Counterexample-Guided Quantifier Instantiation for Synthesis in SMT

Andrew Reynolds, Morgan Deters, Viktor Kuncak,
Cesare Tinelli, and Clark Barrett

Kumar Madhukar

TCS Research, Pune

Formal Methods Update Meet, IIT Mandi, 17-18 July, 2017

Outline

- The Problem
- Restrictions
- Solutions

The (Synthesis) Problem

- Synthesize a function that meets a given specifications.
- Example - Synthesize f such that:
  - $f(x_1, x_2) \geq x_1 \ \wedge$
  - $f(x_1, x_2) \geq x_2 \ \wedge$
  - $f(x_1, x_2) \approx x_1 \vee f(x_1, x_2) \approx x_2$
- Applicable in synthesis of functional programs, program sketching, synthesis of reactive systems, etc.

If P is a formula that encodes the specification,

$$P[f, x_1, x_2] = f\left(x_1, x_2\right) \geq x_1 \wedge f\left(x_1, x_2\right) \geq x_2 \wedge \\ \left(f\left(x_1, x_2\right) \approx x_1 \vee f\left(x_1, x_2\right) \approx x_2\right)$$

then we must have

$$\forall x_1 x_2 .\ P[f, x_1, x_2]$$

And the question that we are asking is

$$\exists f .\ \forall x_1 x_2 .\ P[f, x_1, x_2]$$

- Or, more generally,

$$\underbrace{\exists f.}_{\text{Exists a function s.t.}} \quad \underbrace{\forall x_1, x_2, \ldots x_n . P(f, x_1, x_2, \ldots x_n)}_{\text{for all } \bar{x}\text{, P(f,}\bar{x}\text{) is true}}$$

- An SMT solver may treat f as an uninterpreted function, but the real challenge here is the universal quantification over $\bar{x}$.
- The solver must construct (a finite representation of) an interpretation for f which is *true* for all $\bar{x}$.

- In contrast, there are effective techniques to show **un**satisfiability of universally quantified formulas.
- SMT solvers use instantiation-based methods - generate ground instances until a refutation is found.
- Can we transform our problem into one of checking unsatisfiability?

If satisfiability $(F) \Rightarrow$ validity (F),

$$\boxed{(F \text{ is sat}) \Leftrightarrow (\neg F \text{ is not valid}) \Leftrightarrow (\neg F \text{ is unsatisfiable})}$$

Restriction

1. Satisfiability $\Rightarrow$ Validity

   - In other words, we will only consider theories that are satisfaction complete wrt the formulas we are interested in.
   - Most theories used in SMT (e.g. various theories of integers, reals, strings, algebraic datatypes, bit-vectors, etc.) are satisfaction complete wrt the class of closed first-order formulas.

$$\boxed{\exists f \,.\, \forall \bar{x} \,.\, P(f, \bar{x})} \quad \text{sat}$$

$$\downarrow \text{negate}$$

$$\boxed{\neg\, \exists f \,.\, \forall \bar{x} \,.\, P(f, \bar{x})} \quad \text{unsat}$$

$$\downarrow \text{push } \neg$$

$$\boxed{\forall f \,.\, \exists \bar{x} \,.\, \neg\, P(f, \bar{x})} \quad \text{unsat}$$

- Another challenge: Negation introduces second-order universal quantification (over function f).
- What if we restrict ourselves to the class of synthesis problems $\exists f \,.\, \forall \bar{x} \,.\, P[f, \bar{x}]$, where every occurrence of f in P is of the form $f(\bar{x})$.
- In that case, we can transform the synthesis problem to: $\forall \bar{x} \,.\, \exists y \,.\, Q[\bar{x}, y]$.

Restrictions

1. Satisfiability $\Rightarrow$ Validity
   - In other words, we will only consider theories that are satisfaction complete wrt the formulas we are interested in
   - Most theories used in SMT (e.g. various theories of integers, reals, strings, algebraic datatypes, bit-vectors, etc.) are satisfaction complete wrt the class of closed first-order formulas.
2. P consists of single-invocation properties

$$f(x_1, x_2) \geq x_1 \wedge f(x_1, x_2) \geq x_2 \wedge (f(x_1, x_2) \approx x_1 \vee f(x_1, x_2) \approx x_2)$$

$$c(x_1, x_2) \approx c(x_2, x_1)$$

Recall

Synthesis conjecture:

$$\exists f.\ \forall x_1 \ldots x_n.\ P[f, x_1, \ldots, x_n]$$

- avoid second-order quantification, and
- solve an unsatisfiability (universal quantification) problem instead of a satisfiability one.

So far..

$$\boxed{\exists f \,.\, \forall \bar{x} \,.\, P(f, \bar{x}) \quad \text{sat}}$$

$$\big\downarrow \text{(single-invocation property)}$$

$$\boxed{\forall \bar{x} \,.\, \exists g \,.\, P(g, \bar{x}) \quad \text{sat}}$$

$$\big\downarrow \text{(satisfaction-complete theory)}$$

$$\boxed{\forall \bar{x} \,.\, \exists g \,.\, P(g, \bar{x}) \quad \text{valid}}$$

$$\big\downarrow \text{(negate)}$$

$$\boxed{\neg \forall \bar{x} \,.\, \exists g \,.\, P(g, \bar{x}) \quad \text{unsat}}$$

$$\big\downarrow \text{(push } \neg\text{)}$$

$$\boxed{\exists \bar{x} \,.\, \forall g \,.\, \neg P(g, \bar{x}) \quad \text{unsat}}$$

Our first example

$$\exists f.\ \forall x_1 x_2.(f(x_1, x_2) \geq x_1 \wedge f(x_1, x_2) \geq x_2 \wedge (f(x_1, x_2) \approx x_1 \vee f(x_1, x_2) \approx x_2))\ \text{sat}$$

$\downarrow$ (single-invocation property)

$$\forall x_1 x_2.\ \exists g.\ (g \geq x_1 \wedge g \geq x_2 \wedge (g \approx x_1 \vee g \approx x_2))\ \text{sat}$$

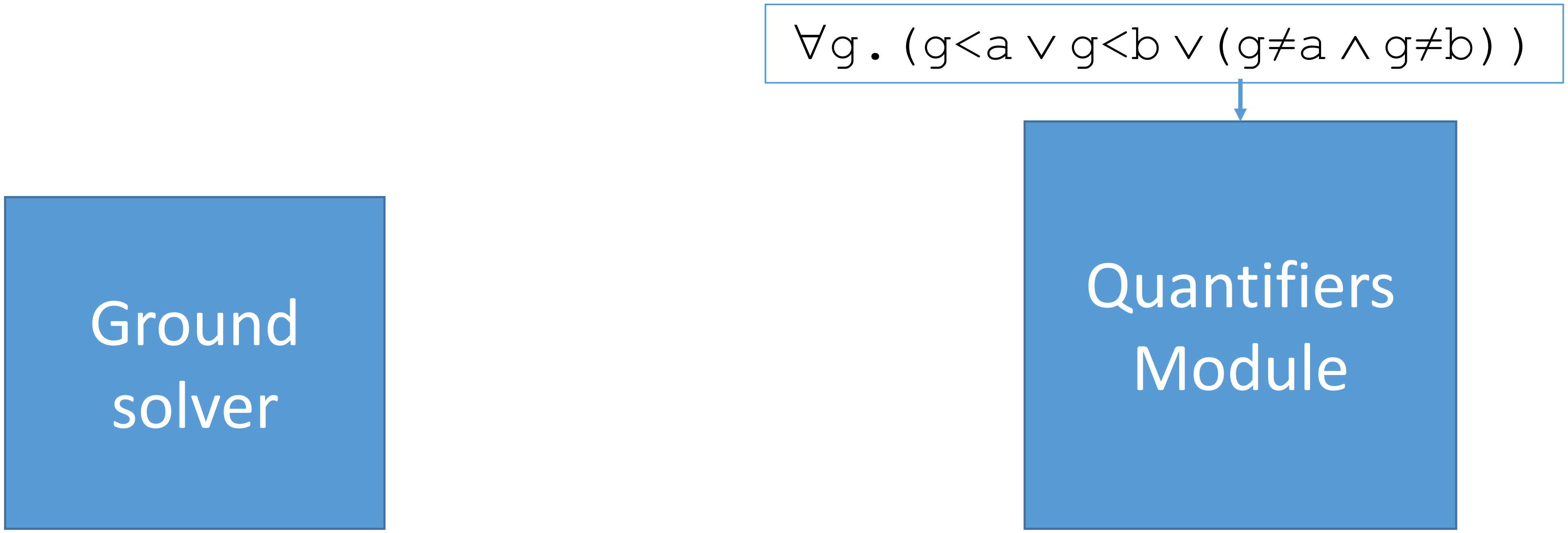
$\downarrow$ negate (satisfaction-complete theory)

$$\exists x_1 x_2.\ \forall g.\ (g < x_1 \vee g < x_2 \vee (g \not\approx x_1 \wedge g \not\approx x_2))\ \text{unsat}$$

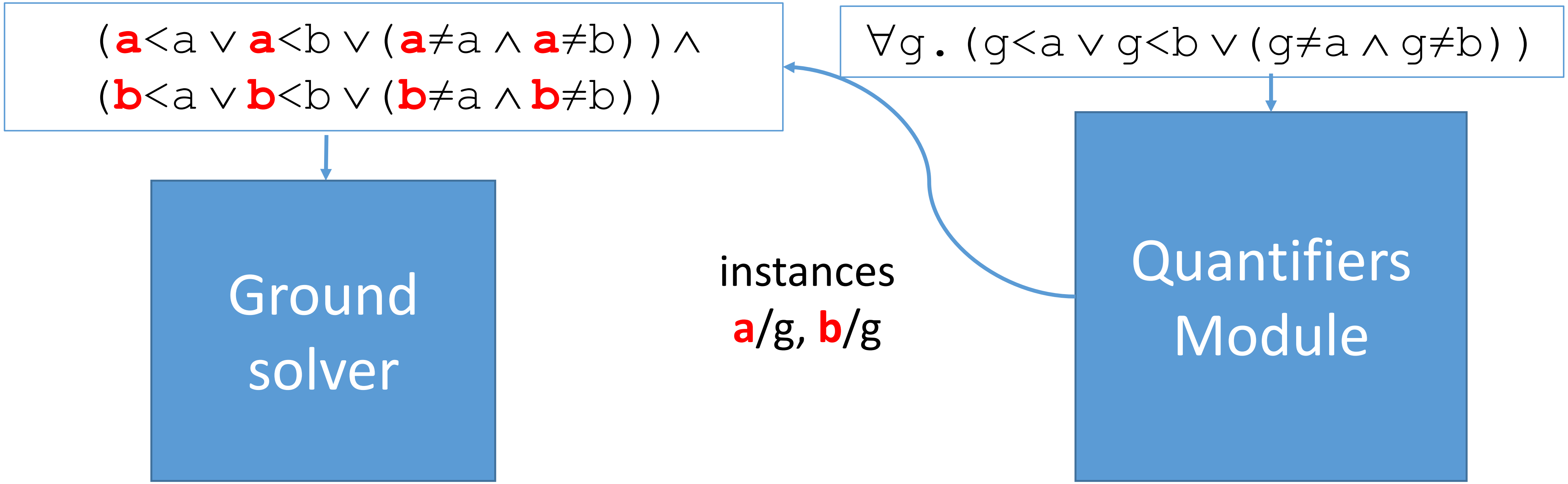
$\downarrow$ Skolemize, for fresh a, b

$$\forall g.\ (g < a \vee g < b \vee (g \not\approx a \wedge g \not\approx b))\ \text{unsat}$$

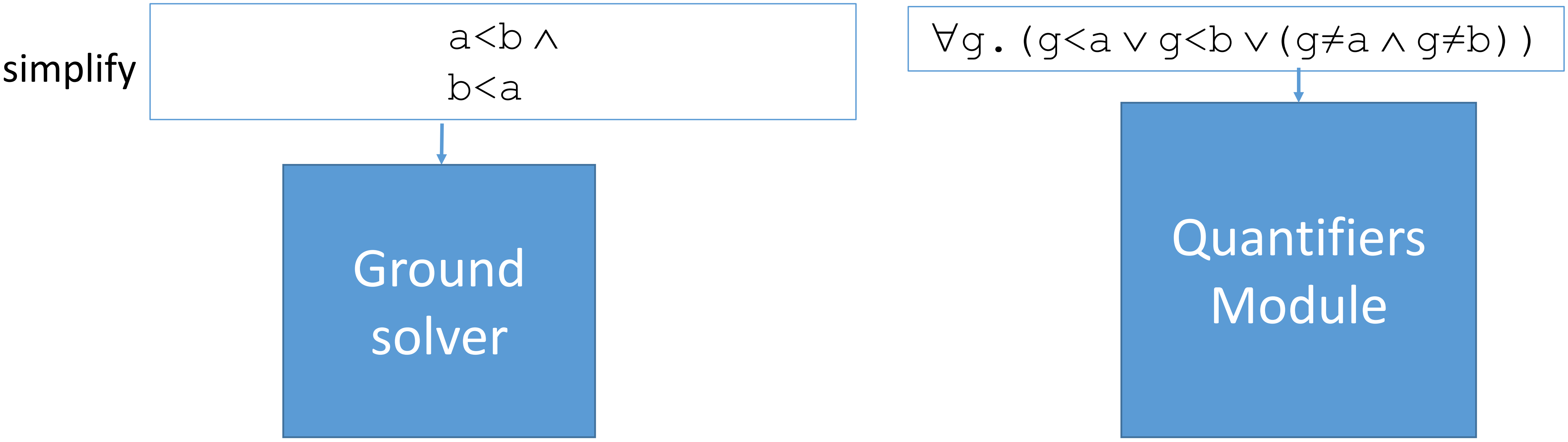
Solving Max Example



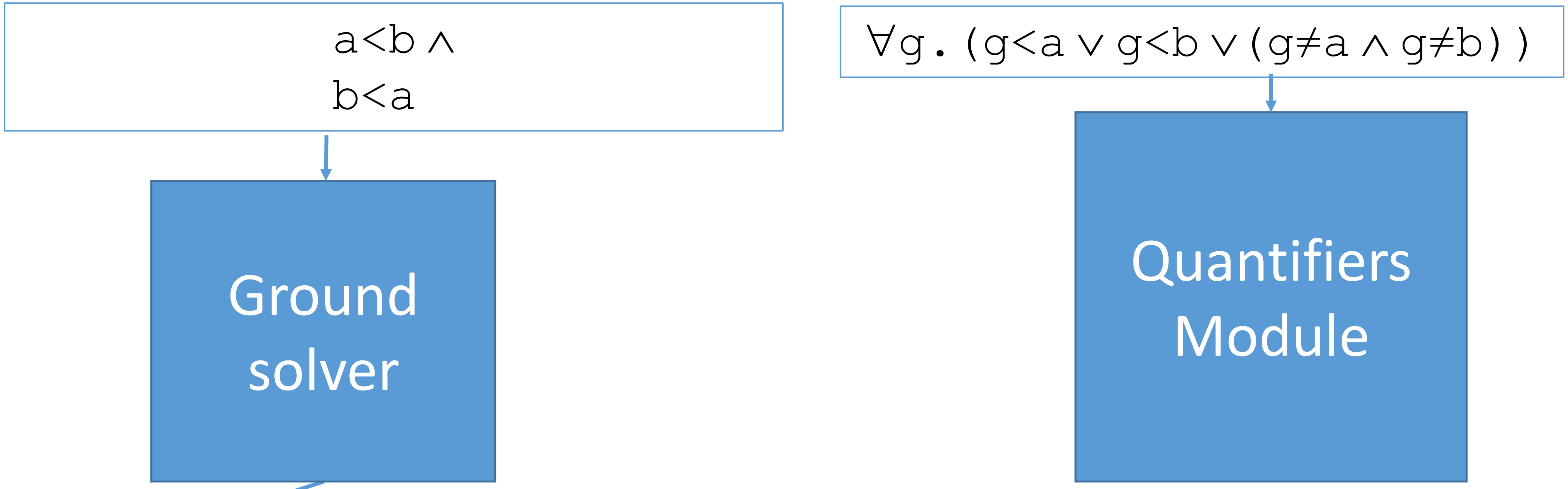
Solving Max Example



Solving Max Example

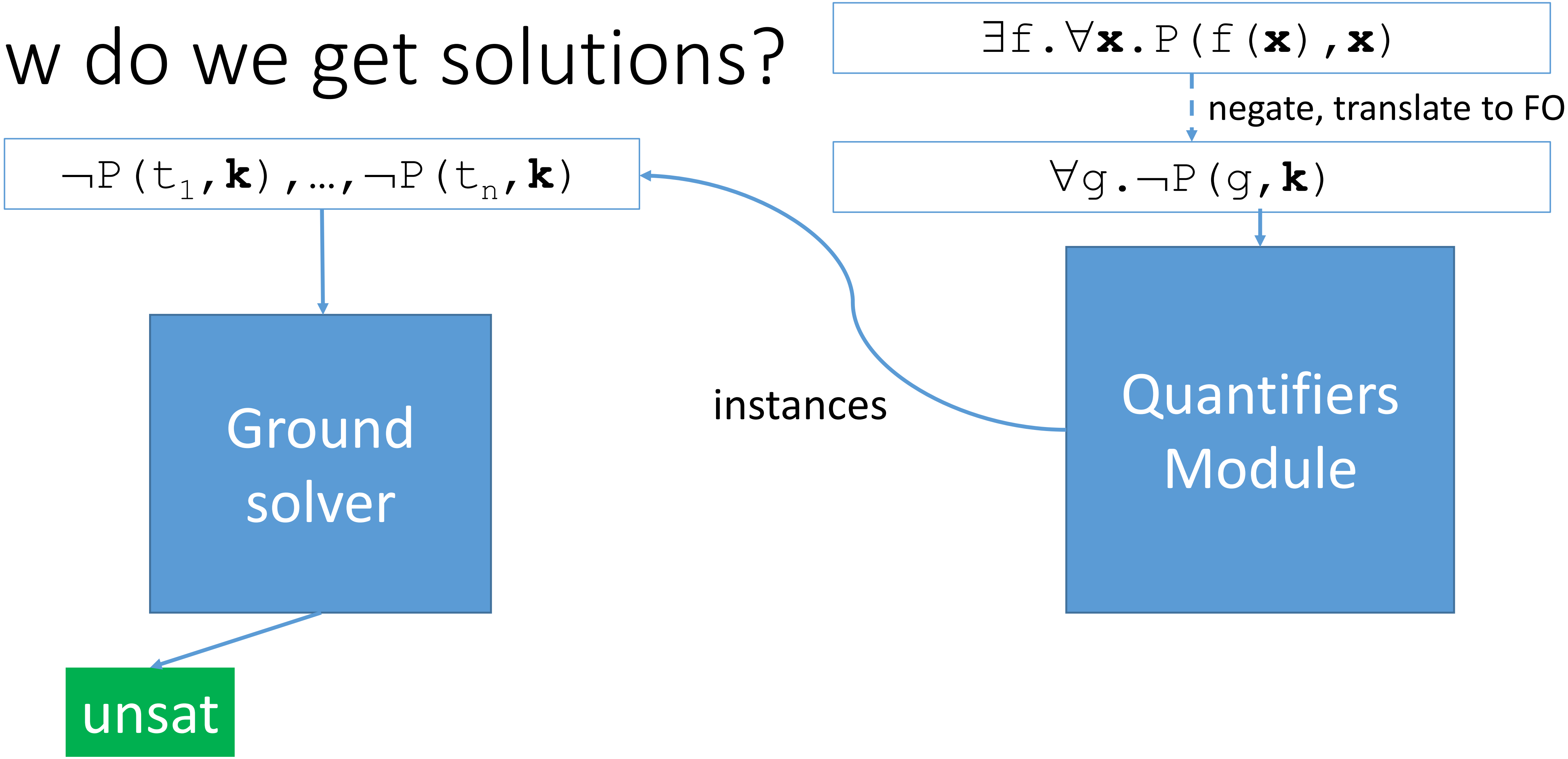


Solving Max Example



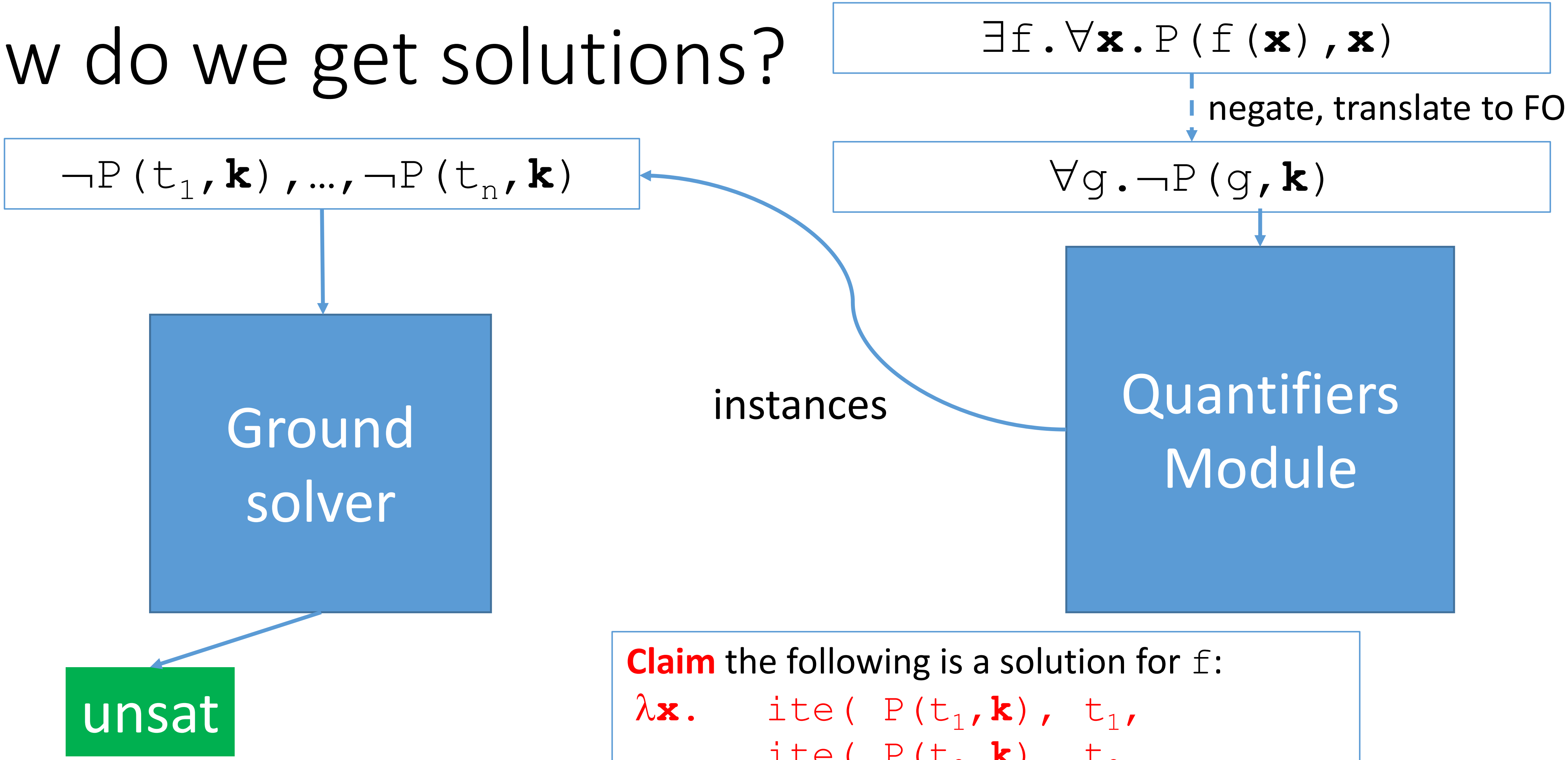
unsat ⇒ ∀g.(g<a ∨ g<b ∨ (g≠a ∧ g≠b)) is unsatisfiable,
implies original synthesis conjecture has a solution

How do we get solutions?



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

How do we get solutions?



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

Claim the following is a solution for f:

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

Why is this a solution?

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

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 $\mathtt{f}$:

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

Why is this a solution?

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

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

Claim the following is a solution for f:

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

If P holds for t_1, return t_1

Why is this a solution?

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

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 $\mathtt{f}$:

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

If $\mathtt{P}$ holds for $\mathtt{t}_2$, return $\mathtt{t}_2$

Why is this a solution?

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

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:

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

If ${\color{red} P}$ **holds for** ${\color{red} t_{n-1}}$, return t_{n-1}

Why is this a solution?

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

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

Claim the following is a solution for f:

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

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

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:

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

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:

$\lambda xy.\ ite(\ x \geq x \land x \geq y \land (x=x \lor x=y),\ x,$
$y)\ldots)$

Solution for Max Example

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

Found
$$\begin{matrix} \neg(a\geq a \wedge a\geq b \wedge (a=a \vee a=b)), \\ \neg(b\geq a \wedge b\geq b \wedge (b=a \vee b=b)) \end{matrix} \quad \models \mathtt{false}$$

Claim the following is a solution for $\mathtt{f}$:

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

Lifting the single-invocation property restriction

- Can we still refute negated synthesis conjectures?
- Yes, under syntactic restrictions.

Example: Syntax-Guided Synthesis

- Syntactic restriction for the solution space, expressed by these algebraic datatypes:

$$
\begin{aligned}
S &:= t_1 \mid t_2 \mid \mathit{zero} \mid \mathit{one} \mid \mathit{plus}(S, S) \mid \mathit{minus}(S, S) \mid \mathit{if}\,(C, S, S) \\
C &:= \mathit{leq}(S, S) \mid \mathit{eq}(S, S) \mid \mathit{and}(C, C) \mid \mathit{not}(C)
\end{aligned}
$$

Example: Syntax-Guided Synthesis

- Syntactic restriction for the solution space, expressed by these algebraic datatypes:

$$S := t_1 \mid t_2 \mid zero \mid one \mid plus(S, S) \mid minus(S, S) \mid if\,(C, S, S)$$
$$C := leq(S, S) \mid eq(S, S) \mid and(C, C) \mid not(C)$$

- And an interpretation of these datatypes in terms of the original theory.

1. $ev^{S \times Int \times Int \to Int}$: embedding S in Int.
2. $ev^{C \times Int \times Int \to Bool}$: embedding C in Bool.

The evaluation operators

$$ev(t_1, x, y) \approx x$$

$$ev(zero, x, y) \approx 0$$

$$ev(not(c), x, y) \approx \neg ev(c, x, y)$$

$$ev(and(c_1, c_2), x, y) \approx ev(c_1, x, y) \wedge ev(c_2, x, y)$$

$$ev(plus(s_1, s_2), x, y) \approx ev(s_1, x, y) + ev(s_2, x, y)$$

$$ev(if\,(c, s_1, s_2), x, y) \approx ite(ev(c, x, y), ev(s_1, x, y), ev(s_2, x, y))$$

Another example

$$\boxed{P[c, x_1, x_2] = \; c(x_1, x_2) \approx c(x_2, x_1)}$$

- can be restated as follows, where g is a variable of type S:

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \approx ev(g, x_2, x_1)}$$

- Now, instead of finding a witness for $\exists c.\ \forall x_1 x_2 . P[c, x_1, x_2]$ we will determine the unsatisfiability of $\exists x_1 x_2 .\ \forall g . \neg P_{ev}[g, x_1, x_2]$.

Positive Example : $P[c, x_1, x_2] = \; c(x_1, x_2) \approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \approx ev(g, x_2, x_1)}$$

Model	Added Formula

Positive Example : $P[c, x_1, x_2] = \; c(x_1, x_2) \approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \rightarrow t_1]$	$ev(t_1, a_1, b_1) \not\approx ev(t_1, b_1, a_1)$

Positive Example : $P[c, x_1, x_2] = \ c(x_1, x_2) \approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \ ev(g, x_1, x_2) \approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \not\approx ev(t_1, b_1, a_1)$
$[a_1 \to 1, b_1 \to 0]$	$G \Rightarrow ev(g, 1, 0) \approx ev(g, 0, 1)$

Positive Example : $P[c, x_1, x_2] = \ c(x_1, x_2) \approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \ ev(g, x_1, x_2) \approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \not\approx ev(t_1, b_1, a_1)$
$[a_1 \to 1, b_1 \to 0]$	$G \Rightarrow ev(g, 1, 0) \approx ev(g, 0, 1)$
$[g \to zero]$	$ev(zero, a_2, b_2) \not\approx ev(zero, b_2, a_2)$

Positive Example : $P[c, x_1, x_2] = \; c(x_1, x_2) \approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \not\approx ev(t_1, b_1, a_1)$
$[a_1 \to 1, b_1 \to 0]$	$G \Rightarrow ev(g, 1, 0) \approx ev(g, 0, 1)$
$[g \to zero]$	$ev(zero, a_2, b_2) \not\approx ev(zero, b_2, a_2)$
none	

Positive Example : $P[c, x_1, x_2] = \; c(x_1, x_2) \approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \not\approx ev(t_1, b_1, a_1)$
$[a_1 \to 1, b_1 \to 0]$	$G \Rightarrow ev(g, 1, 0) \approx ev(g, 0, 1)$
$[g \to zero]$	$ev(zero, a_2, b_2) \not\approx ev(zero, b_2, a_2)$
none	

Solution: $c(x_1, x_2) = 0$

Negative Example: $P[c, x_1, x_2] = c(x_1, x_2) \not\approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = ev(g, x_1, x_2) \not\approx ev(g, x_2, x_1)}$$

Negative Example: $P[c, x_1, x_2] = \; c(x_1, x_2) \not\approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \not\approx ev(g, x_2, x_1)}$$

Model	Added Formula

Negative Example: $P[c, x_1, x_2] = \; c(x_1, x_2) \not\approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \not\approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \approx ev(t_1, b_1, a_1)$

Negative Example: $P[c, x_1, x_2] = c(x_1, x_2) \not\approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = ev(g, x_1, x_2) \not\approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \approx ev(t_1, b_1, a_1)$
$[a_1 \to 0, b_1 \to 0]$	$G \Rightarrow ev(g, 0, 0) \not\approx ev(g, 0, 0)$

Negative Example: $P[c, x_1, x_2] = \; c(x_1, x_2) \not\approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \not\approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \approx ev(t_1, b_1, a_1)$
$[a_1 \to 0, b_1 \to 0]$	$G \Rightarrow ev(g, 0, 0) \not\approx ev(g, 0, 0)$
none	

Negative Example: $P[c, x_1, x_2] = \; c(x_1, x_2) \not\approx c(x_2, x_1)$

$$\boxed{P_{ev}[g, x_1, x_2] = \; ev(g, x_1, x_2) \not\approx ev(g, x_2, x_1)}$$

Model	Added Formula
$[g \to t_1]$	$ev(t_1, a_1, b_1) \approx ev(t_1, b_1, a_1)$
$[a_1 \to 0, b_1 \to 0]$	$G \Rightarrow ev(g, 0, 0) \not\approx ev(g, 0, 0)$
none	

No Solution

The procedure has following properties:

- *Solution Soundness*: Every term that it returns can be mapped to a solution of the original synthesis conjecture $\exists f\,.\forall \bar{x}.\ P[f, \bar{x}]$.
- *Refutation Soundness*: If it does not find a solution (up to a given length), the original conjecture has no solution under the syntactic restrictions up to that length.
- *Solution Completeness*: If the original synthesis conjecture has a solution under these restrictions, the procedure will find one.

To conclude

- Refutation based approach for syntax-guided synthesis.
- Implemented in CVC4; winner in General and LIA tracks at SyGuS-Comp 2014.
- Single-invocation - appears to be restrictive but not quite so in practice; 176 benchmarks out of 243 at SyGuS-Comp 2014 were single-invocation.

Thank you.

Questions?