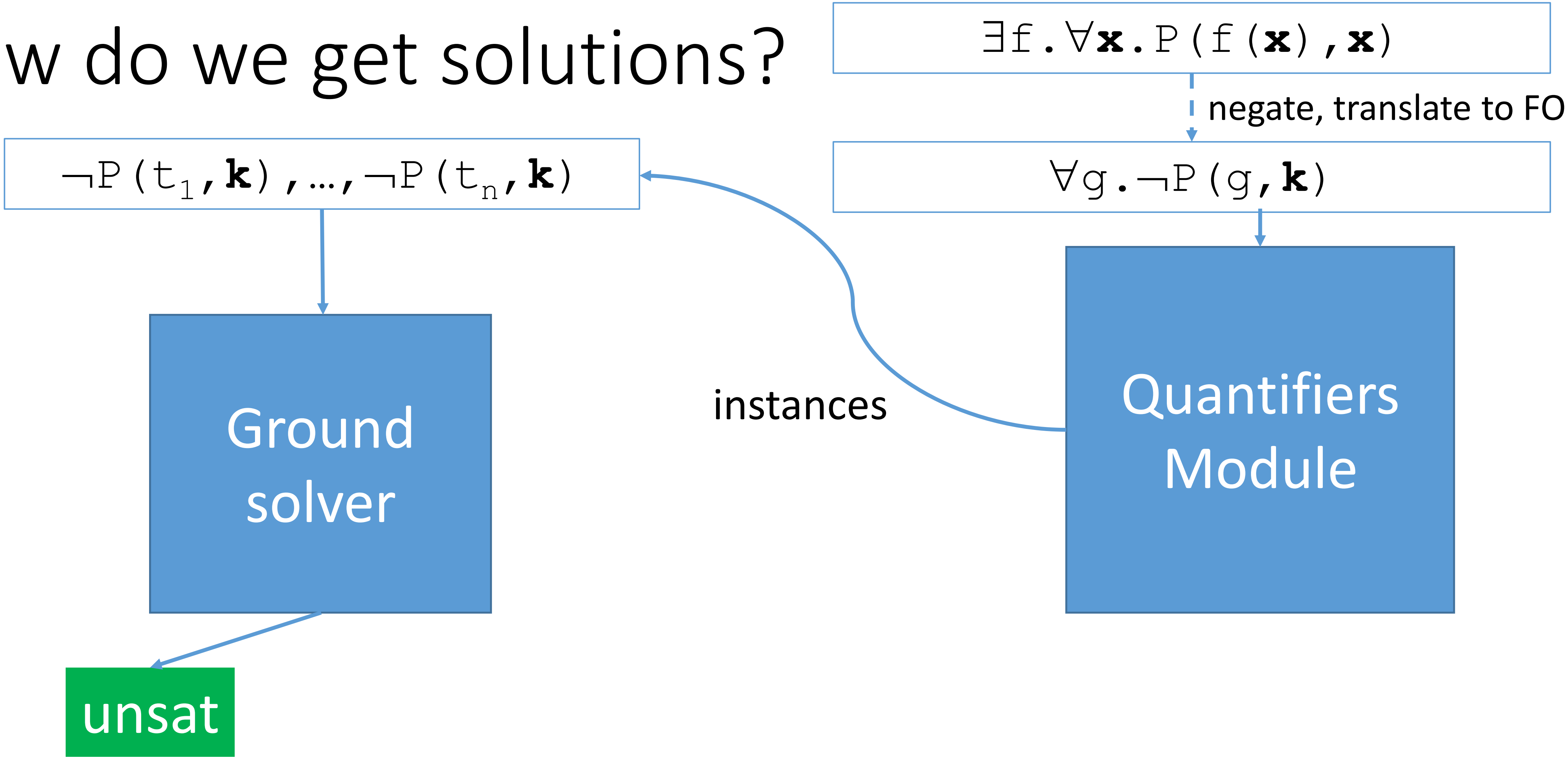
How do we get solutions?



How do we get solutions?

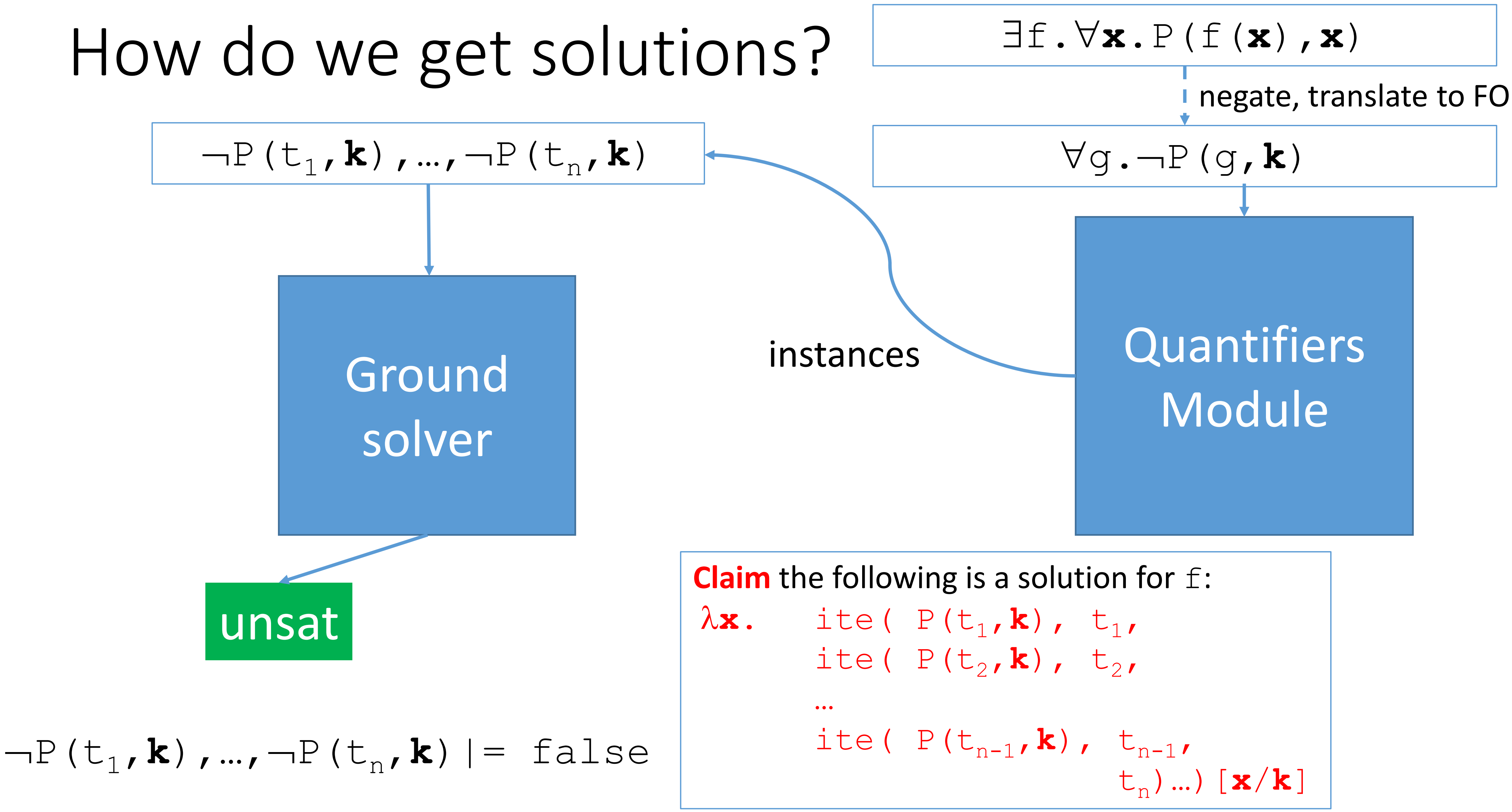


How do we get solutions?



$\neg P(t_1, \mathbf{k}), \ldots, \neg P(t_n, \mathbf{k}) \models \text{false}$

How do we get solutions?



Why is this a solution?

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

Found $\neg \mathtt{P}(\mathtt{t}_1,\mathbf{k}),\ldots,\neg \mathtt{P}(\mathtt{t}_n,\mathbf{k}) \models \mathtt{false}$

Claim the following is a solution for f:

$$
\begin{aligned}
\lambda \mathbf{x}.\ & \mathtt{ite}(\ \mathtt{P}(\mathtt{t}_1,\mathbf{k}),\ \mathtt{t}_1, \\
& \mathtt{ite}(\ \mathtt{P}(\mathtt{t}_2,\mathbf{k}),\ \mathtt{t}_2, \\
& \ldots \\
& \mathtt{ite}(\ \mathtt{P}(\mathtt{t}_{n-1},\mathbf{k}),\ \mathtt{t}_{n-1}, \\
& \qquad\qquad\qquad \mathtt{t}_n)\ldots)[\mathbf{x}/\mathbf{k}]
\end{aligned}
$$

Why is this a solution?

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

Found $\neg P(t_1, \mathbf{k}), \ldots, \neg P(t_n, \mathbf{k}) \models \text{false}$

Claim the following is a solution for f:

$\lambda \mathbf{x}.\ \text{ite}(\ P(t_1,\mathbf{k}),\ t_1,$
$\quad \text{ite}(\ P(t_2,\mathbf{k}),\ t_2,$
$\quad \ldots$
$\quad \text{ite}(\ P(t_{n-1},\mathbf{k}),\ t_{n-1},$
$\quad\quad t_n)\ldots)\,[\mathbf{x}/\mathbf{k}]$

If P holds for t_1, return t_1

Why is this a solution?

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

Found $\neg P(t_1, \mathbf{k}), \ldots, \neg P(t_n, \mathbf{k}) \models \text{false}$

Claim the following is a solution for `f`:

$$\lambda \mathbf{x}.\ ite(\ P(t_1,\mathbf{k}),\ t_1,$$
$$ite(\ P(t_2,\mathbf{k}),\ t_2,$$
$$\ldots$$
$$ite(\ P(t_{n-1},\mathbf{k}),\ t_{n-1},$$
$$t_n)\ldots)\ [\mathbf{x}/\mathbf{k}]$$

If `P` holds for t_2, return t_2

Why is this a solution?

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

Found $\neg P(t_1,\mathbf{k}),\ldots,\neg P(t_n,\mathbf{k}) \models \text{false}$

Claim the following is a solution for f:

$$\lambda \mathbf{x}.\ \text{ite}(\ P(t_1,\mathbf{k}),\ t_1,$$
$$\text{ite}(\ P(t_2,\mathbf{k}),\ t_2,$$
$$\ldots$$
$$\text{ite}(\ P(t_{n-1},\mathbf{k}),\ t_{n-1},$$
$$t_n)\ldots)\ [\mathbf{x}/\mathbf{k}]$$

If P holds for t_{n-1}, return t_{n-1}

Why is this a solution?

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

Found $\neg P(t_1,\mathbf{k}),\ldots,\neg P(t_n,\mathbf{k}) \models \text{false}$

Claim the following is a solution for f:

$$
\begin{array}{l}
\lambda \mathbf{x}.\ \text{ite}(\ P(t_1,\mathbf{k}),\ t_1, \\
\quad \text{ite}(\ P(t_2,\mathbf{k}),\ t_2, \\
\quad \ldots \\
\quad \text{ite}(\ P(t_{n-1},\mathbf{k}),\ t_{n-1}, \\
\qquad\qquad t_n)\ldots)\,[\mathbf{x}/\mathbf{k}]
\end{array}
$$

Why does $P(t_n,\mathbf{k})$ hold?

Why is this a solution?

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

Found $\neg P(t_1, \mathbf{k}), \ldots, \neg P(t_{n-1}, \mathbf{k}) \models P(t_n, \mathbf{k})$

Claim the following is a solution for f:

$$
\begin{aligned}
\lambda \mathbf{x}.\ & ite(\ P(t_1, \mathbf{k}),\ t_1, \\
& ite(\ P(t_2, \mathbf{k}),\ t_2, \\
& \ldots \\
& ite(\ P(t_{n-1}, \mathbf{k}),\ t_{n-1}, \\
& \qquad\qquad t_n) \ldots) [\mathbf{x}/\mathbf{k}]
\end{aligned}
$$

Due to unsatisfiable core

Solution for Max Example

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

Solution for Max Example

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

Found $\neg(a \geq a \wedge a \geq b \wedge (a=a \vee a=b))$, $\neg(b \geq a \wedge b \geq b \wedge (b=a \vee b=b))$ $\models$ false

Solution for Max Example

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

Found $\neg(a \geq a \wedge a \geq b \wedge (a=a \vee a=b))$, $\neg(b \geq a \wedge b \geq b \wedge (b=a \vee b=b))$ $\models$ false

Claim the following is a solution for `f`:

$\lambda xy.$ ite($a \geq a \wedge a \geq b \wedge (a=a \vee a=b)$, a,
b)…) [x/a] [y/b]

Solution for Max Example

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

Found $\neg(a \geq a \wedge a \geq b \wedge (a=a \vee a=b))$, $\neg(b \geq a \wedge b \geq b \wedge (b=a \vee b=b))$ $\models$ false

Claim the following is a solution for f:

$\lambda xy.\ ite(\ x \geq x \wedge x \geq y \wedge (x=x \vee x=y),\ x,$
$y)\ldots)$

Solution for Max Example

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

Found $\neg(a \geq a \land a \geq b \land (a=a \lor a=b)),$
$\neg(b \geq a \land b \geq b \land (b=a \lor b=b))$ $\models$ false

Claim the following is a solution for f:

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

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

	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	3	467.6	2	71.6	50	888	0	0	5	1380.4	2	0.1	7	0.6	69	2808.3
cvc4	30	1448.6	5	0.1	52	2311.3	0	0	6	0.1	2	0.5	7	0.1	102	3760.7

- In total,
  - cvc4 finds solution for 35 that ESolver does not
  - ESolver finds solution for 2 that cvc4 does not
- Solves 25 benchmarks unsolved by any other known solver
  - Many of these in fraction of a second

Results : Max Example

	2	3	4	5	6	7	8	9	10
Esolver	0.01	1377.10	–	–	–	–	–	–	–
cvc4	0.01	0.02	0.03	0.05	0.1	0.3	1.6	8.9	81.5

- For class of properties synthesizing function taking max of n integers
  - cvc4 scales well to max9+
  - No solver from SyGuS competition synthesized max5 with timeout of an hour

Summary

- Refutation-based approach for synthesis
- Solutions constructed from unsatisfiable core of instantiations
- Implemented in CVC4
- Highly competitive for single invocation properties

$\Rightarrow$ *For more details, see CAV 15 paper*

"Counterexample Guided Quantifier Instantiation for Synthesis in SMT"

with Morgan Deters, Viktor Kuncak, Cesare Tinelli, and Clark Barrett

Thanks!

- CVC4 publicly available at:

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

- Handles inputs in the sygus language format *.sl
  - Techniques in this presentation enabled by argument "--cegqi-si"