Quantifier Instantiation Beyond E-Matching

Andrew Reynolds

SMT Workshop

July 22, 2017

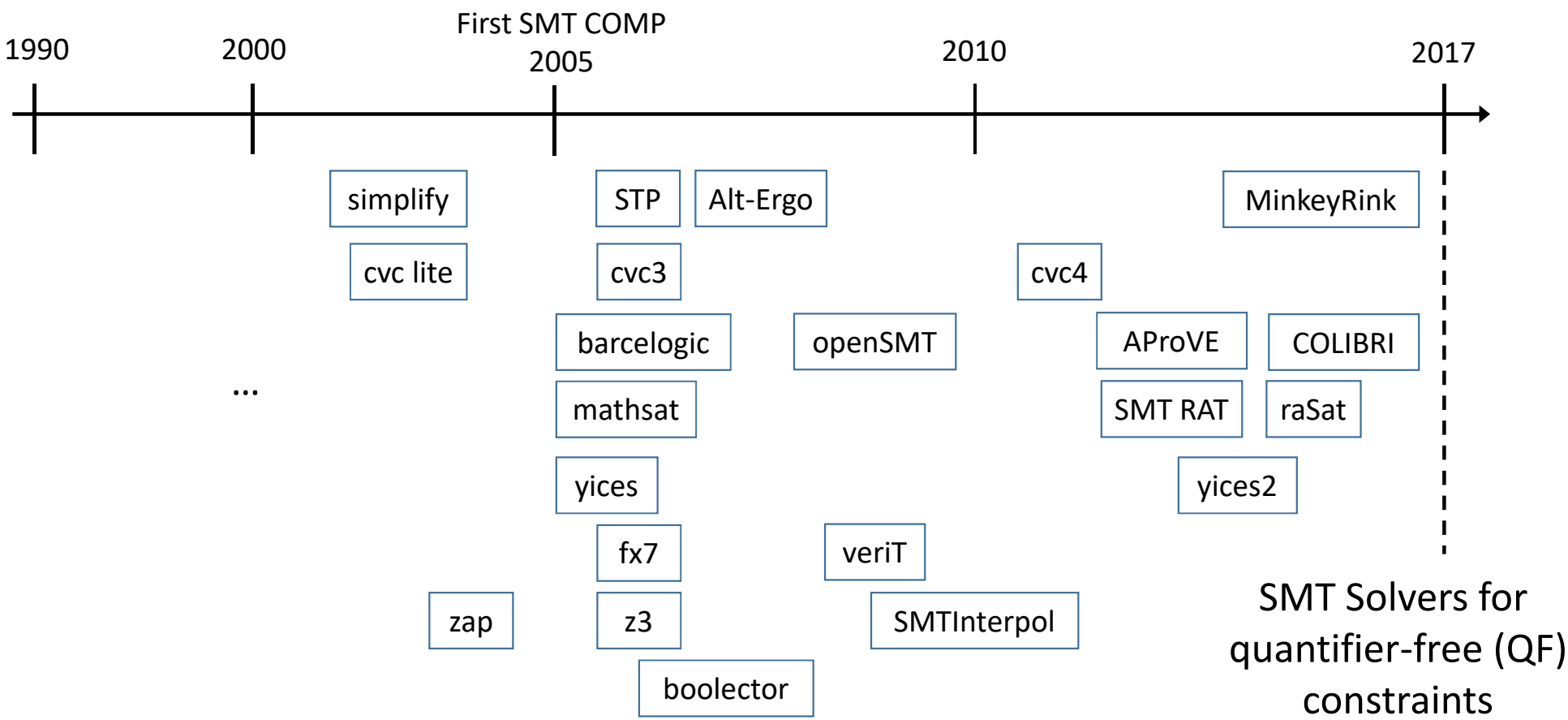
The University of Iowa

Acknowledgements

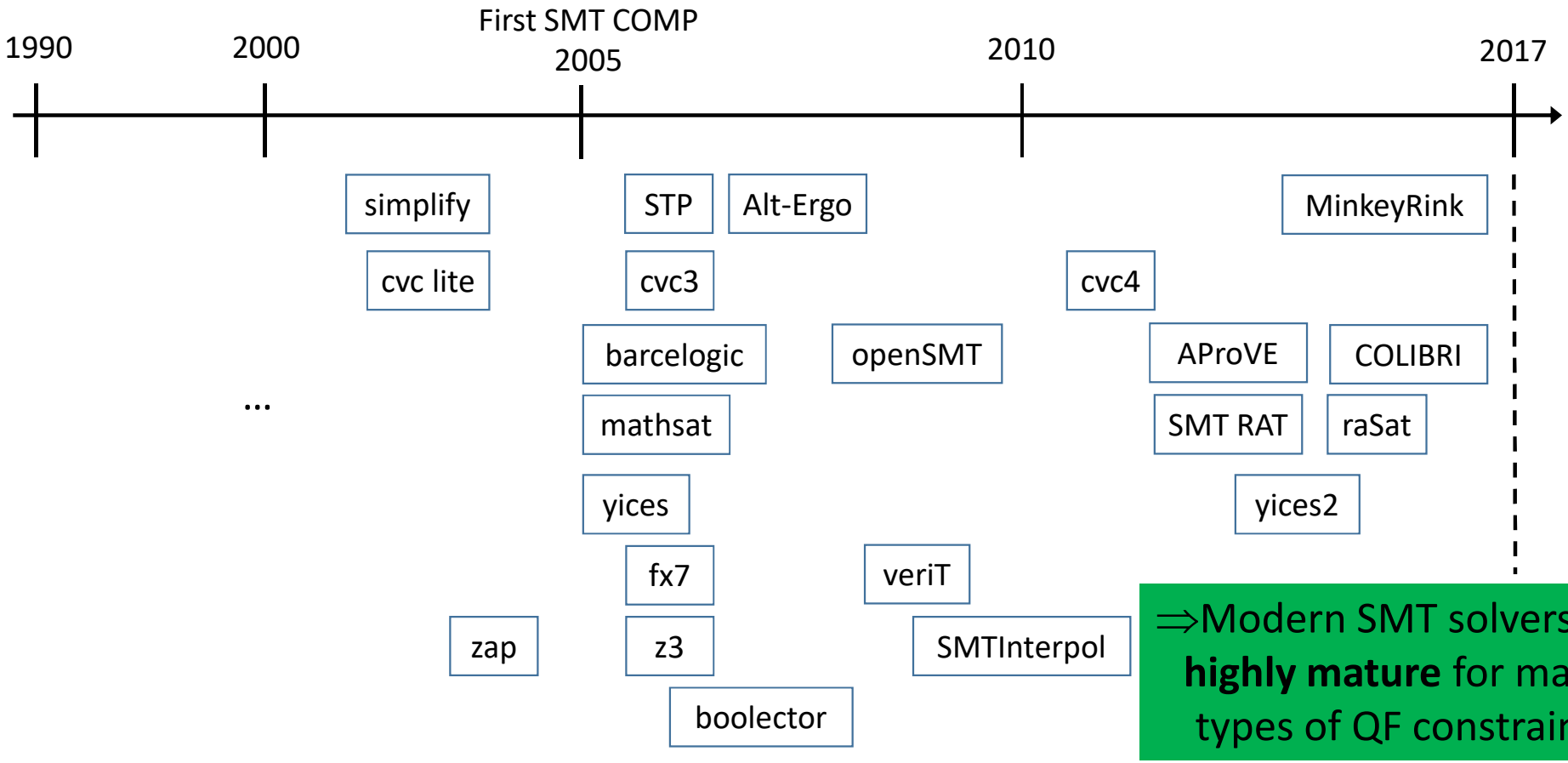
- Thanks to collaborators:
  - Cesare Tinelli, Clark Barrett, Viktor Kuncak, Tim King, Morgan Deters, Francois Bobot, Amit Goel, Sava Krstic, Leonardo de Moura, Thomas Wies, Kshitij Bansal, Haniel Barbosa, Pascal Fontaine

- Past and present members of development team of CVC4:
  - Dejan Jovanovic, Liana Hadarean, Tianyi Liang, Christopher Conway, Guy Katz, Andres Noetzli, Paul Meng

Timeline of Modern SMT Solvers for QF (Approximate)

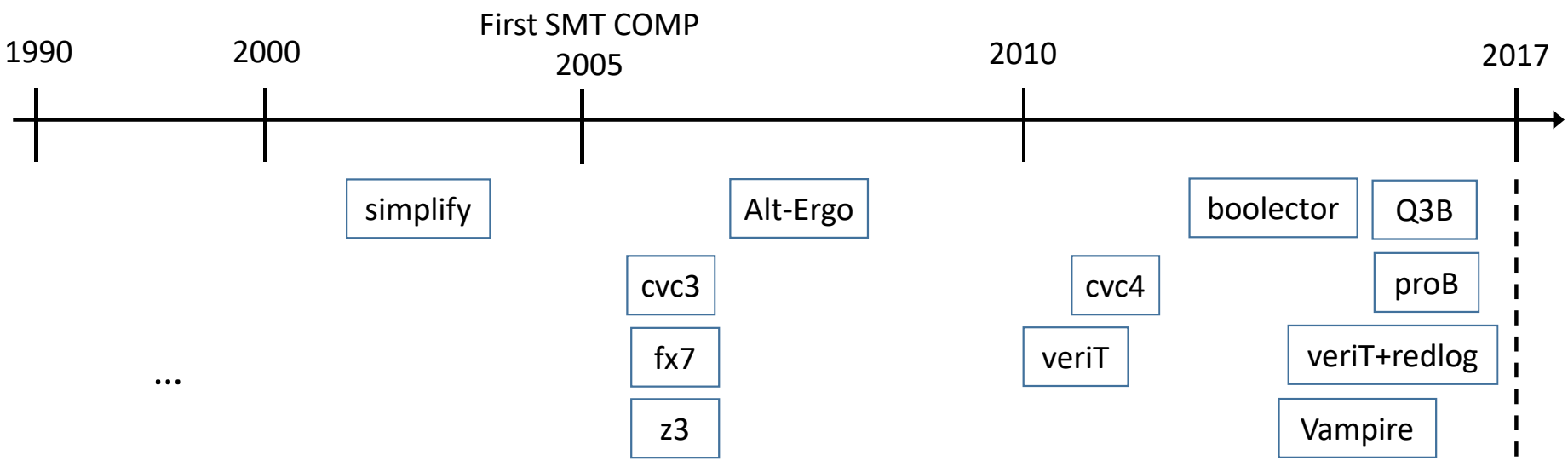


Timeline of Modern SMT Solvers for QF (Approximate)



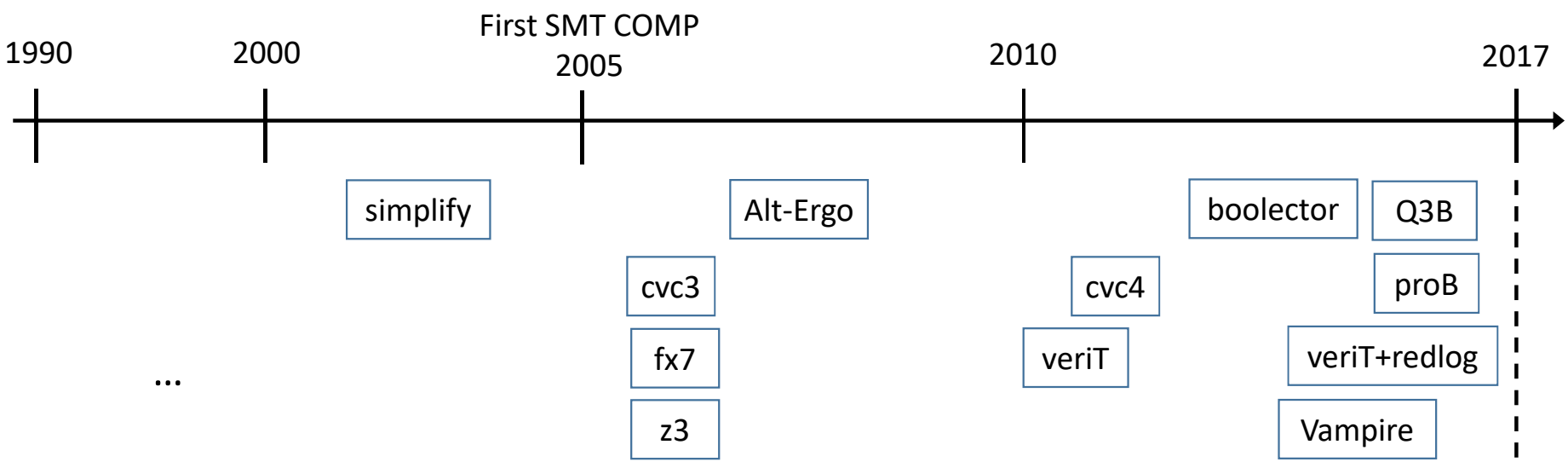
⇒Modern SMT solvers are **highly mature** for many types of QF constraints

Timeline of Modern SMT Solvers for ∀ (Approximate)



⇒ **Subset** of these solvers support ∀+T-constraints

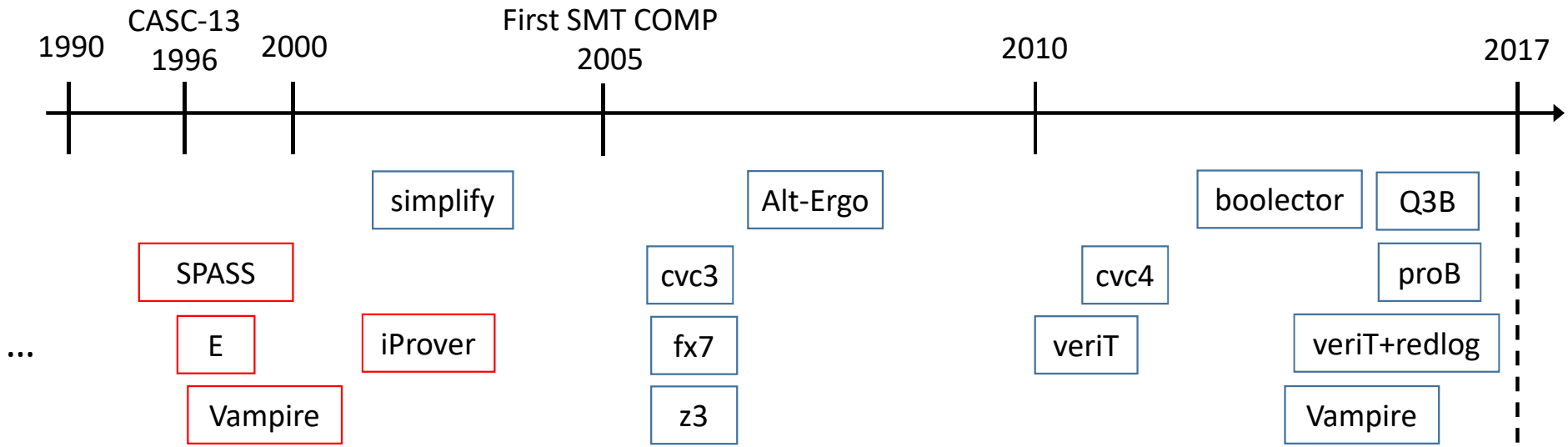
Timeline of Modern SMT Solvers for ∀ (Approximate)



⇒ **Subset** of these solvers support ∀+T-constraints

⇒SMT Solvers are a **developing technology** for ∀+T-constraints

Timeline of Modern SMT Solvers for ∀ (Approximate)



⇒SMT Solvers are a **developing technology** for ∀+T-constraints

⇒ Build on earlier work from ATP community for ∀-constraints

Applications of ∀ in SMT

- Are used for:
  - **Automated theorem proving**:
    - Background axioms {∀x.g(e,x)=g(x,e)=x, ∀x.g(x,g(y,z))=g(g(x,y),x),∀x.g(x,i(x))=e}
  - **Software verification**:
    - Unfolding ∀x.foo(x)=bar(x+1), code contracts ∀x.pre(x)⇒post(f(x))
    - Frame axioms ∀x.x≠t ⇒ A'(x)=A(x)
  - **Function Synthesis**:
    - Synthesis conjectures ∀i:input.∃o:output.R[o,i]
  - **Planning**:
    - Specifications ∃p:plan.∀t:time.F[P,t]

Solvers for $\forall$

- First order theorem provers focus on $\forall$ reasoning
  …but have been extended in the past decade to theory reasoning

- SMT solvers focus mostly on quantifier-free theory reasoning
  …but have been extended in the past decade to $\forall$ reasoning

Solvers for ∀

- First order theorem provers focus on ∀ reasoning
  …but have been extended in the past decade to theory reasoning:
  - **Vampire, E, SPASS**
    - First-order resolution + superposition **[Robinson 65, Nieuwenhuis/Rubio 99, Prevosto/Waldman 06]**
    - AVATAR **[Voronkov 14, Reger et al 15]**
  - **iProver**
    - InstGen calculus **[Ganzinger/Korovin 03]**
  - **Princess, Beagle**, …
- SMT solvers focus mostly on quantifier-free theory reasoning
  …but have been extended in the past decade to ∀ reasoning

Solvers for ∀

- First order theorem provers focus on ∀ reasoning
  …but have been extended in the past decade to theory reasoning:
  - **Vampire, E, SPASS**
    - First-order resolution + superposition **[Robinson 65, Nieuwenhuis/Rubio 99, Prevosto/Waldman 06]**
    - AVATAR **[Voronkov 14, Reger et al 15]**
  - **iProver**
    - InstGen calculus **[Ganzinger/Korovin 03]**
  - **Princess, Beagle**, …
- SMT solvers focus mostly on quantifier-free theory reasoning
  …but have been extended in the past decade to ∀ reasoning:
  - **Z3, CVC4, VeriT, Alt-Ergo**
    - Some superposition-based **[deMoura et al 09]**
    - Mostly instantiation-based **[Detlefs et al 05, deMoura et al 07, Ge et al 07, …]**

Solvers for ∀

- First order theorem provers focus on ∀ reasoning
  …but have been extended in the past decade to theory reasoning:
  - **Vampire, E, SPASS**
    - First-order resolution + superposition **[Robinson 65, Nieuwenhuis/Rubio 99, Prevosto/Waldman 06]**
    - AVATAR **[Voronkov 14, Reger et al 15]**
  - **iProver**
    - InstGen calculus **[Ganzinger/Korovin 03]**
  - **Princess, Beagle**, …
- SMT solvers focus mostly on quantifier-free theory reasoning
  …but have been extended in the past decade to ∀ reasoning:
  - **Z3, CVC4, VeriT, Alt-Ergo**
    - Some superposition-based **[deMoura et al 09]**
    - Mostly **instantiation-based** **[Detlefs et al 05, deMoura et al 07, Ge et al 07, …]**

$\Rightarrow$ Focus of this talk

SMT Solvers for ∀ using Quantifier Instantiation

- Traditionally:
  - E-matching **[Detlefs et al 2005,Bjorner et al 2007, Ge et al 2007]**

Implemented in

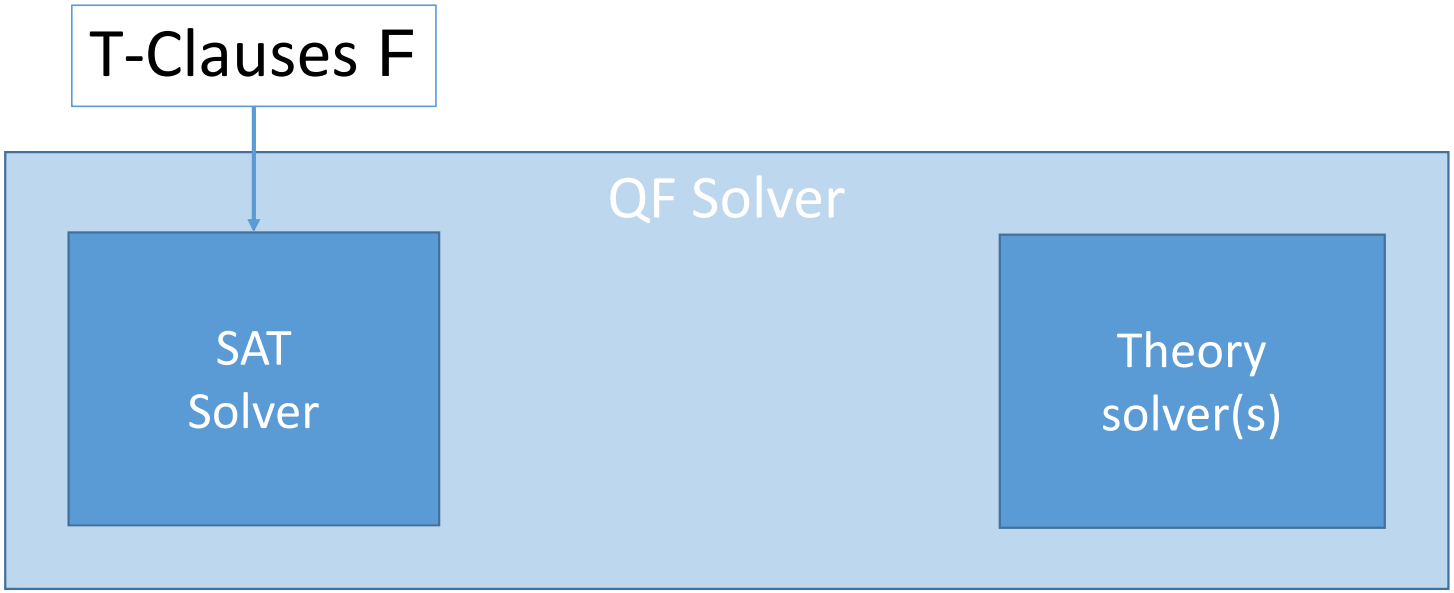
simplify, cvc3, z3, FX7, Alt-Ergo, Princess, cvc4, veriT

SMT Solvers for ∀ using Quantifier Instantiation

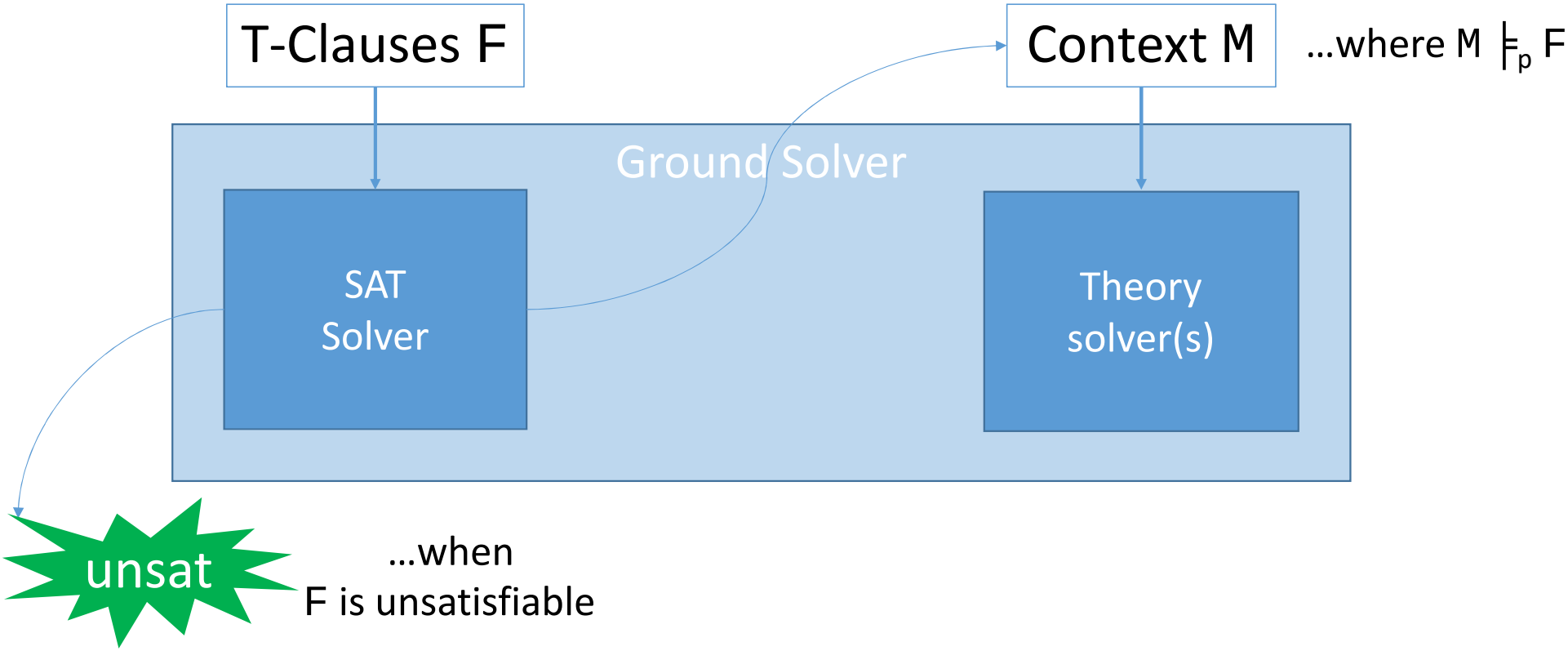
- Traditionally:
  - E-matching [Detlefs et al 2005,Bjorner et al 2007, Ge et al 2007]
- More recently:
  - Model-Based Instantiation [Ge et al 2009, Reynolds et al 2013]
  - Conflict-Based Instantiation [Reynolds et al 2014, Barbosa et al 2017]
  - Theory-specific Approaches
    - Linear arithmetic [Bjorner 2012, Reynolds et al 2015, Janota et al 2015]
    - Bit-Vectors [Wintersteiger et al 2013, Dutertre 2015]

Implemented in
simplify, cvc3, z3, FX7, Alt-Ergo, Princess, cvc4, veriT
z3, cvc4
cvc4, veriT
z3, cvc4, yices, veriT+redlog

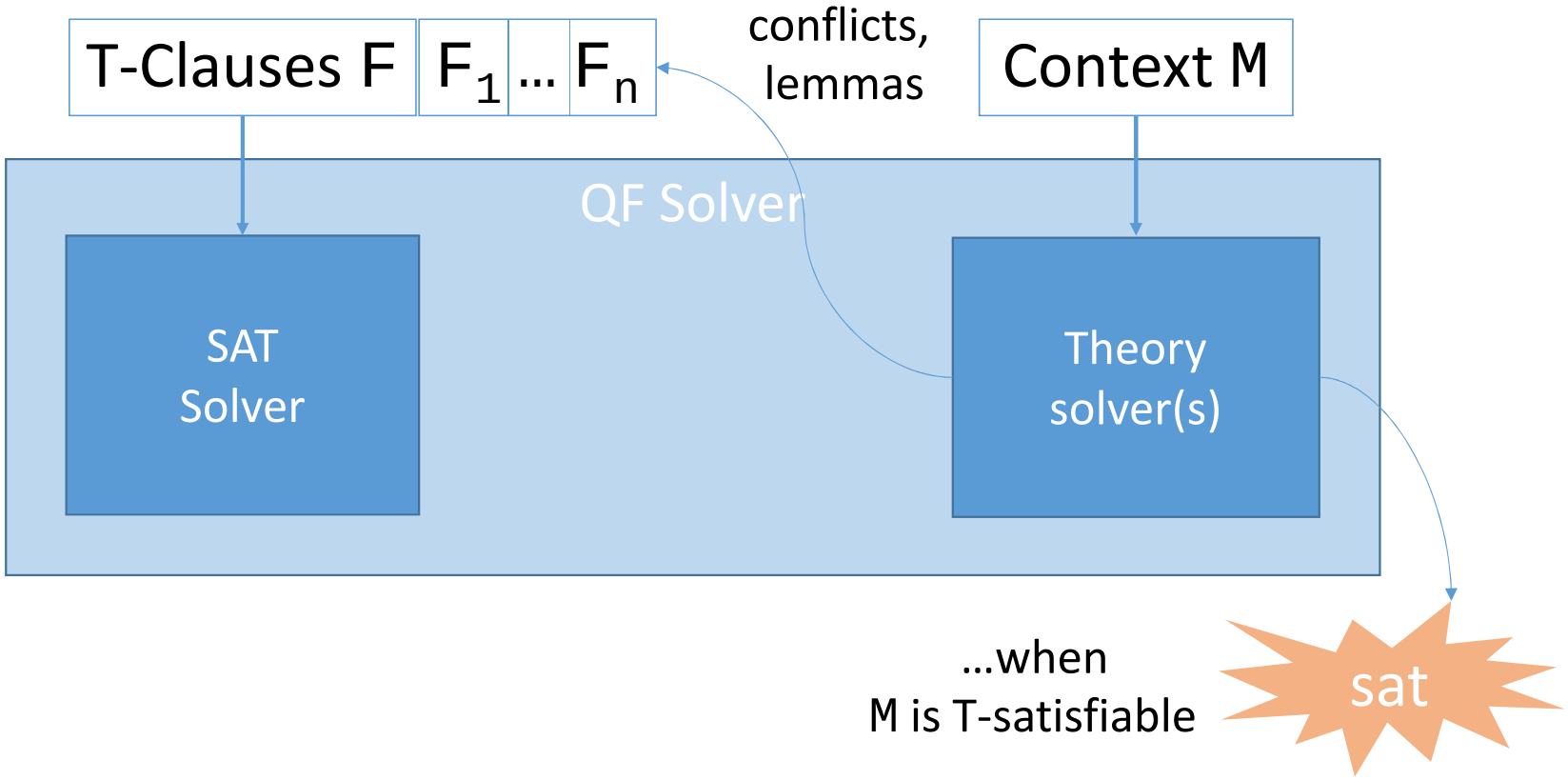
DPLL(T)-Based SMT Solvers



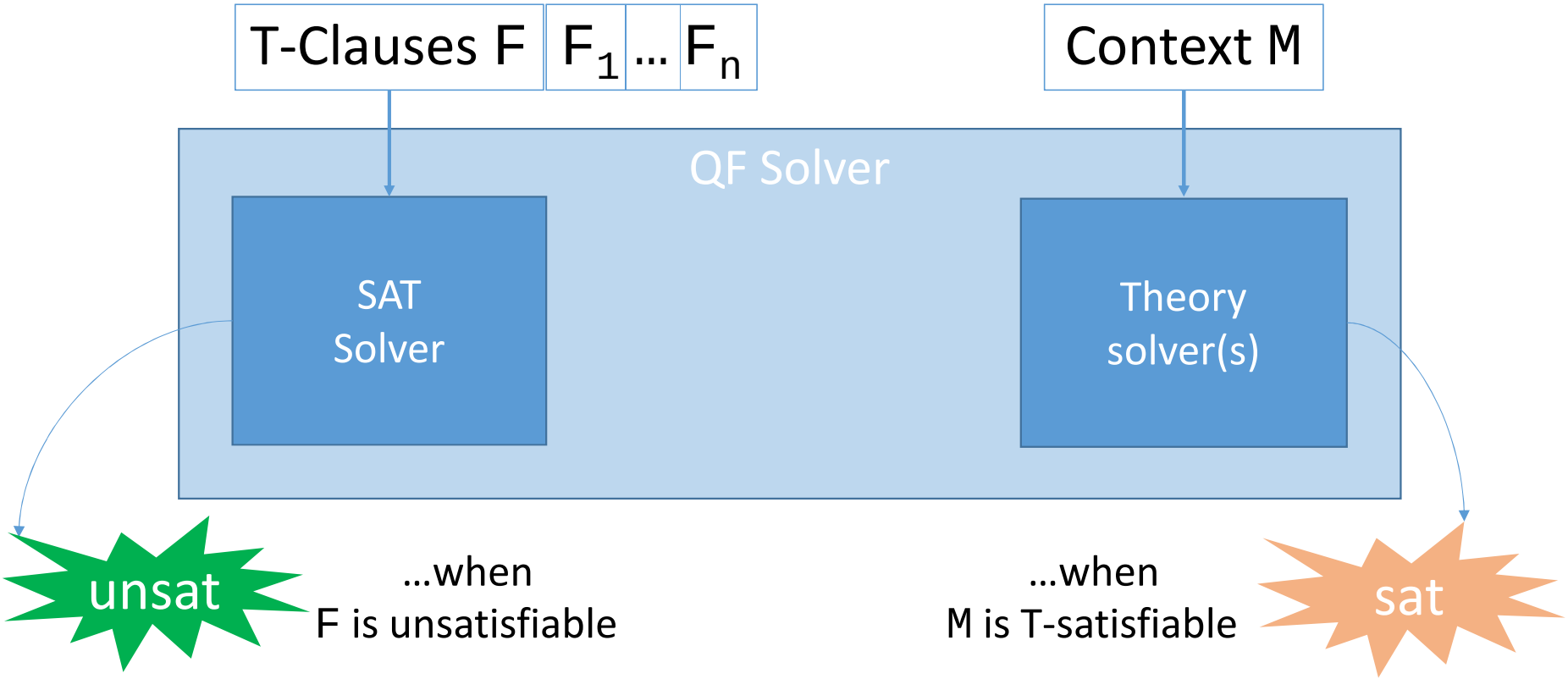
DPLL(T)-Based SMT Solvers



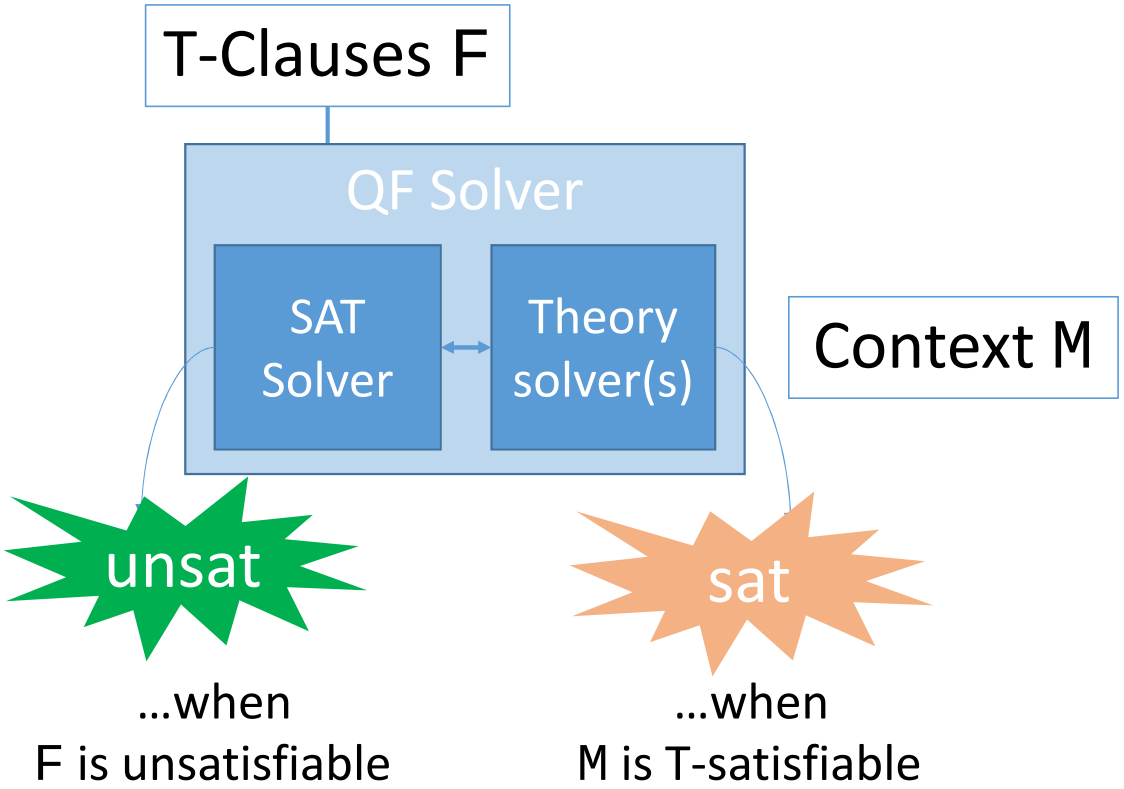
DPLL(T)-Based SMT Solvers



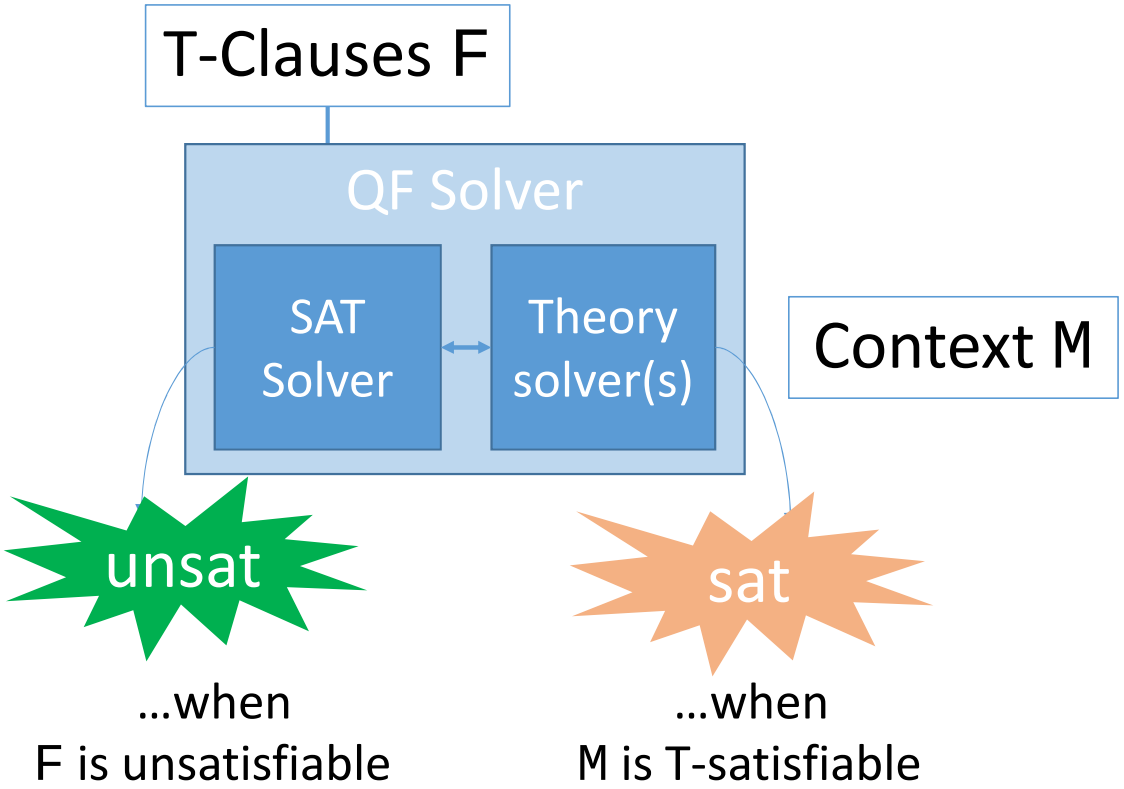
DPLL(T)-Based SMT Solvers



DPLL(T)-Based SMT Solvers + ∀ Instantiation

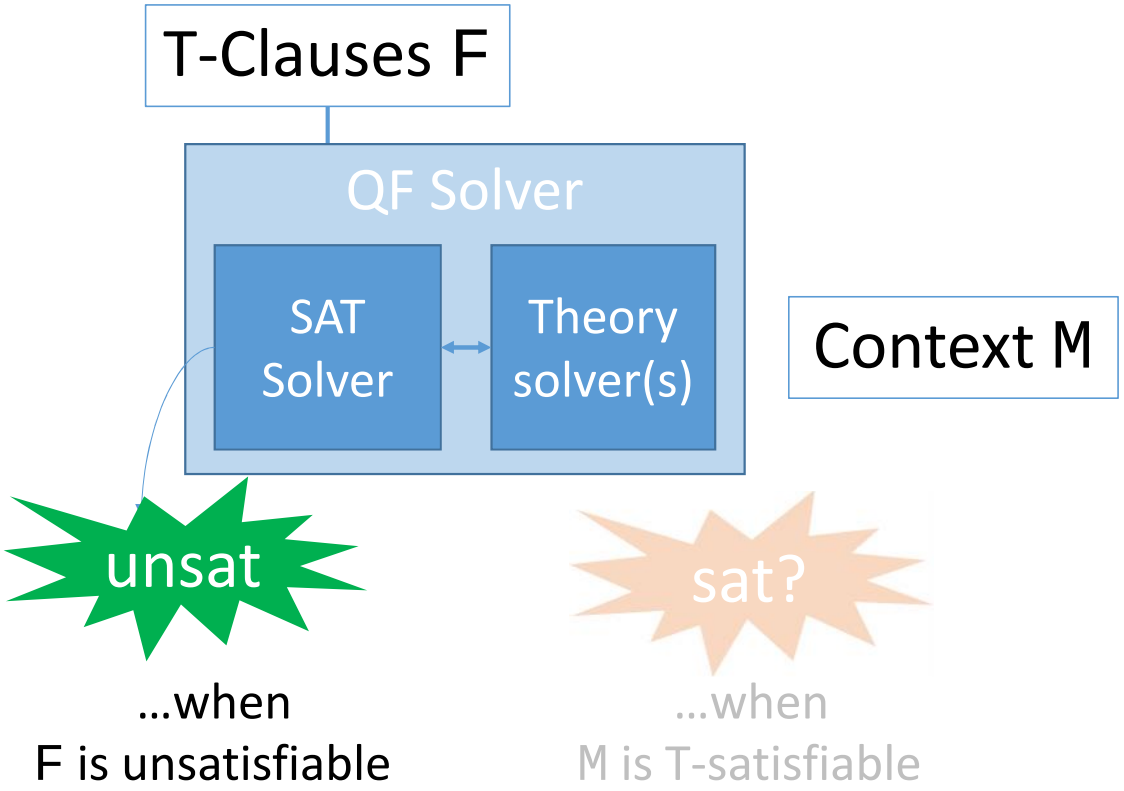


DPLL(T)-Based SMT Solvers + ∀ Instantiation



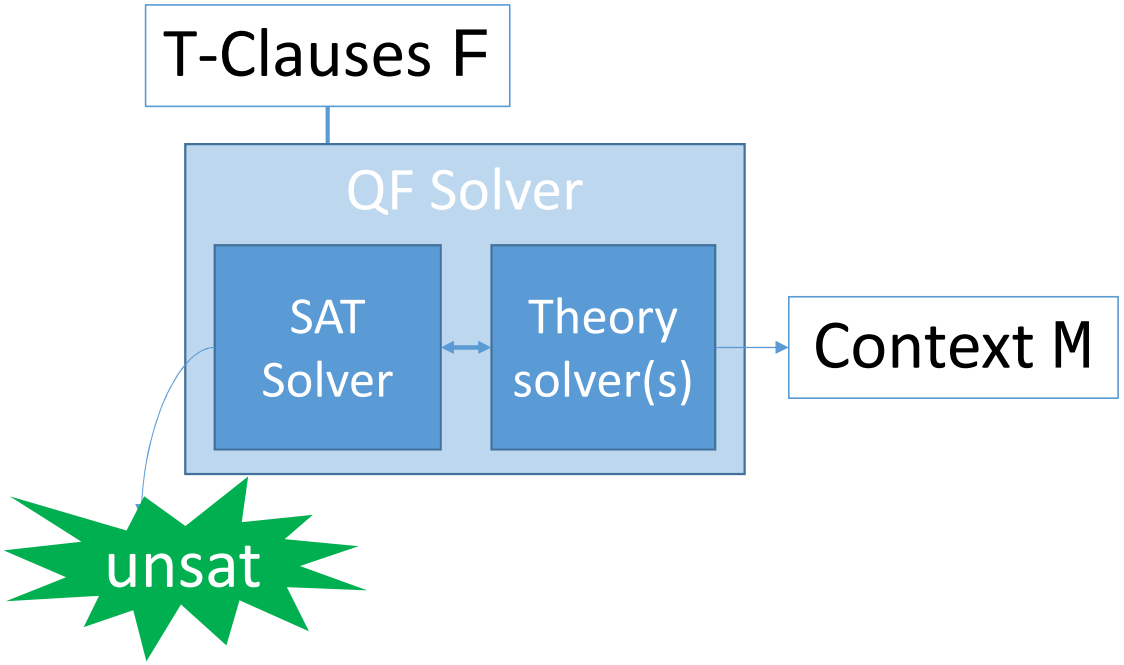
When M contains quantified formulas…

DPLL(T)-Based SMT Solvers + ∀ Instantiation



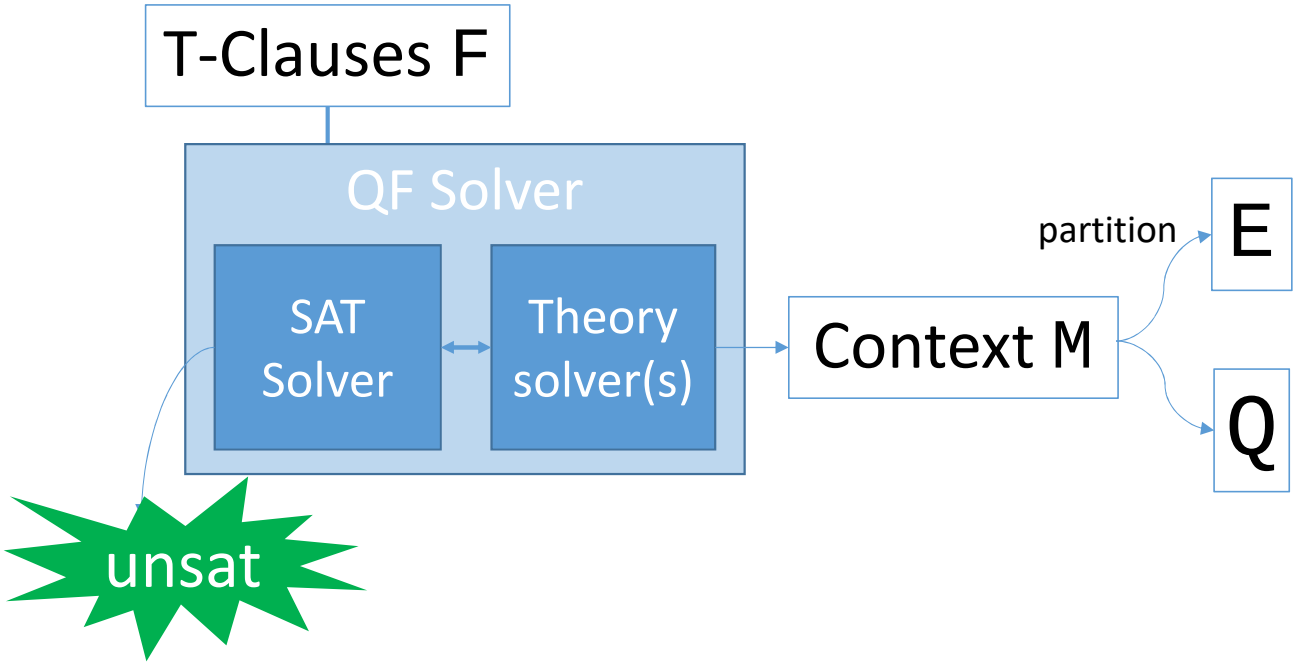
…cannot use QF procedure for establishing M is sat

DPLL(T)-Based SMT Solvers + ∀ Instantiation



…when
F is unsatisfiable

DPLL(T)-Based SMT Solvers + ∀ Instantiation



…when
F is unsatisfiable

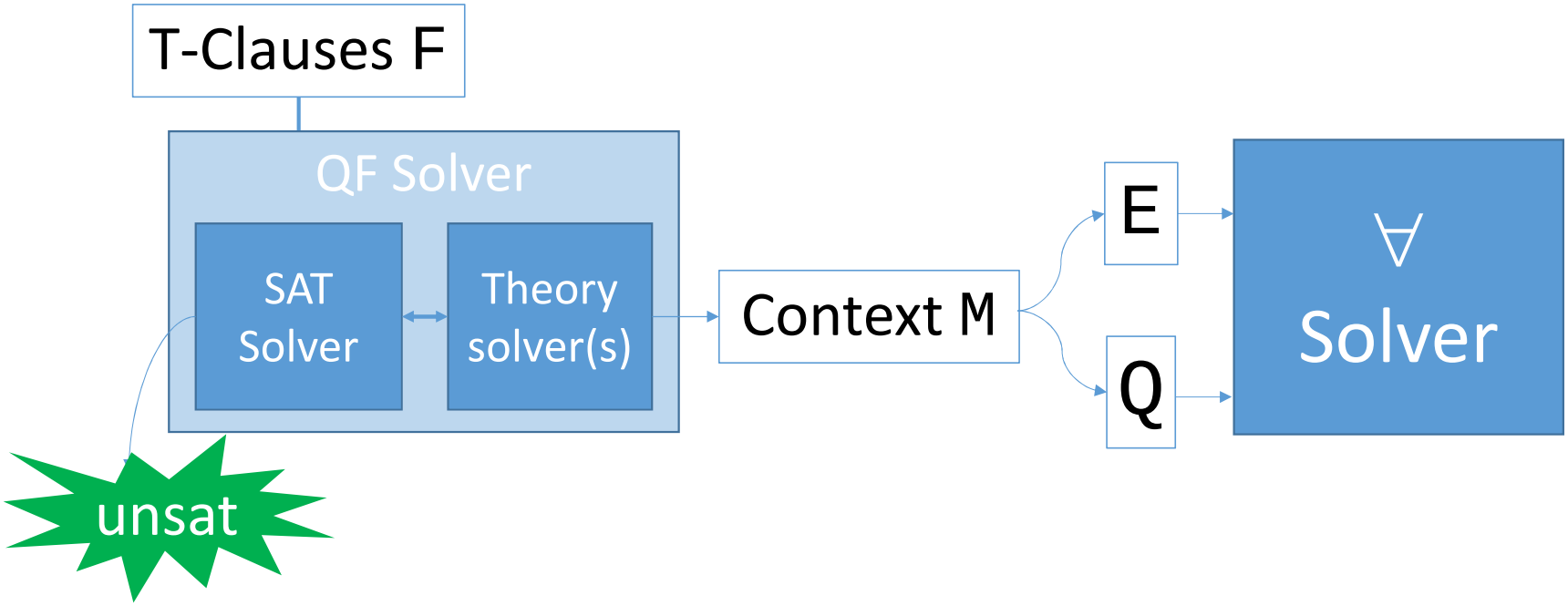
Set of ground equalities and disequalities

- $\{f(a)=b, P(a)=\bot, \ldots\}$

Set of quantified formulas

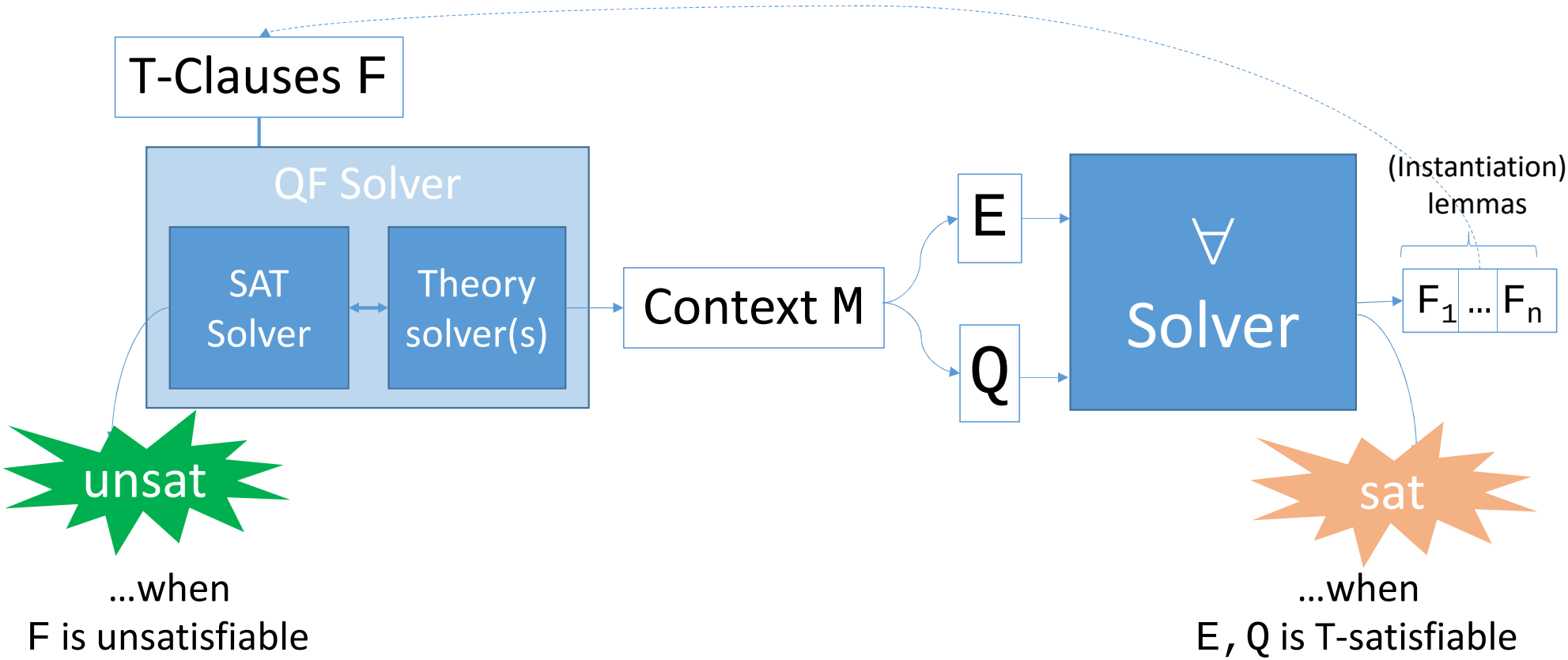
- $\{\forall x.P(x), \ldots\}$

DPLL(T)-Based SMT Solvers + ∀ Instantiation

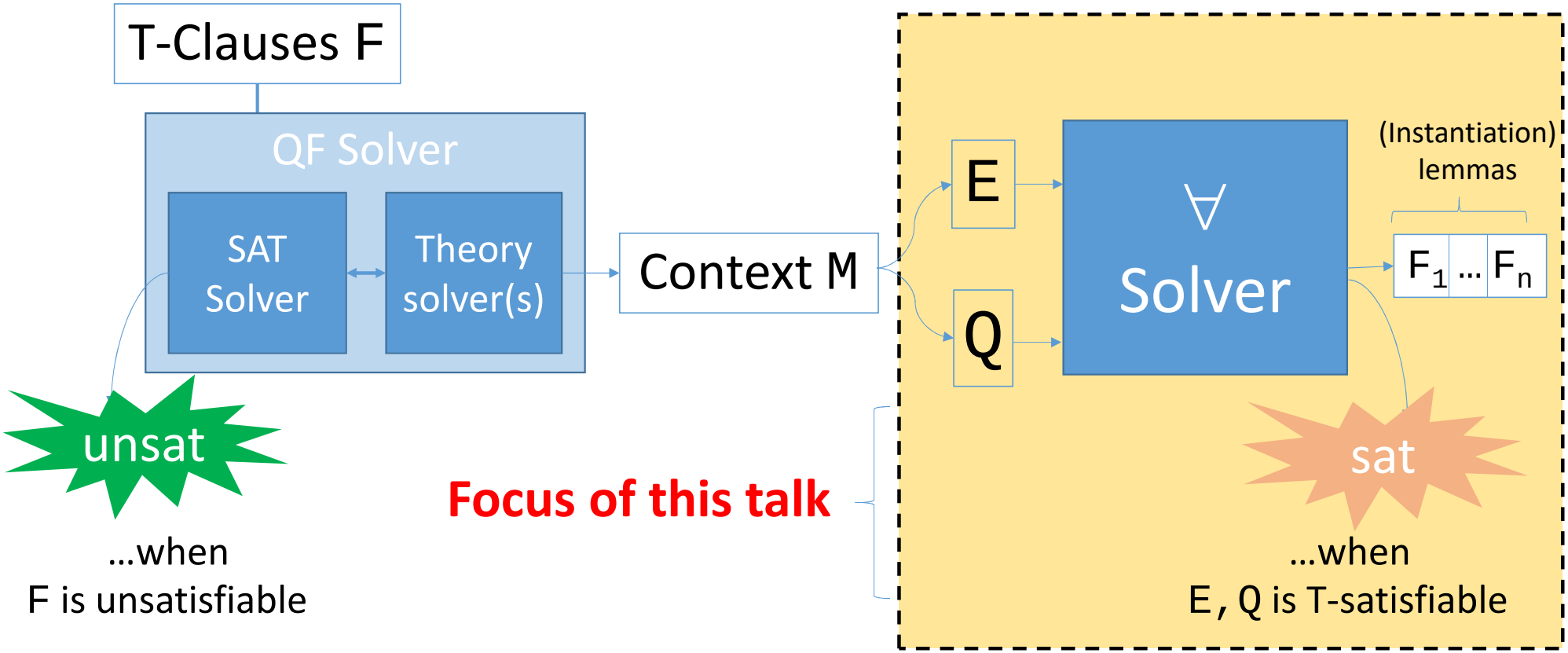


…when
F is unsatisfiable

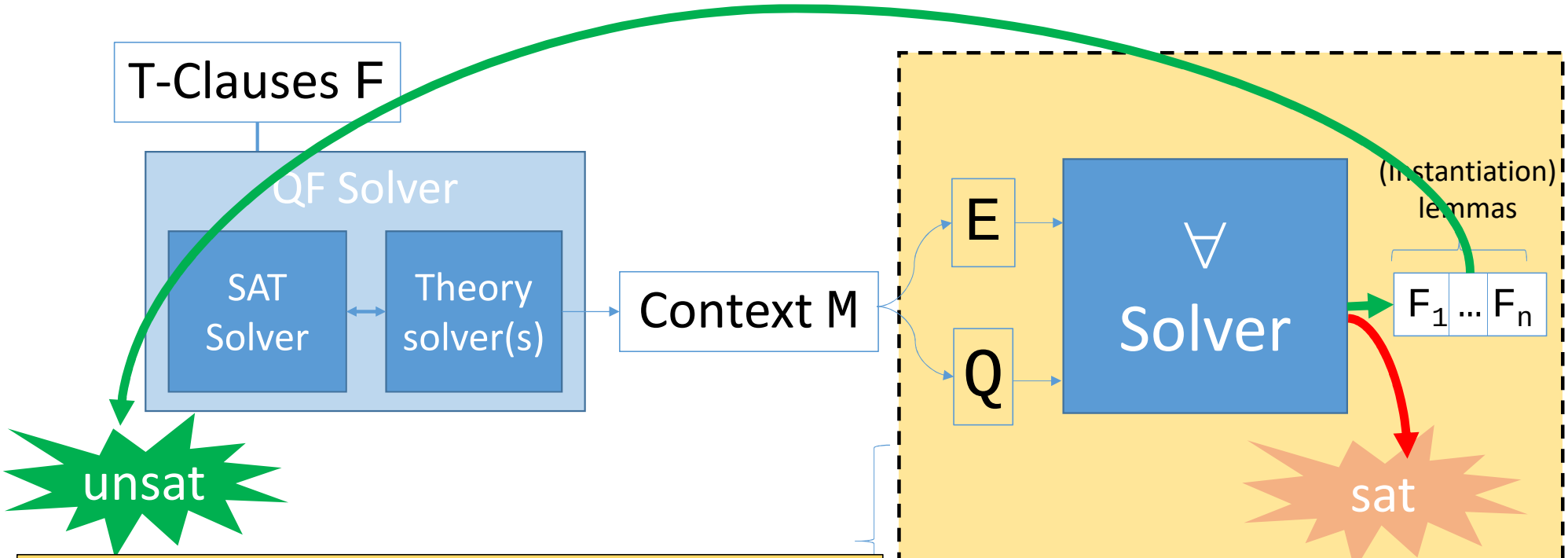
DPLL(T)-Based SMT Solvers + ∀ Instantiation



DPLL(T)-Based SMT Solvers + ∀ Instantiation

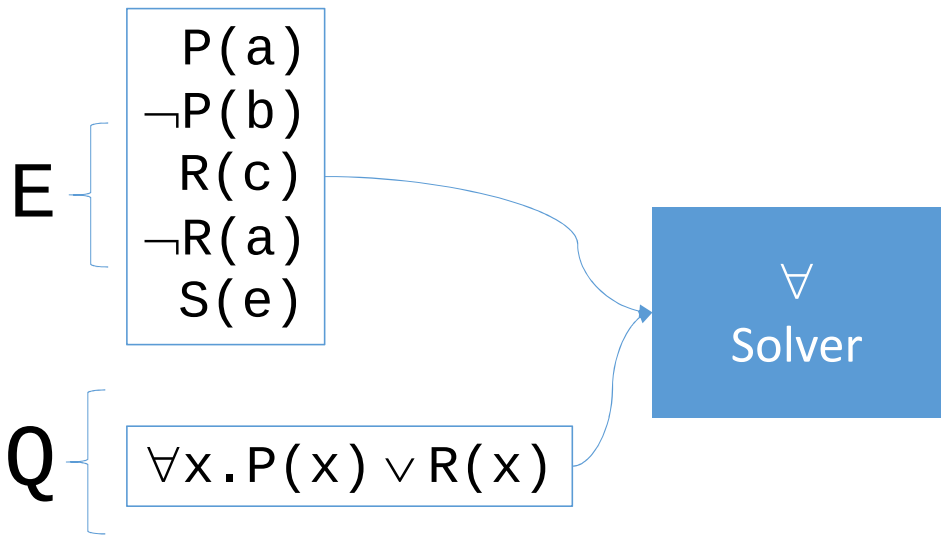


DPLL(T)-Based SMT Solvers + ∀ Instantiation

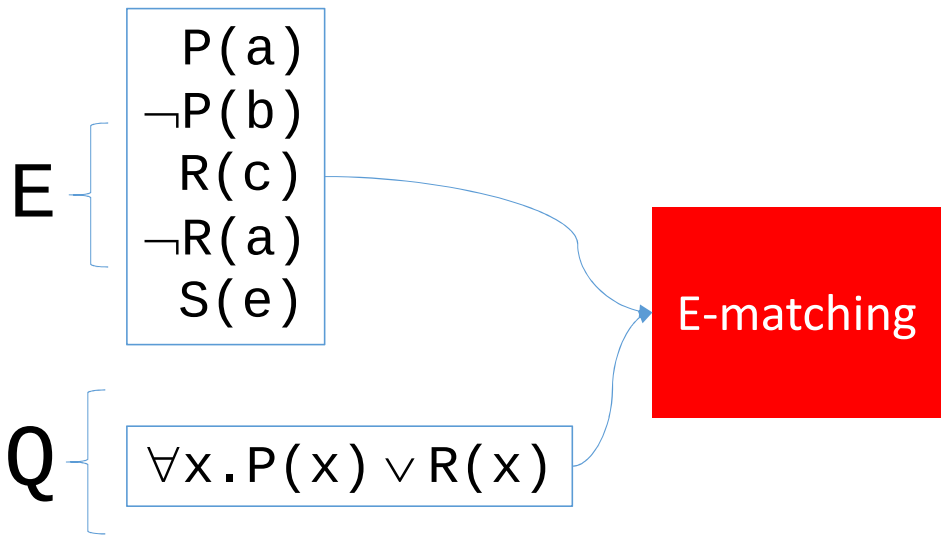


- Which lemmas are likely lead to "**unsat**"?
- When can we answer "**sat**"?

E-matching

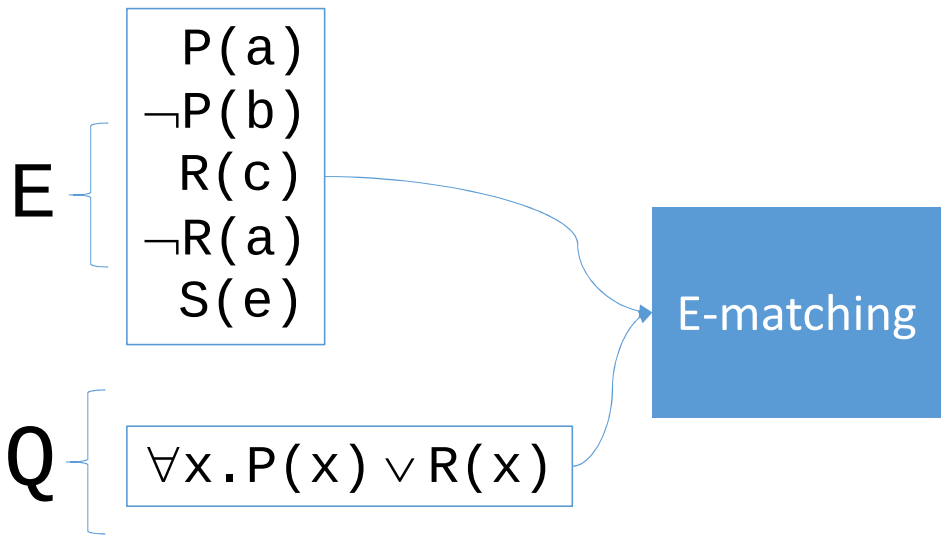


E-matching

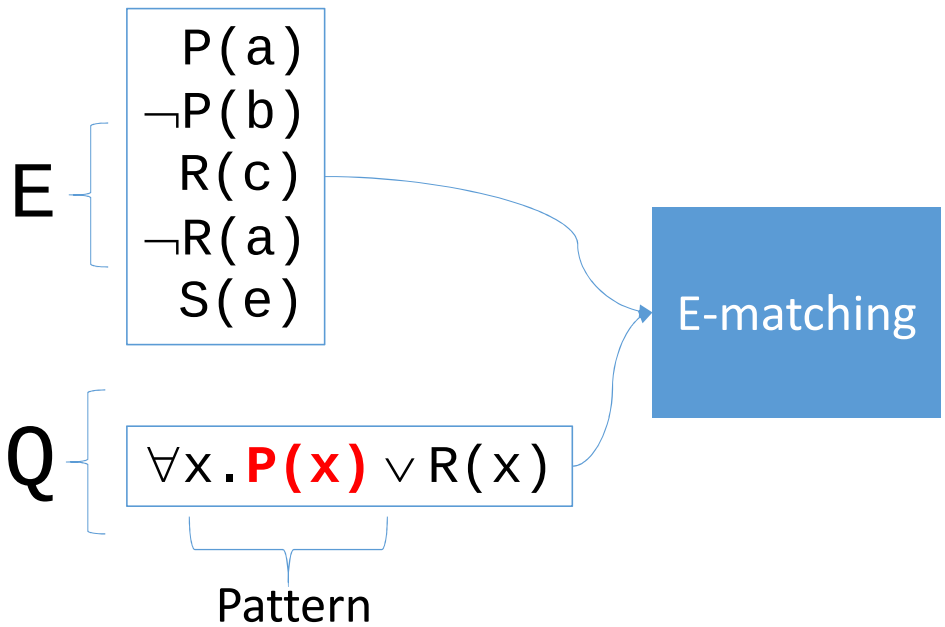


- Introduced in Nelson's Phd Thesis [Nelson 80]

E-matching

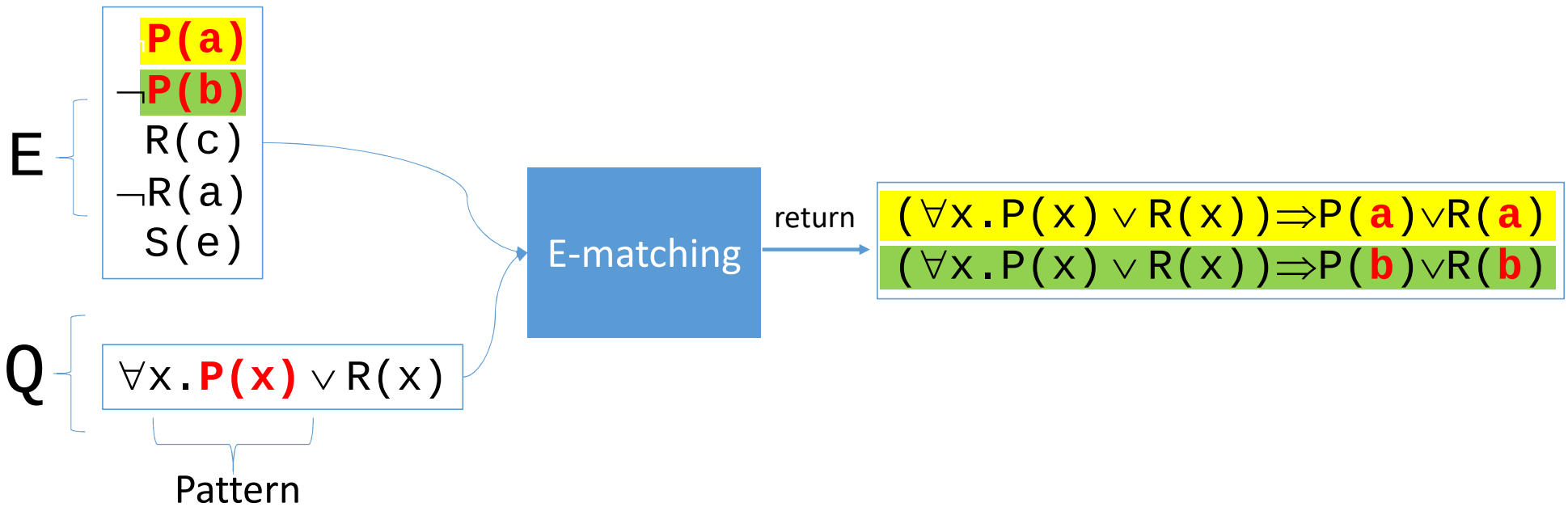


E-matching

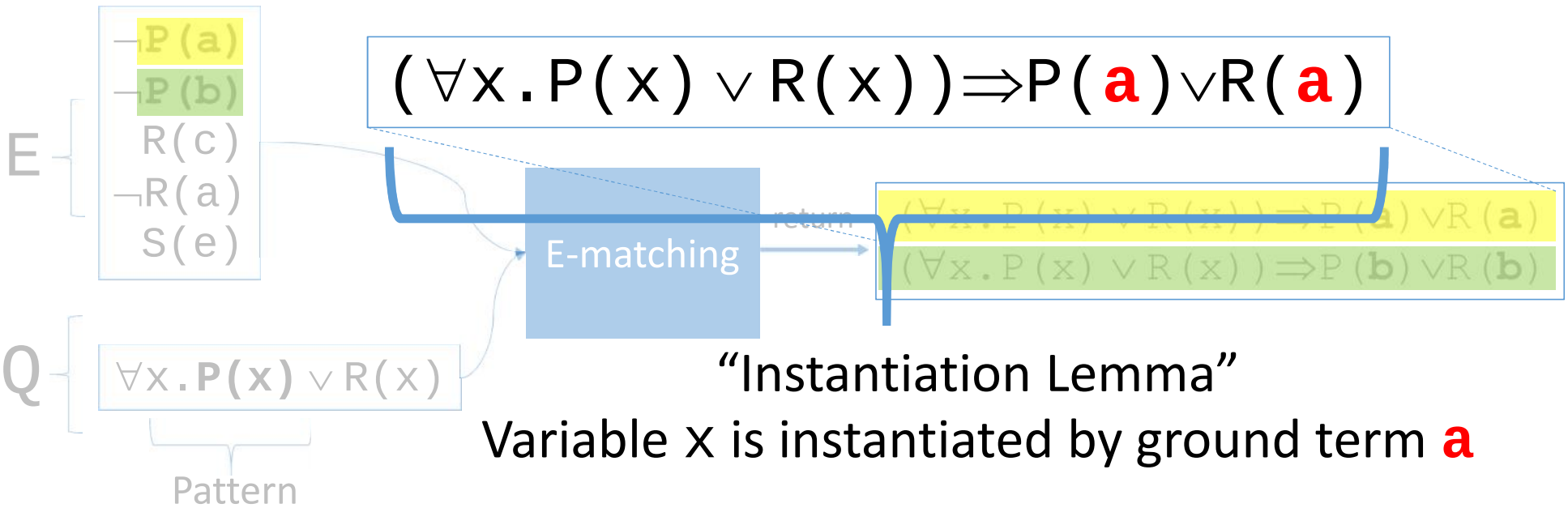


Ĝ **Idea**: choose instances based on pattern matching

E-matching


E
P(a)
¬P(b)
R(c)
¬R(a)
S(e)
Q
∀x.P(x) ∨ R(x)
Pattern
E-matching
return
(∀x.P(x) ∨ R(x))⇒P(a)∨R(a)
(∀x.P(x) ∨ R(x))⇒P(b)∨R(b)

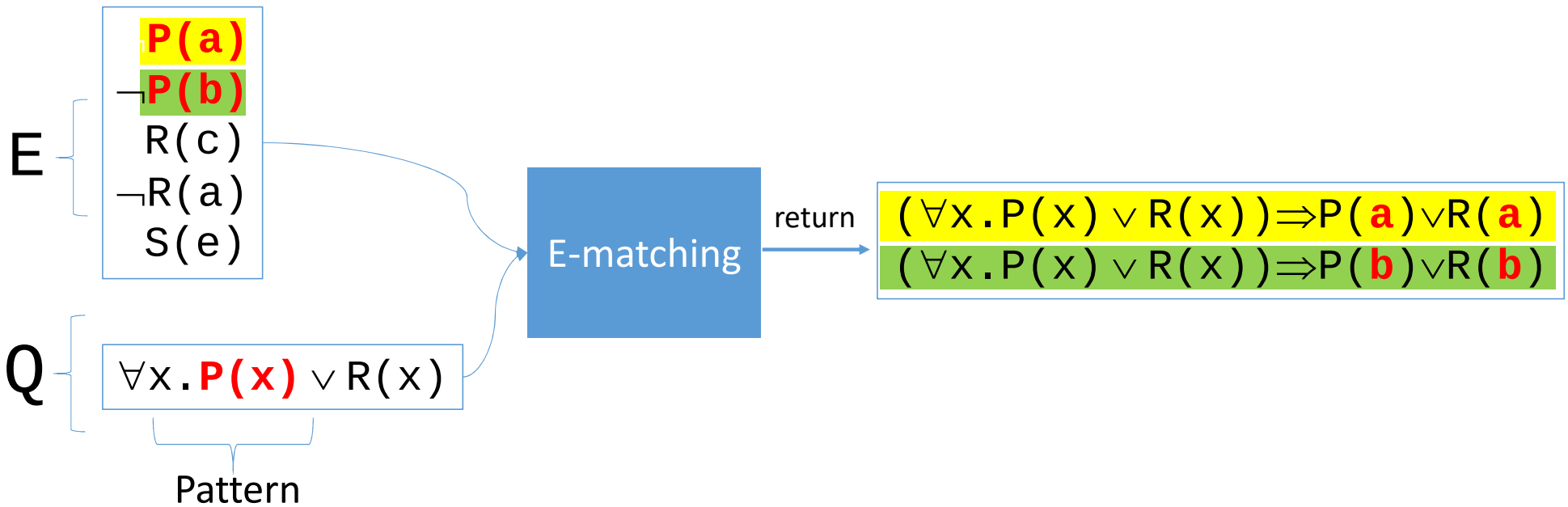
E-matching



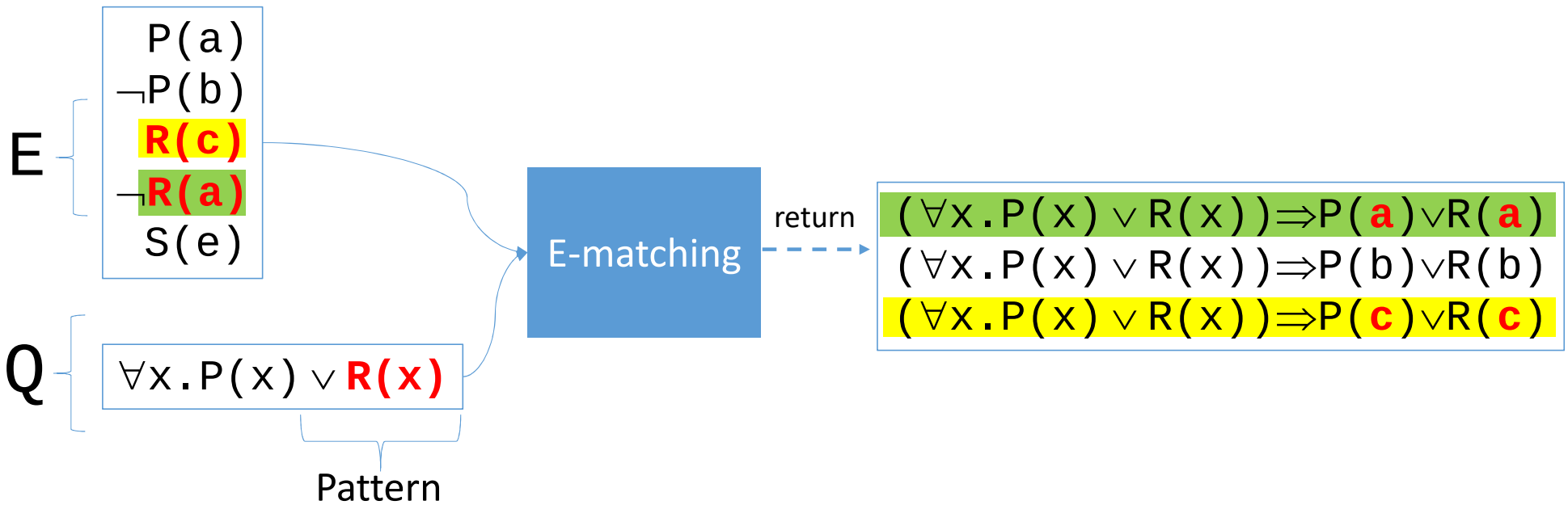
"Instantiation Lemma"

Variable x is instantiated by ground term **a**

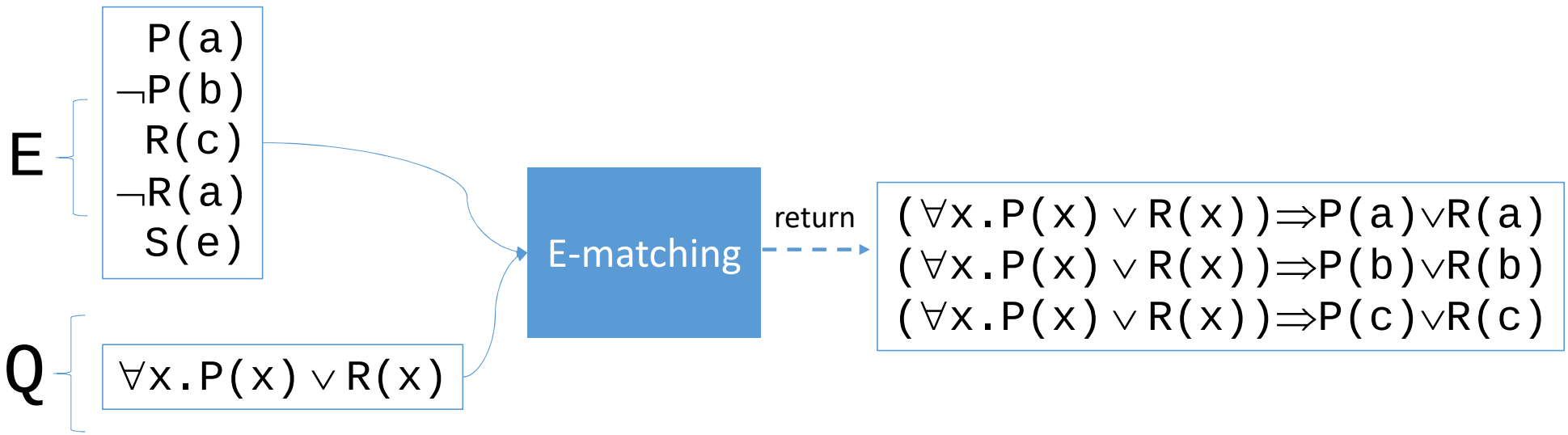
E-matching


E
P(a)
¬P(b)
R(c)
¬R(a)
S(e)
Q
∀x.P(x) ∨ R(x)
Pattern
E-matching
return
(∀x.P(x) ∨ R(x))⇒P(a)∨R(a)
(∀x.P(x) ∨ R(x))⇒P(b)∨R(b)

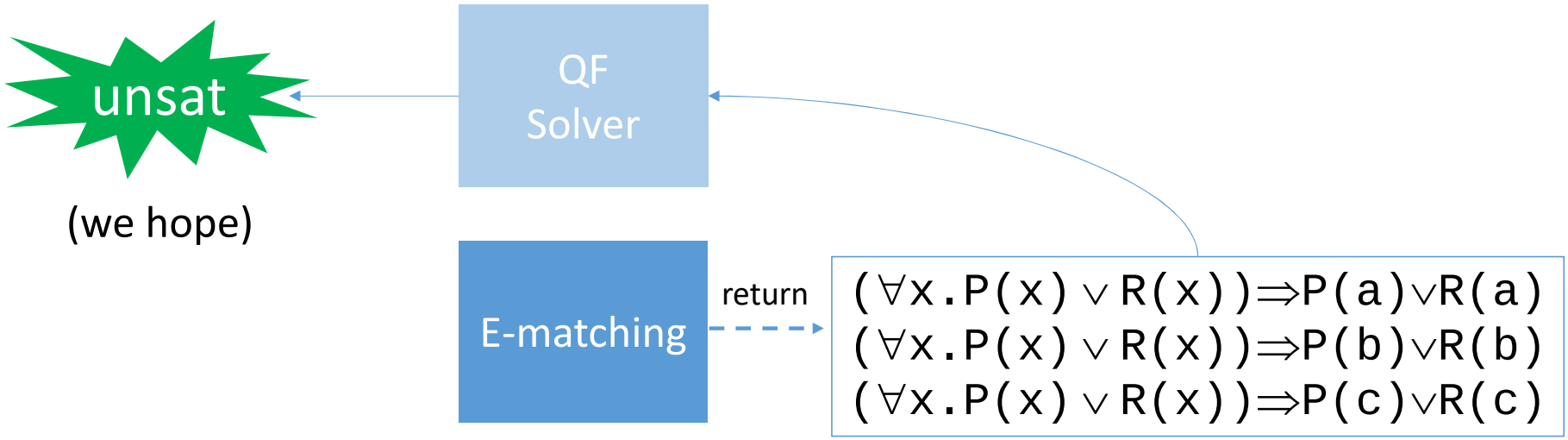
E-matching


E
P(a)
¬P(b)
R(c)
¬R(a)
S(e)
Q
∀x.P(x) ∨ R(x)
Pattern
E-matching
return
(∀x.P(x) ∨ R(x))⇒P(a)∨R(a)
(∀x.P(x) ∨ R(x))⇒P(b)∨R(b)
(∀x.P(x) ∨ R(x))⇒P(c)∨R(c)

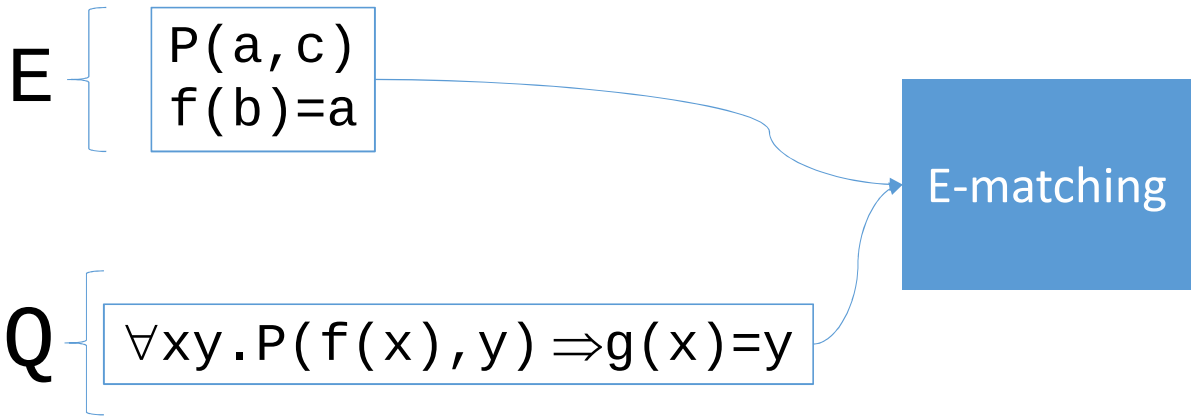
E-matching


E
P(a)
¬P(b)
R(c)
¬R(a)
S(e)
Q
∀x.P(x) ∨ R(x)
E-matching
return
(∀x.P(x) ∨ R(x))⇒P(a)∨R(a)
(∀x.P(x) ∨ R(x))⇒P(b)∨R(b)
(∀x.P(x) ∨ R(x))⇒P(c)∨R(c)

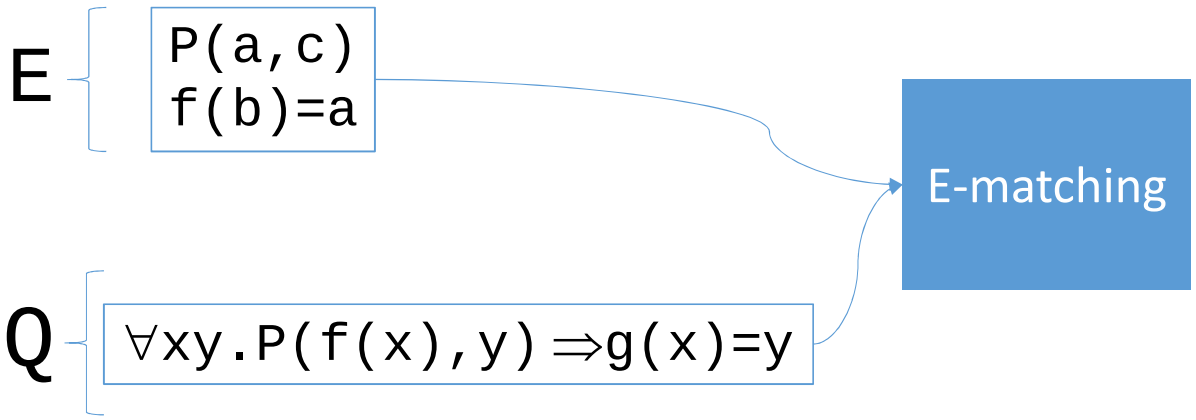
E-matching


unsat
(we hope)
QF
Solver
E-matching
return
(∀x.P(x) ∨ R(x))⇒P(a)∨R(a)
(∀x.P(x) ∨ R(x))⇒P(b)∨R(b)
(∀x.P(x) ∨ R(x))⇒P(c)∨R(c)

E-matching: Functions, Equality

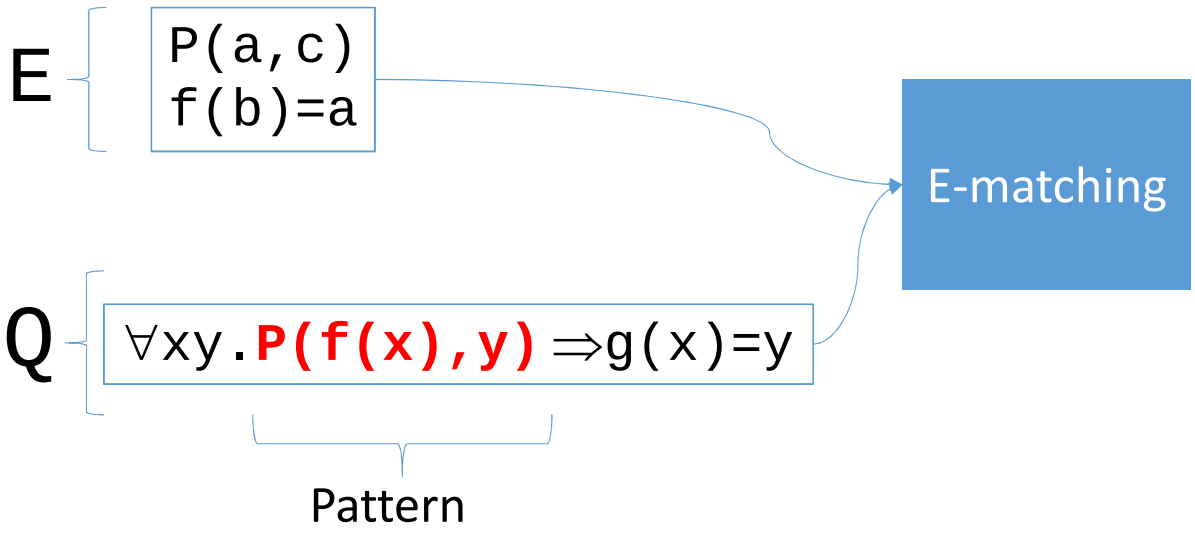


E-matching: Functions, Equality

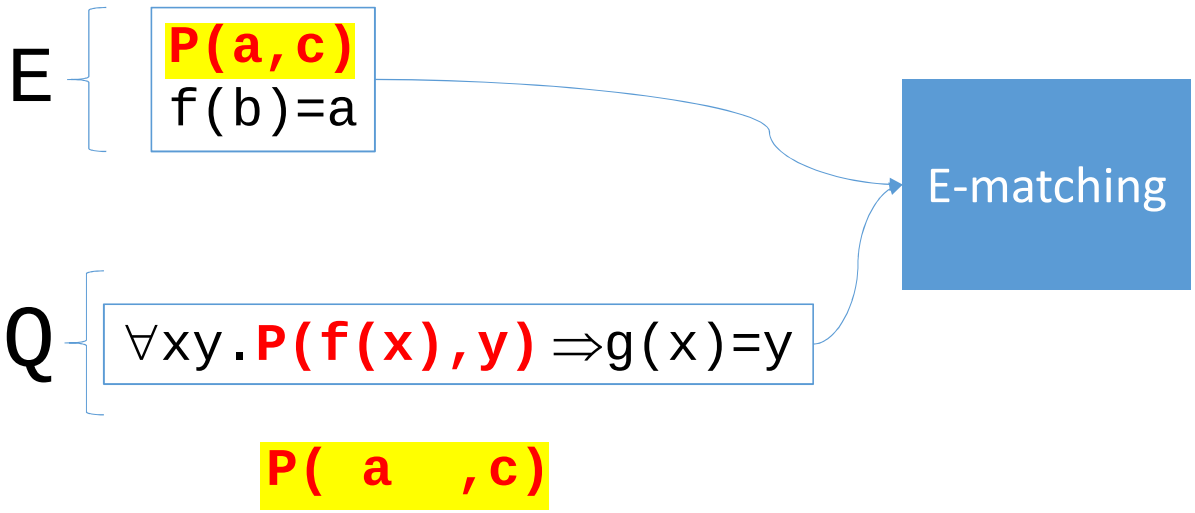


⇒ In E-matching, Pattern ***matching*** takes into account equalities in ***E***

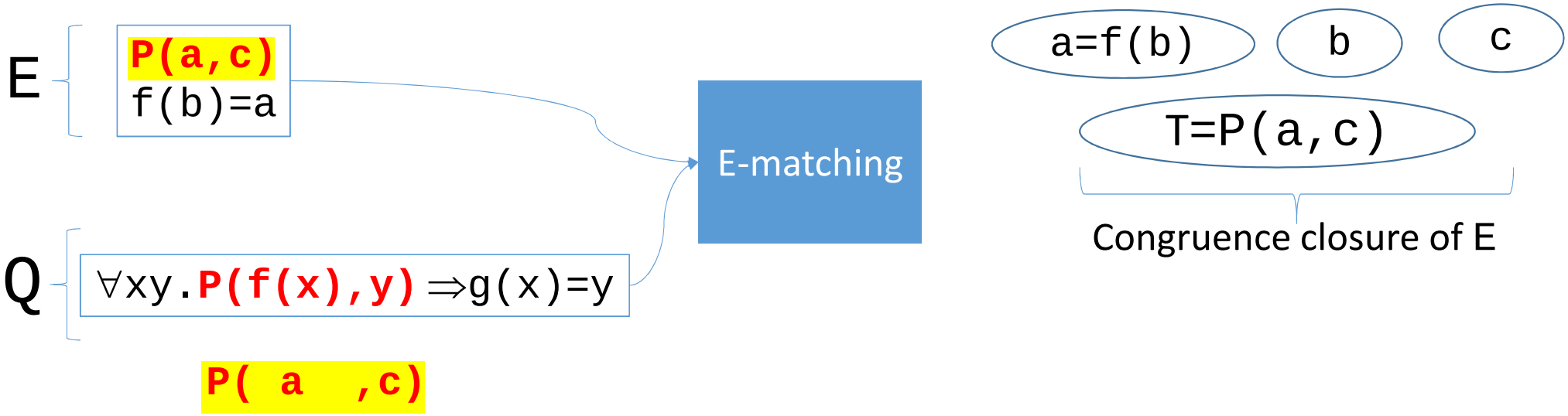
E-matching: Functions, Equality



E-matching: Functions, Equality



E-matching: Functions, Equality



E-matching: Functions, Equality

E

$P(a,c)$
$f(b)=a$

Q

$\forall xy.P(f(x),y) \Rightarrow g(x)=y$

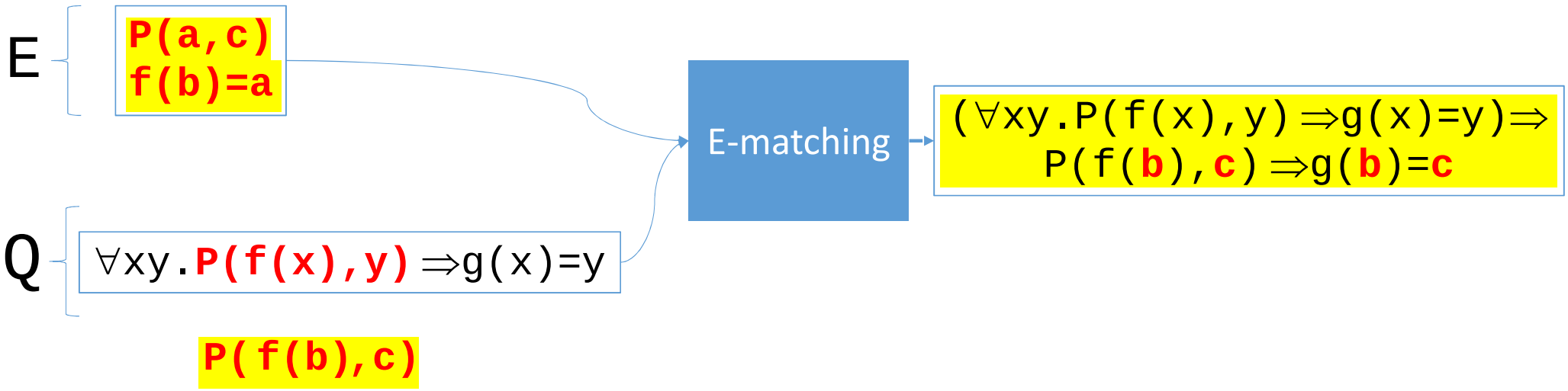
E-matching

$a=f(b)$ b c

$T=P(a,c)$

$P(f(b),c)$ ...E implies $P(a,c) \Leftrightarrow P(f(b),c)$

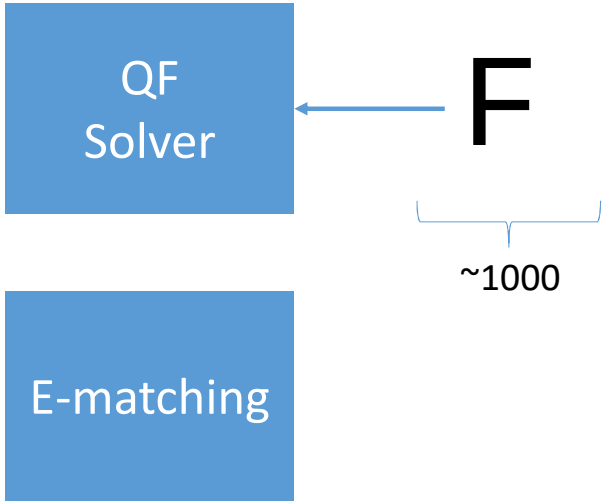
E-matching: Functions, Equality



E-matching

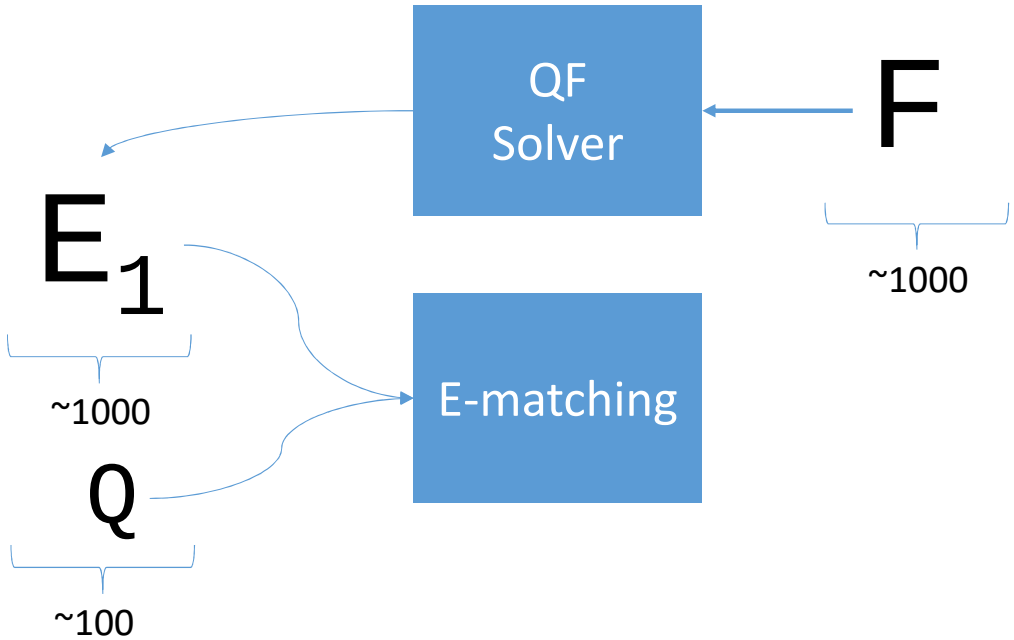
- Most widely used technique for unsatisfiable $\forall$ problems in SMT
  - Variants implemented in:
    - Z3 [deMoura et al 07], CVC3 [Ge et al 07], CVC4, Princess [Ruemmer 12], VeriT, Alt-Ergo
  - Used in:
    - Software verification
      - Boogie, Dafny, Leon, SPARK, Why3
    - Automated Theorem Proving
      - Sledgehammer

Challenge #1 : Too Many Instances



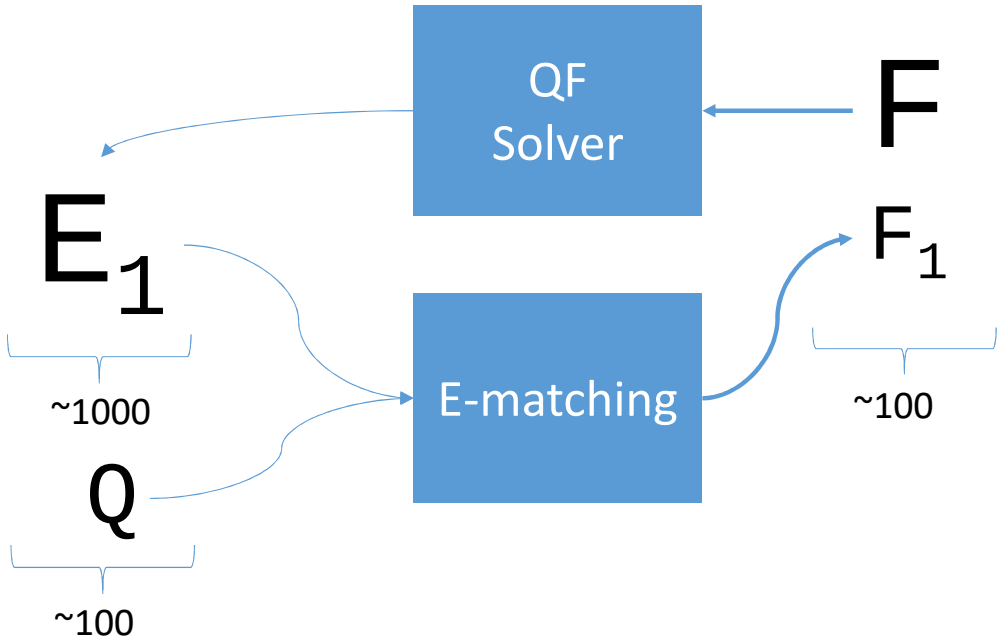
- Typical problems in applications:
  - F contains 1000s of clauses

Challenge #1 : Too Many Instances



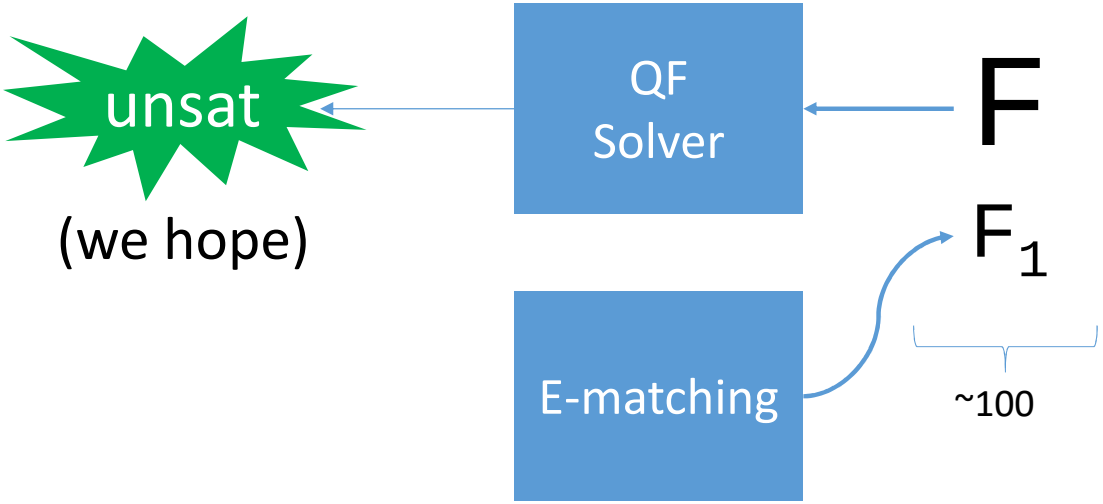
- Typical problems in applications:
  - F contains 1000s of clauses
  - Contexts contain 1000s of terms in E, 100s of ∀ in Q

Challenge #1 : Too Many Instances



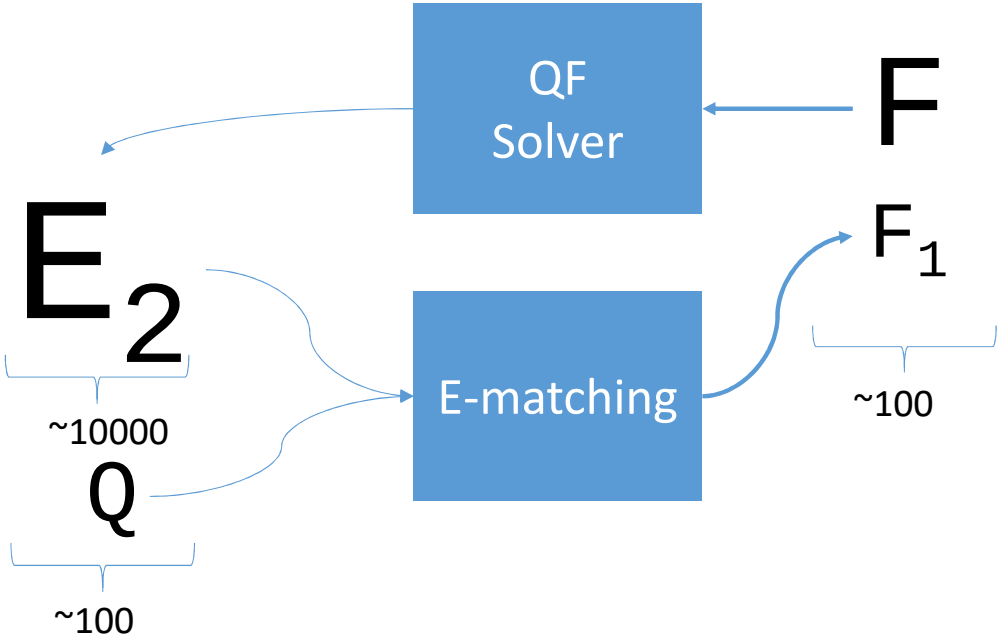
- Typical problems in applications:
  - F contains 1000s of clauses
  - Contexts contain 1000s of terms in E, 100s of ∀ in Q

Challenge #1 : Too Many Instances



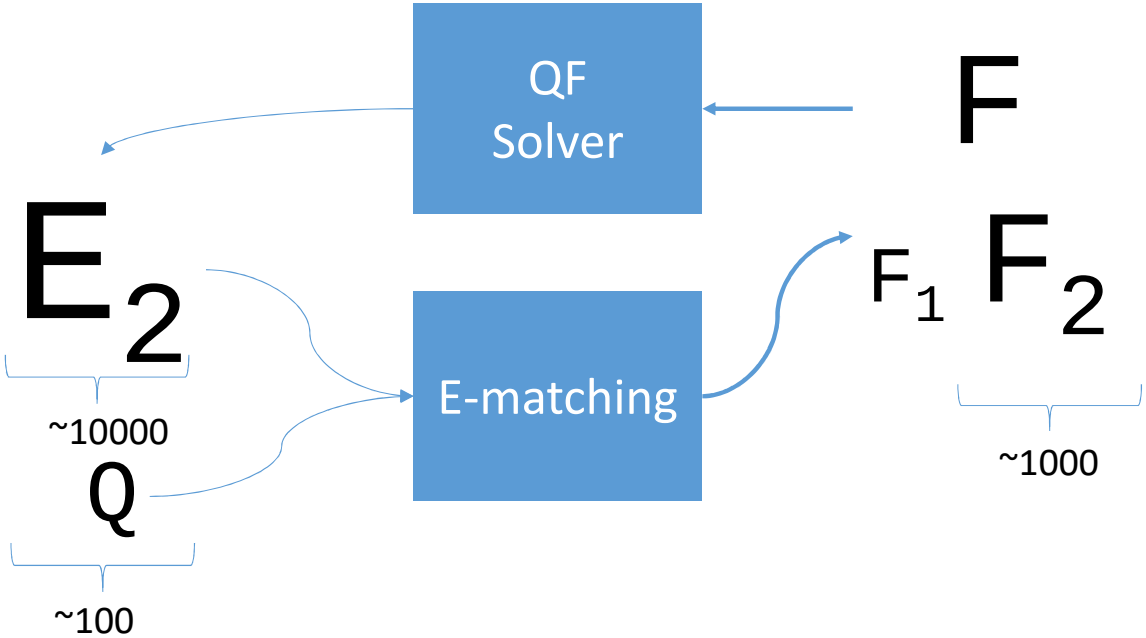
- Typical problems in applications:
  - F contains 1000s of clauses
  - Contexts contain 1000s of terms in E, 100s of ∀ in Q

Challenge #1 : Too Many Instances



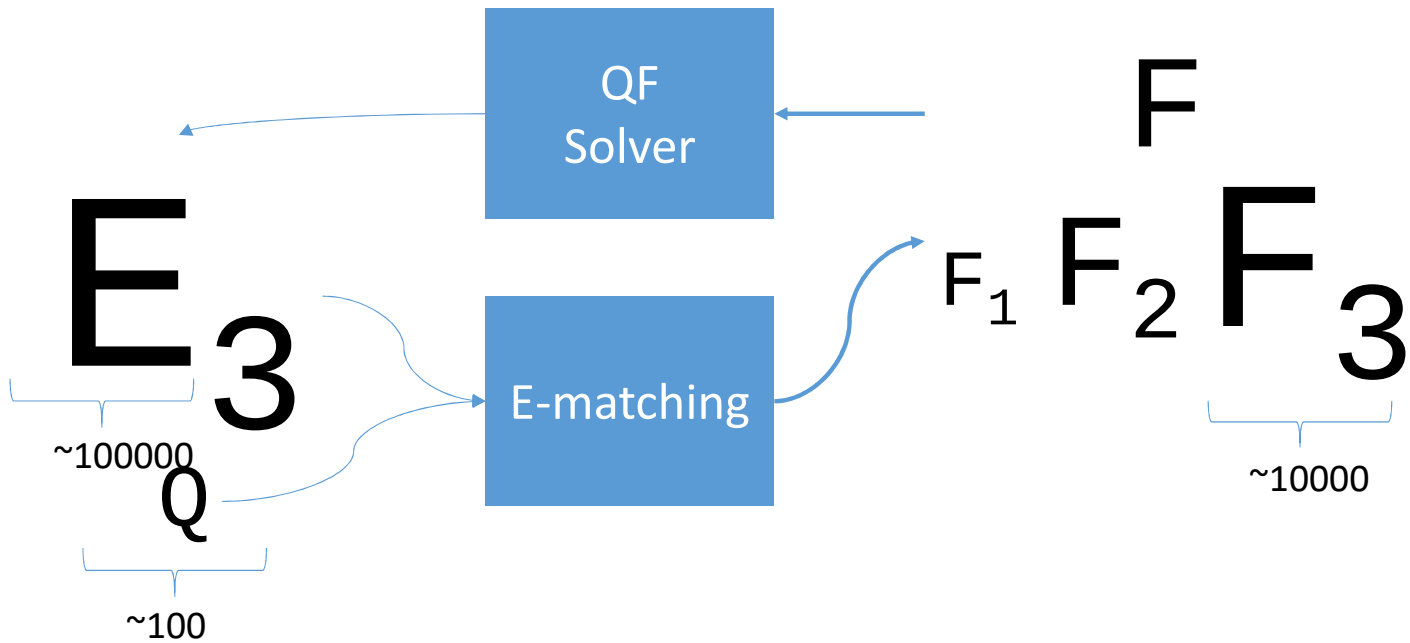
- Typical problems in applications:
  - F contains 1000s of clauses
  - Contexts contain 1000s of terms in E, 100s of ∀ in Q
  - Leads to 100s

Challenge #1 : Too Many Instances



- Typical problems in applications:
  - F contains 1000s of clauses
  - Contexts contain 1000s of terms in E, 100s of ∀ in Q
  - Leads to 100s, 1000s

Challenge #1 : Too Many Instances



- Typical problems in applications:
  - F contains 1000s of clauses
  - Contexts contain 1000s of terms in E, 100s of ∀ in Q
  - Leads to 100s, 1000s, 10000s of instances

Challenge #1 : Too Many Instances

$\Rightarrow$ QF solver is overloaded …solver times out

Challenge #1 : Too Many Instances

# Instances	cvc3		cvc4		z3	
	#	%	#	%	#	%
1-10	1464	13.49%	1007	8.87%	1321	11.43%
10-100	1755	16.17%	1853	16.31%	2554	22.11%
100-1000	**3816**	35.16%	**3680**	32.40%	**4553**	39.41%
1000-10k	1893	17.44%	2468	21.73%	1779	15.40%
10k-100k	1162	10.71%	1414	12.45%	823	7.12%
100k-1M	560	5.16%	607	5.34%	376	3.25%
1M-10M	193	1.78%	330	2.91%	139	1.20%
>10M	10	0.09%	0	0.00%	8	0.07%

(for 8 of benchmarks z3 solves, its E-matching procedure adds more than 10M instances)

- Evaluation on 33032 SMTLIB, TPTP, Isabelle benchmarks
  - E-matching often requires **many instances**

    (Above, 16.6% required >10k, max 19.5M by z3 on a software verification benchmark from TPTP)

Challenge #2 : Incompleteness

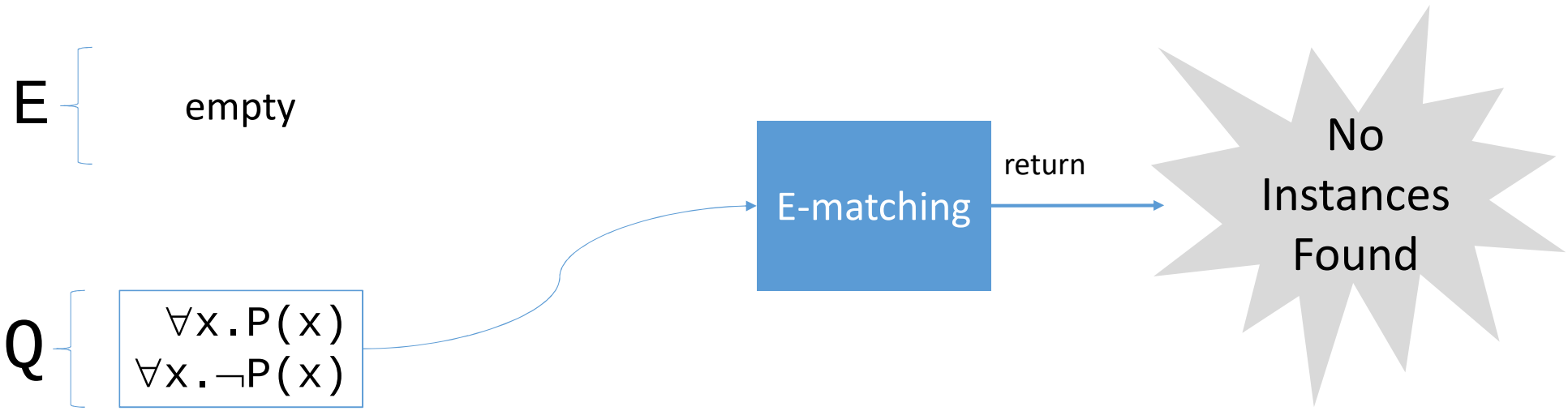
E { empty

Q { $\forall x.P(x)$
$\forall x.\neg P(x)$

E-matching

$\Rightarrow$ E-matching is an incomplete procedure

Challenge #2 : Incompleteness

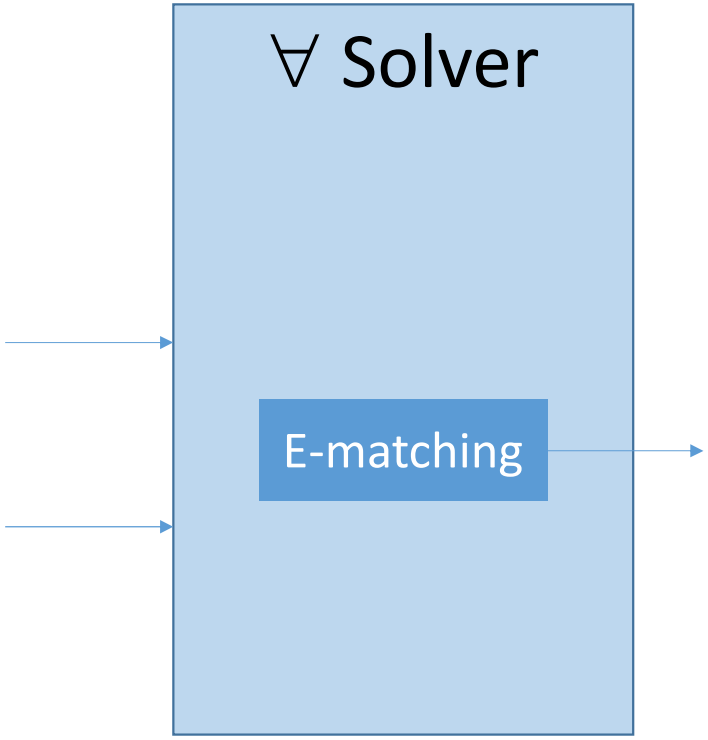


$\Rightarrow$ If E-matching produces no instances,
this *does not guarantee EĆQ is T-satisfiable*

E-matching : Challenges Addressed

- What if there are **too many instances**?

- What if there are **no instances**, and problem maybe "**sat**"?

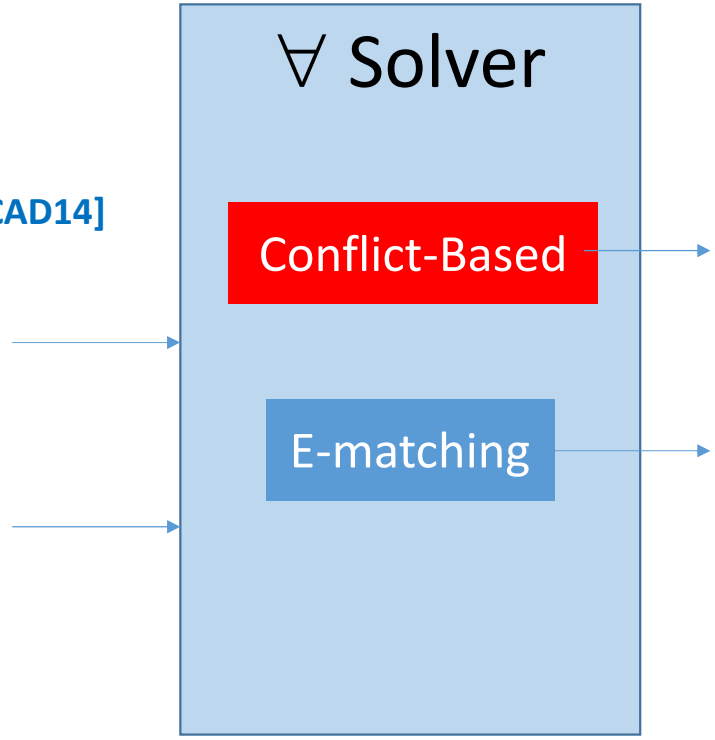


E-matching : Challenges Addressed

- What if there are **too many instances**?

  ⇒*Use conflict-based instantiation* **[Reynolds et al FMCAD14]**

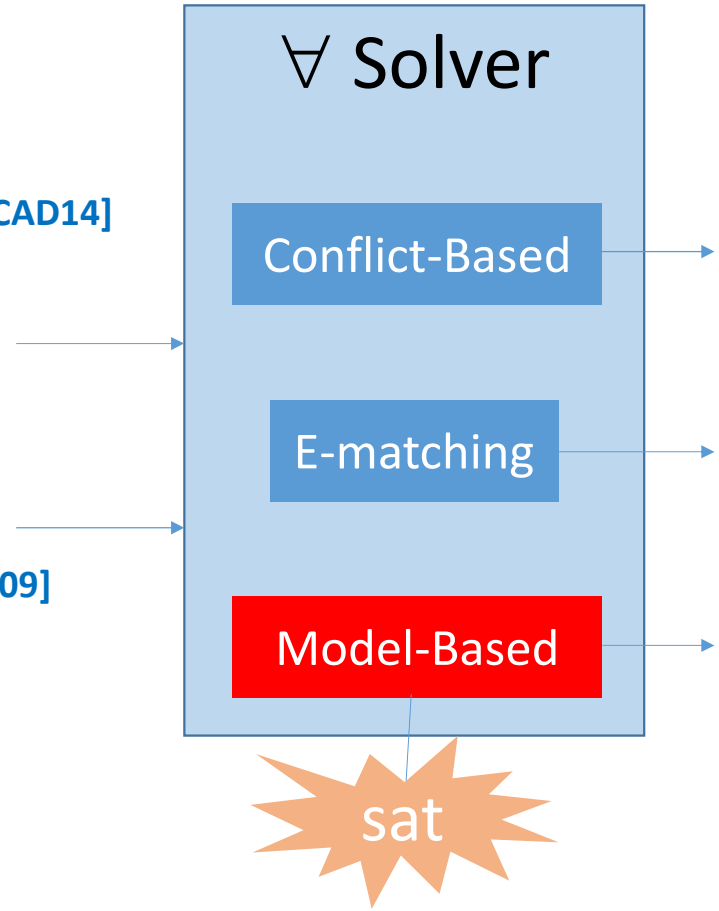
- What if there are no instances, and problem maybe "sat"?



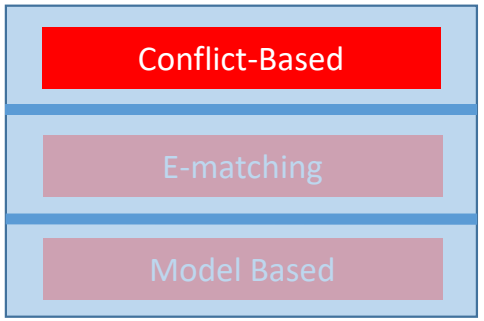
E-matching : Challenges Addressed

- What if there are too many instances?
  ⇒Use conflict-based instantiation **[Reynolds et al FMCAD14]**

- What if there are **no instances**, and problem maybe "**sat**"?
  ⇒*Use model-based instantiation* **[Ge/deMoura CAV2009]**



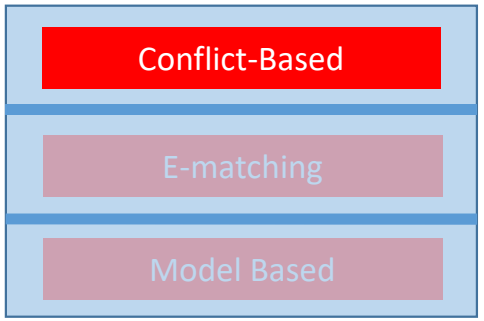
Conflict-Based Instantiation

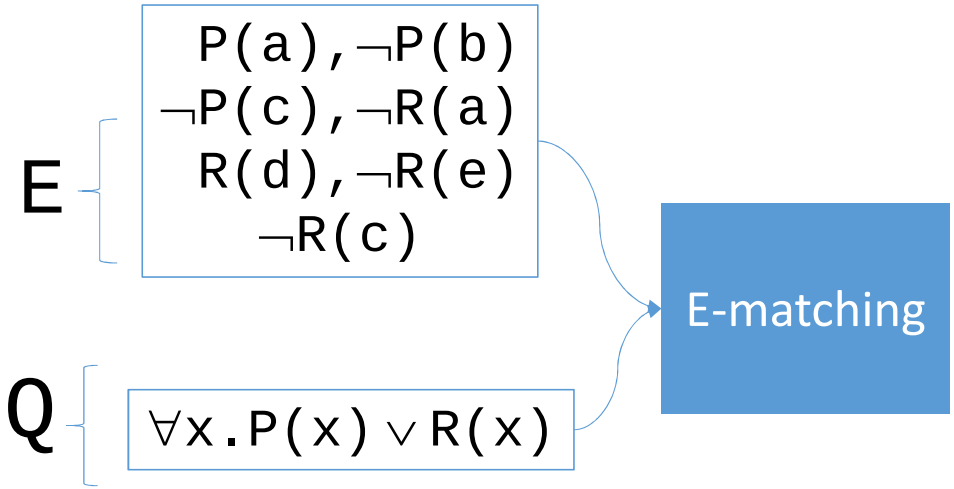


- Basic idea:
  - Since we are interested in whether e.g. $E, \forall x . P(x)$ is satisfiable,
    - Try to find one "conflict instance" such that $E, P(a) \models \bot$
    - If this is possible, don't run E-matching

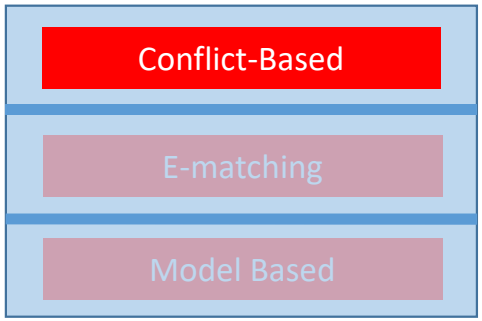
Ĝ *Leads to fewer instances, improved ability to answer* unsat

Conflict-Based Instantiation





Conflict-Based Instantiation



E

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

Q

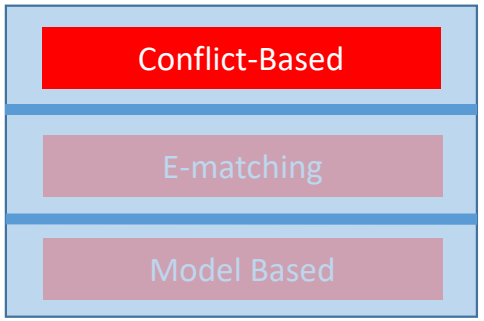
$\forall x.P(x) \vee R(x)$

E-matching

$(\forall x.P(x) \vee R(x)) \Rightarrow P(\mathbf{a}) \vee R(\mathbf{a})$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(\mathbf{b}) \vee R(\mathbf{b})$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(\mathbf{c}) \vee R(\mathbf{c})$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(\mathbf{d}) \vee R(\mathbf{d})$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(\mathbf{e}) \vee R(\mathbf{e})$

$\Rightarrow$ E-matching would produce $\{x \rightarrow \mathbf{a}\}$, $\{x \rightarrow \mathbf{b}\}$, $\{x \rightarrow \mathbf{c}\}$, $\{x \rightarrow \mathbf{d}\}$, $\{x \rightarrow \mathbf{e}\}$

Conflict-Based Instantiation



E

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

Q

$\forall x. P(x) \vee R(x)$

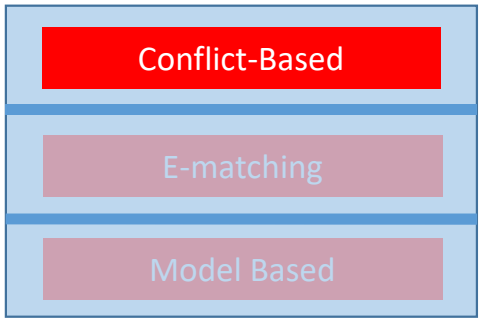
E-matching

$(\forall x. P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models P(a) \vee R(a)$
$E, P(b) \vee R(b) \models P(b) \vee R(b)$
$E, P(c) \vee R(c) \models P(c) \vee R(c)$
$E, P(d) \vee R(d) \models P(d) \vee R(d)$
$E, P(e) \vee R(e) \models P(e) \vee R(e)$

Conflict-Based Instantiation



E

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

Q

$$\forall x.P(x) \vee R(x)$$

E-matching

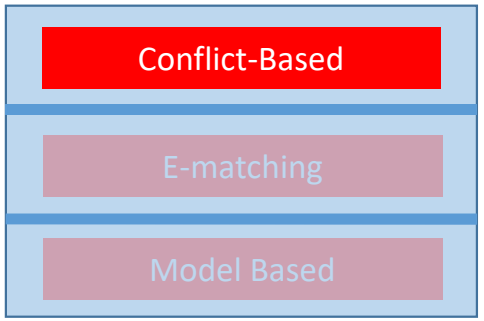
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$$

$\Rightarrow$ Consider what we learn from these instances:

$$E, P(a) \vee R(a) \models T \vee R(a)$$
$$E, P(b) \vee R(b) \models P(b) \vee R(b)$$
$$E, P(c) \vee R(c) \models P(c) \vee R(c)$$
$$E, P(d) \vee R(d) \models P(d) \vee R(d)$$
$$E, P(e) \vee R(e) \models P(e) \vee R(e)$$

By E, we know $P(a) \Leftrightarrow T$

Conflict-Based Instantiation



E

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

Q

$\forall x. P(x) \vee R(x)$

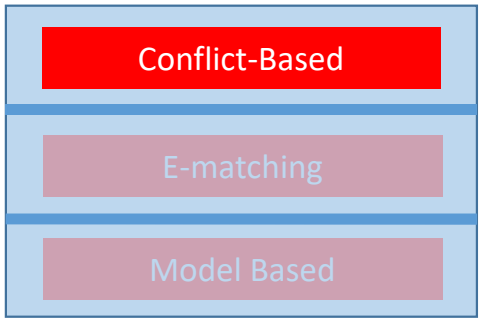
E-matching

$(\forall x. P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$$
\begin{aligned}
E, P(a) \vee R(a) &\models T \\
E, P(b) \vee R(b) &\models P(b) \vee R(b) \\
E, P(c) \vee R(c) &\models P(c) \vee R(c) \\
E, P(d) \vee R(d) &\models P(d) \vee R(d) \\
E, P(e) \vee R(e) &\models P(e) \vee R(e)
\end{aligned}
$$

Conflict-Based Instantiation



E

$P(a), \dot{E}P(b)$
$\neg P(c), \neg R(a)$
$R(d), \neg R(e)$
$\neg R(c)$

Q

$\forall x.P(x) \vee R(x)$

E-matching

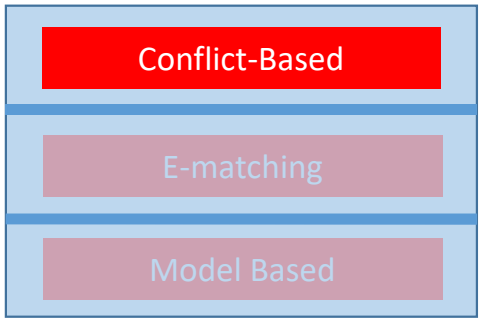
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$$
\begin{array}{lcl}
E, P(a) \vee R(a) & \models & \top \\
E, P(b) \vee R(b) & \models & \bot \vee R(b) \\
E, P(c) \vee R(c) & \models & P(c) \vee R(c) \\
E, P(d) \vee R(d) & \models & P(d) \vee R(d) \\
E, P(e) \vee R(e) & \models & P(e) \vee R(e)
\end{array}
$$

We know $P(b) \Leftrightarrow \{$

Conflict-Based Instantiation



E

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

Q

$\forall x. P(x) \vee R(x)$

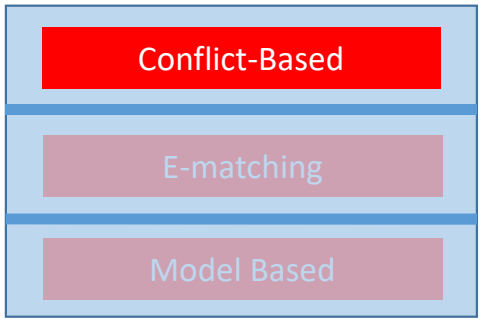
E-matching

$(\forall x. P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models T$
$E, P(b) \vee R(b) \models R(b)$
$E, P(c) \vee R(c) \models P(c) \vee R(c)$
$E, P(d) \vee R(d) \models P(d) \vee R(d)$
$E, P(e) \vee R(e) \models P(e) \vee R(e)$

Conflict-Based Instantiation



E

$P(a), \neg P(b)$
$\dot{E}\mathbf{P(c)}, \neg R(a)$
$R(d), \neg R(e)$
$\neg R(c)$

Q

$\forall x.P(x) \vee R(x)$

E-matching

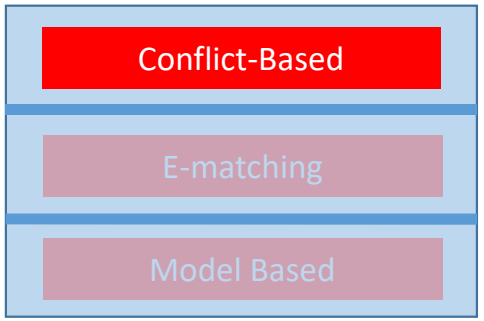
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models T$
$E, P(b) \vee R(b) \models R(b)$
$E, P(c) \vee R(c) \models R(c)$
$E, P(d) \vee R(d) \models P(d) \vee R(d)$
$E, P(e) \vee R(e) \models P(e) \vee R(e)$

We know $\mathbf{P(c)} \Leftrightarrow \{$

Conflict-Based Instantiation



E

$P(a), \neg P(b)$
$\neg P(c), \neg R(a)$
$\mathbf{R(d)}, \neg R(e)$
$\neg R(c)$

Q

$\forall x. P(x) \vee R(x)$

E-matching

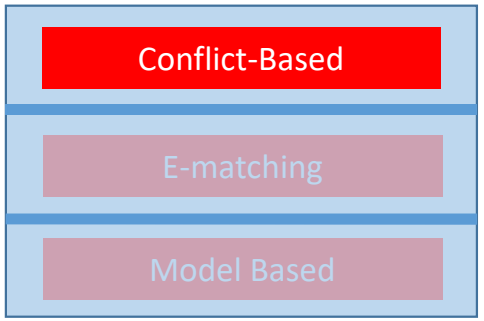
$(\forall x. P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models T$
$E, P(b) \vee R(b) \models R(b)$
$E, P(c) \vee R(c) \models R(c)$
$E, P(d) \vee R(d) \models T$
$E, P(e) \vee R(e) \models P(e) \vee R(e)$

We know $\mathbf{R(d)} \Leftrightarrow \mathbf{T}$

Conflict-Based Instantiation



E

$P(a), \neg P(b)$
$\neg P(c), \neg R(a)$
$R(d), \dot{E}R(e)$
$\neg R(c)$

Q

$\forall x.P(x) \vee R(x)$

E-matching

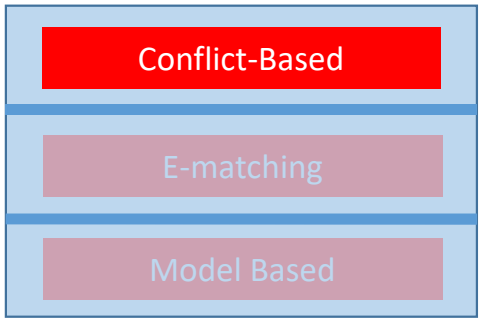
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models T$
$E, P(b) \vee R(b) \models R(b)$
$E, P(c) \vee R(c) \models R(c)$
$E, P(d) \vee R(d) \models T$
$E, P(e) \vee R(e) \models P(e)$

We know $R(e) \Leftrightarrow$ {

Conflict-Based Instantiation



E

$P(a), \neg P(b)$
$\neg P(c), \neg R(a)$
$R(d), \neg R(e)$
$\dot{E}R(c)$

Q

$\forall x.P(x) \vee R(x)$

E-matching

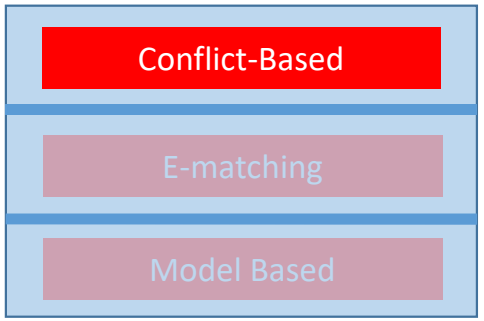
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models T$
$E, P(b) \vee R(b) \models R(b)$
$E, P(c) \vee R(c) \models \{$
$E, P(d) \vee R(d) \models T$
$E, P(e) \vee R(e) \models P(e)$

We know $R(c) \Leftrightarrow \{$

Conflict-Based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

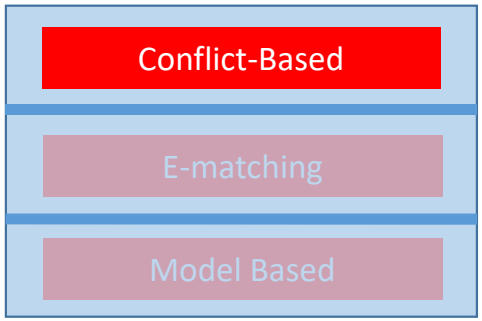
E-matching

$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models T$
$E, P(b) \vee R(b) \models R(b)$
$E, P(c) \vee R(c) \models \{$
$E, P(d) \vee R(d) \models T$
$E, P(e) \vee R(e) \models P(e)$

Conflict-Based Instantiation



E

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

Q

$\forall x. P(x) \vee R(x)$

E-matching

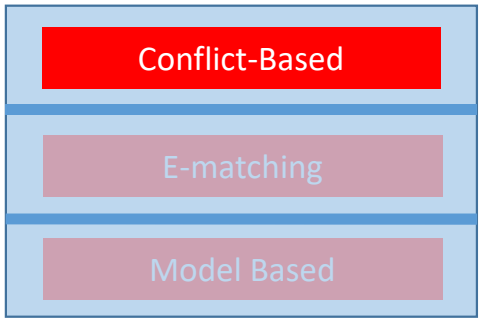
$(\forall x. P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ Consider what we learn from these instances:

$E, P(a) \vee R(a) \models T$
$E, P(b) \vee R(b) \models R(b)$
$\mathbf{E, P(c) \not\models R(c) \models \{}$
$E, P(d) \vee R(d) \models T$
$E, P(e) \vee R(e) \models P(e)$

*$P(c) \not\models R(c)$ is a **conflicting instance** for (E, Q)!*

Conflict-Based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

Conflict-based Instantiation

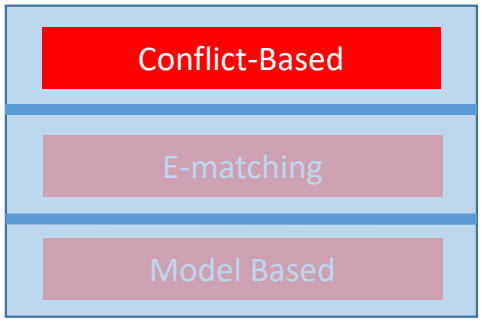
$\cancel{(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)}$
$\cancel{(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)}$
$(?x.P(x) \text{Ę} R(x)) \hat{\text{G}} P(c) \text{Ę} R(c)$
$\cancel{(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)}$
$\cancel{(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)}$

$\Rightarrow$ Consider what we learn from these instances:

$$
\begin{array}{lcl}
E, P(a) \vee R(a) & \models & T \\
E, P(b) \vee R(b) & \models & R(b) \\
E, P(c) \vee R(c) & \models & \{ \\
E, P(d) \vee R(d) & \models & T \\
E, P(e) \vee R(e) & \models & P(e)
\end{array}
$$

Since $P(c) \vee R(c)$ suffices to derive $\{$, return ***only*** this instance

Conflict-Based Instantiation: EUF



E

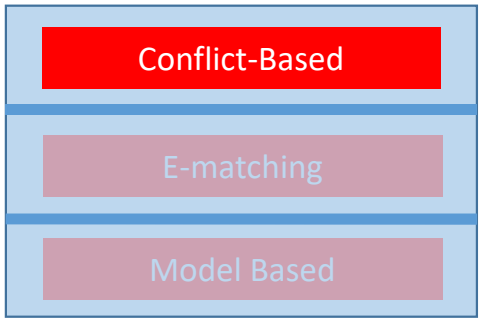
$a \neq c, f(b)=b,$
$g(b)=a, f(a)=a,$
$h(f(a))=d, h(b)=c$

Q

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

CBQI

Conflict-Based Instantiation: EUF



E

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

Q

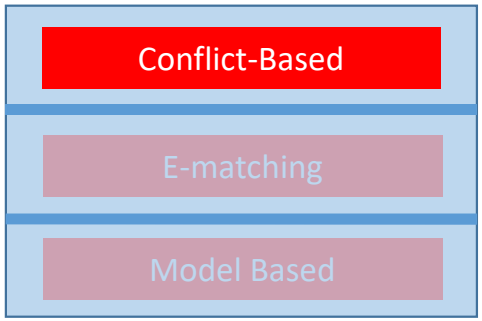
$\forall x. f(g(x))=h(f(x))$

CBQI

$\Rightarrow$ Consider the instance $\forall x. f(g(x))=h(f(x)) \Rightarrow f(g(\mathbf{b}))=h(f(\mathbf{b}))$

- Is this conflicting for (E, Q)?

Conflict-Based Instantiation: EUF



E

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

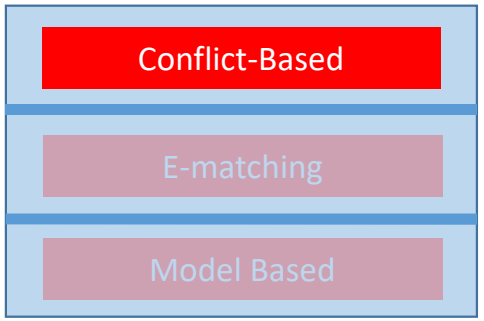
Q

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

CBQI

$$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$$

Conflict-Based Instantiation: EUF



E

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

Q

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

CBQI

$a=g(b)=f(a)$

$b=f(b)$

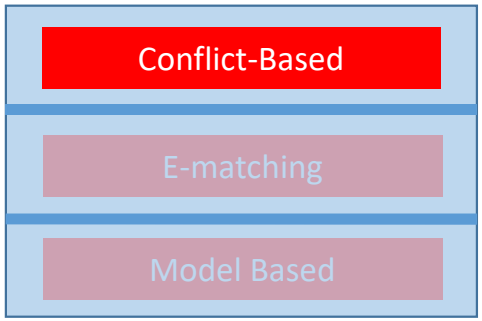
$c=h(b)$

$d=h(f(a))$

Consider the *equivalence classes* of E

$$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$$

Conflict-Based Instantiation: EUF



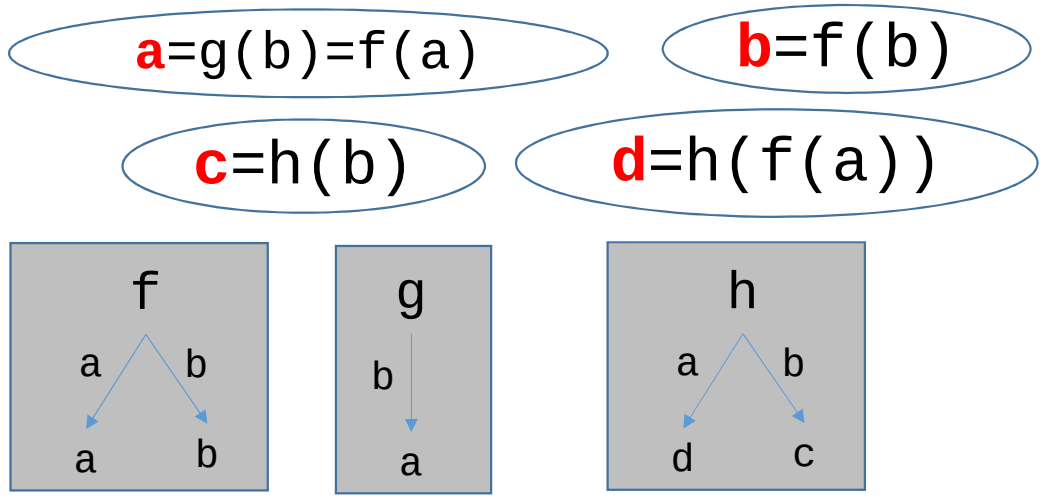
E

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

Q

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

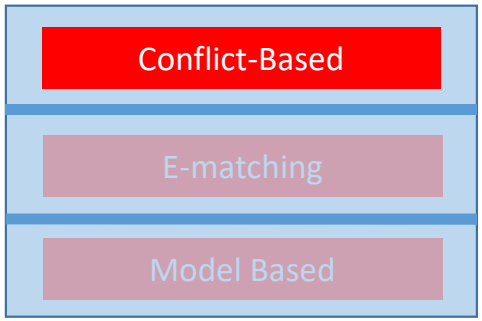
CBQI



Build partial definitions for functions in terms of *representatives*

$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$

Conflict-Based Instantiation: EUF



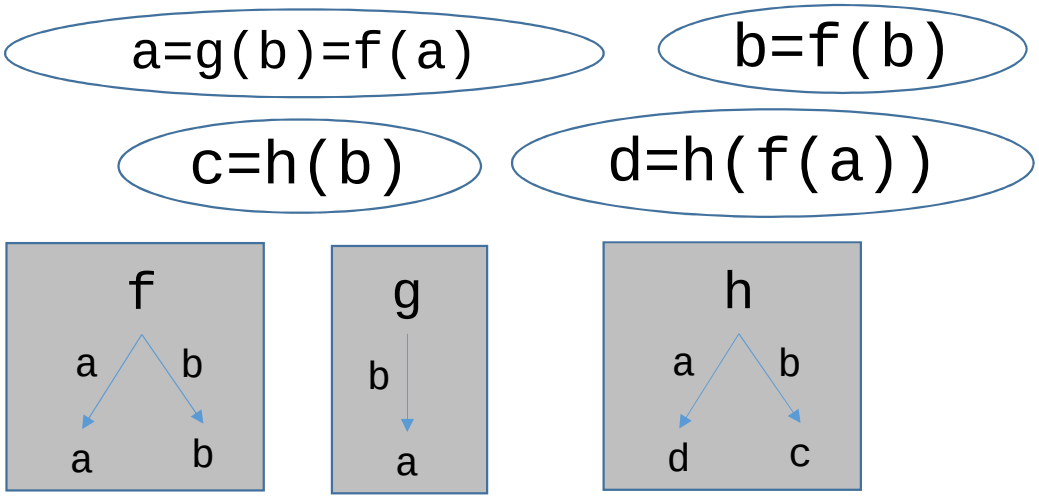
E

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

Q

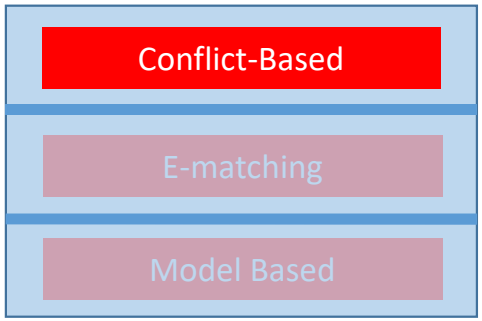
$\forall x. f(g(x))=h(f(x))$

CBQI



$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$

Conflict-Based Instantiation: EUF



E

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

Q

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

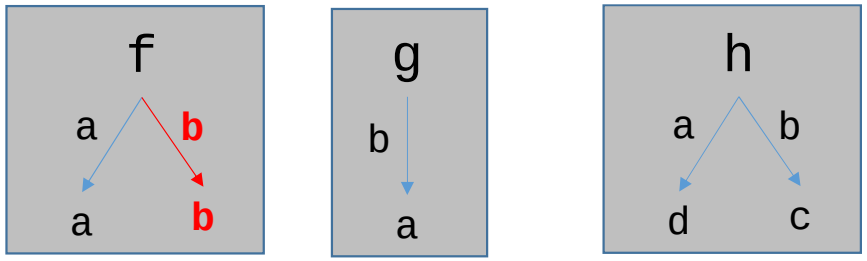
CBQI

$a=g(b)=f(a)$

$b=f(b)$

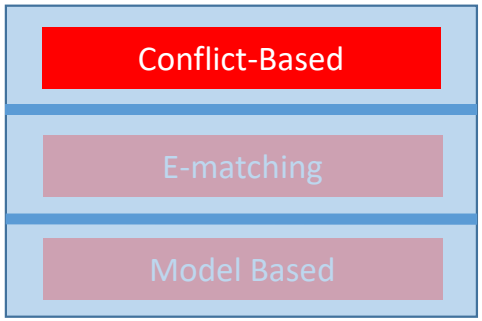
$c=h(b)$

$d=h(f(a))$



$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(\ \mathbf{b}\ )$

Conflict-Based Instantiation: EUF



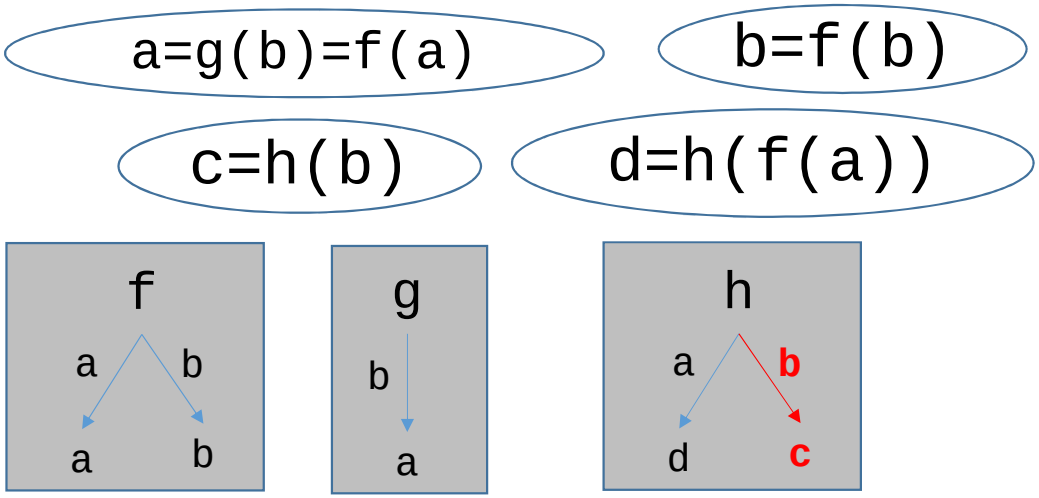
E

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

Q

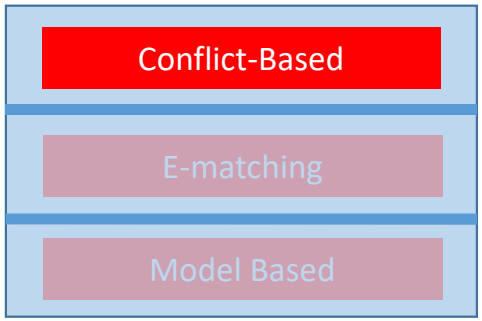
$\forall x. f(g(x))=h(f(x))$

CBQI



$E, f(g(b))=h(f(b)) \models_E f(g(b))=$ **c**

Conflict-Based Instantiation: EUF



E

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

Q

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

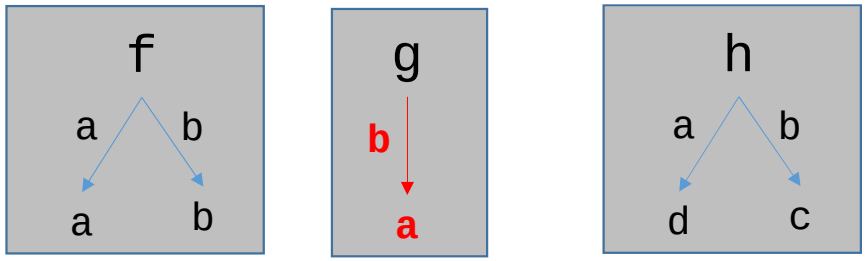
CBQI

$a=g(b)=f(a)$

$b=f(b)$

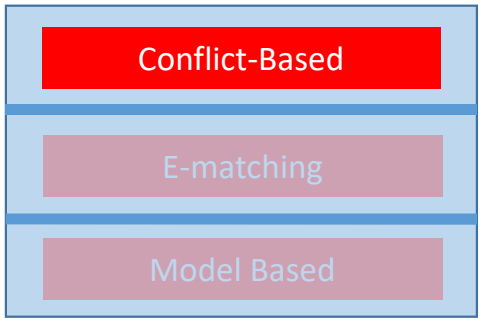
$c=h(b)$

$d=h(f(a))$



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

Conflict-Based Instantiation: EUF



E

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

Q

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

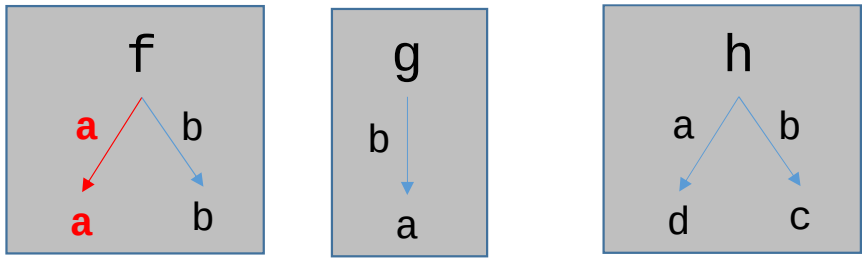
CBQI

$a=g(b)=f(a)$

$b=f(b)$

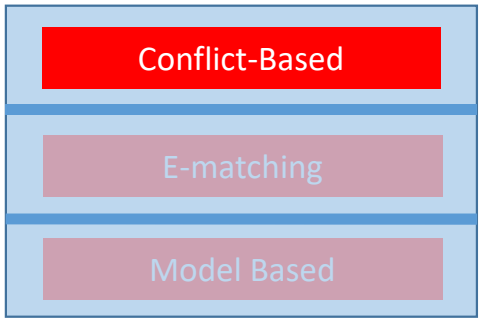
$c=h(b)$

$d=h(f(a))$



$E, f(g(b))=h(f(b)) \models_E$ **a** = c

Conflict-Based Instantiation: EUF



E

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

Q

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

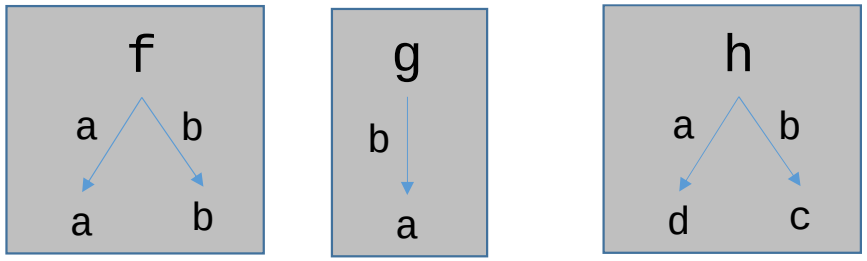
CBQI

$a=g(b)=f(a)$

$b=f(b)$

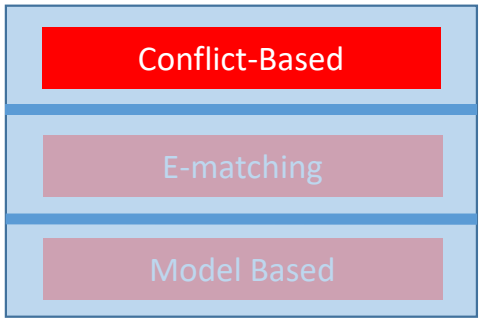
$c=h(b)$

$d=h(f(a))$



$E, f(g(b))=h(f(b)) \models_E \quad a=c$

Conflict-Based Instantiation: EUF



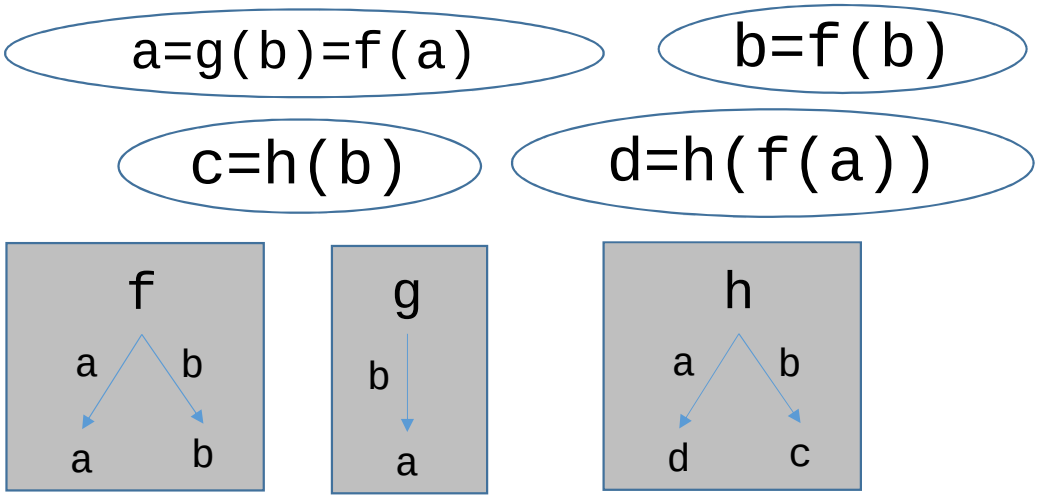
E

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

Q

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

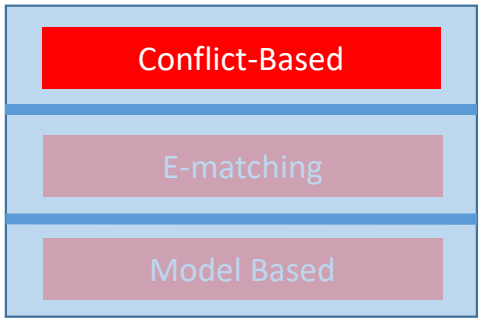
CBQI



$E, f(g(b))=h(f(b)) \models_E$ {

From E, we know $a \neq c$

Conflict-Based Instantiation: EUF



E

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

Q

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

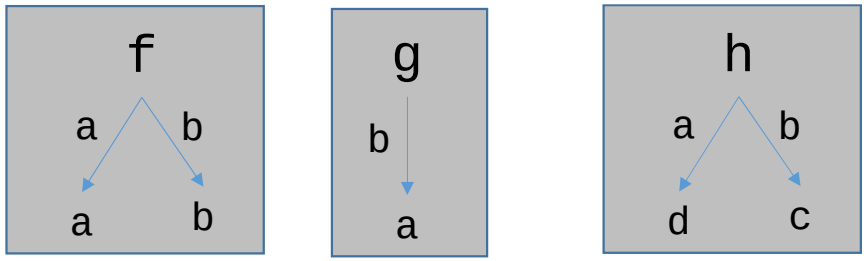
CBQI

$a=g(b)=f(a)$

$b=f(b)$

$c=h(b)$

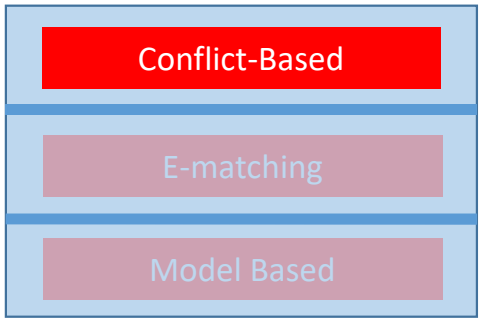
$d=h(f(a))$



$E, f(g(b))=h(f(b)) \models_E \bot$

$f(g(b))=h(f(b))$ is a conflicting instance for (E, Q)!

Conflict-Based Instantiation: EUF



E

```
...,f(b)=b,
g(b)=a,f(a)=a,
h(f(a))=d,h(b)=c
```

Q

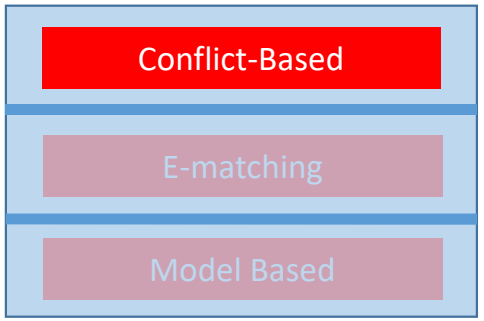
$\forall x.f(g(x))=h(f(x))$

CBQI

$\Rightarrow$ Consider the same example, but where we don't know $a \neq c$

- Is the instance $f(g(b))=h(f(b))$ still useful?

Conflict-Based Instantiation: EUF



E

```
...,f(b)=b,
g(b)=a,f(a)=a,
h(f(a))=d,h(b)=c
```

Q

```
∀x.f(g(x))=h(f(x))
```

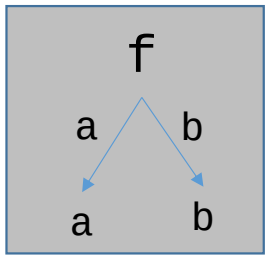
CBQI

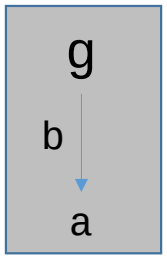
a=g(b)=f(a)

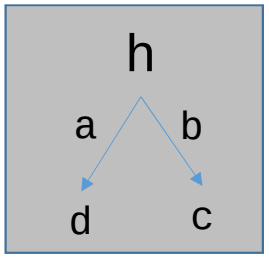
b=f(b)

c=h(b)

d=h(f(a))

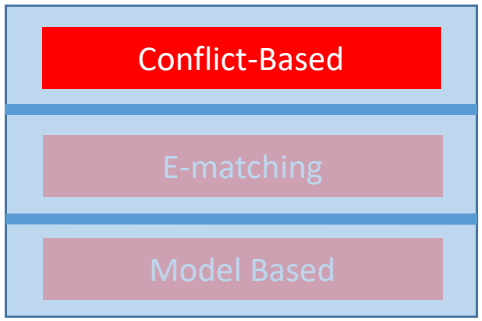






Build partial definitions

Conflict-Based Instantiation: EUF



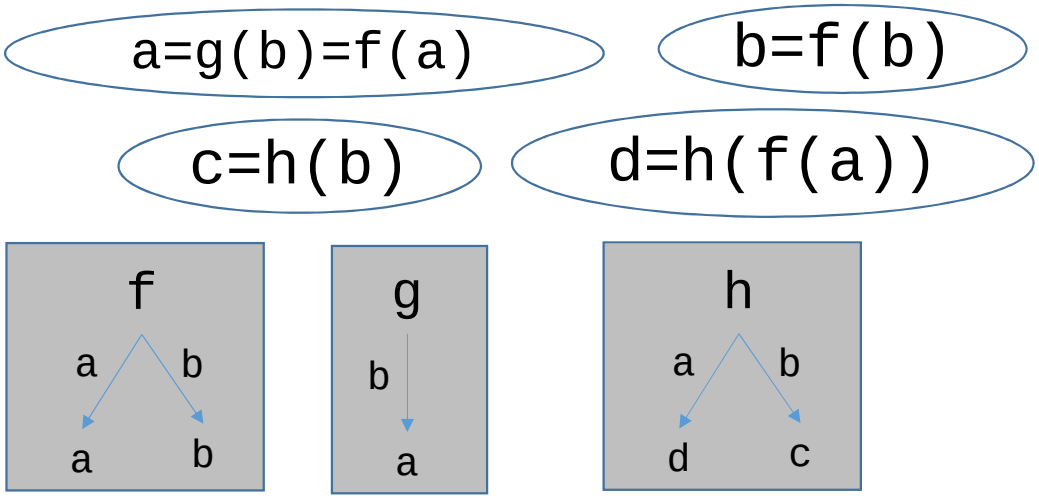
E

```
...,f(b)=b,
g(b)=a,f(a)=a,
h(f(a))=d,h(b)=c
```

Q

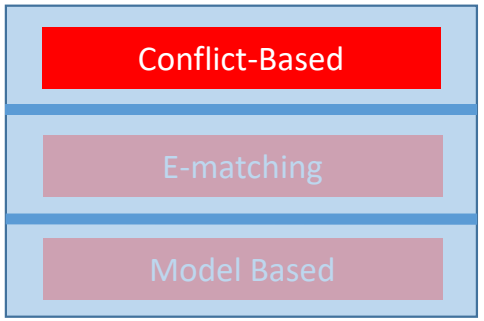
$\forall x.f(g(x))=h(f(x))$

CBQI



$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$ Check entailment

Conflict-Based Instantiation: EUF



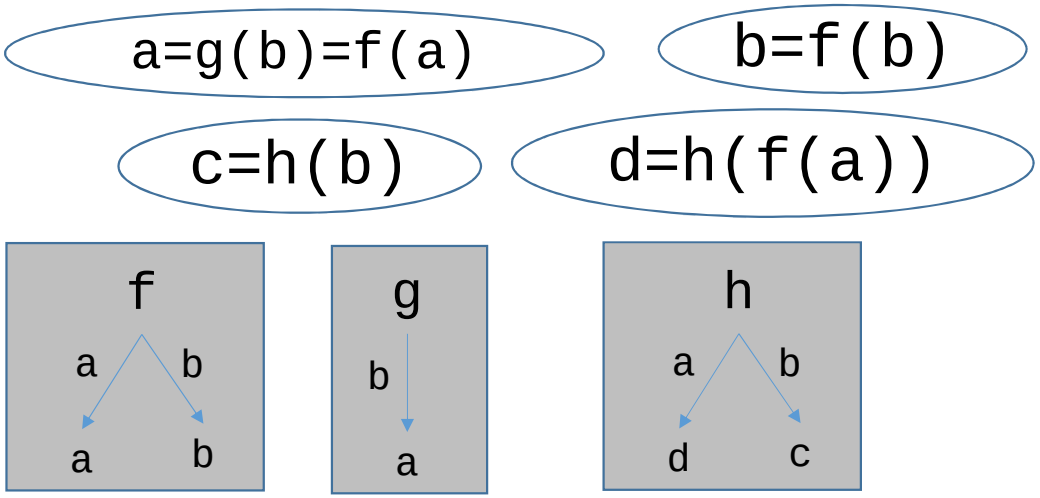
E

```
...,f(b)=b,
g(b)=a,f(a)=a,
h(f(a))=d,h(b)=c
```

Q

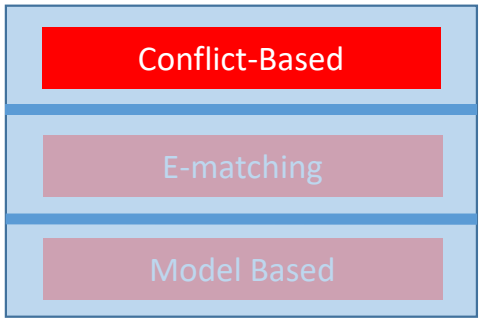
$\forall x.f(g(x))=h(f(x))$

CBQI



$E, f(g(b))=h(f(b)) \models_E a=c$

Conflict-Based Instantiation: EUF



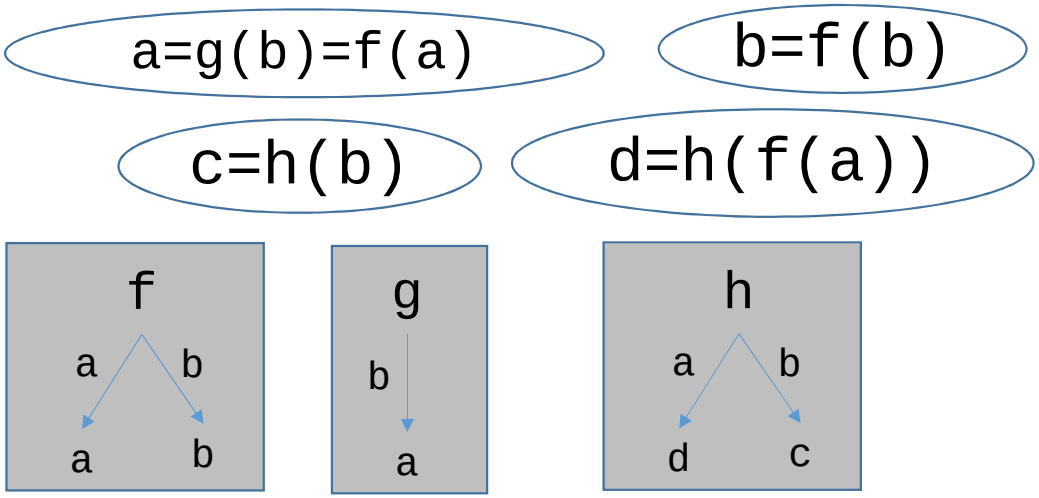
E

```
...,f(b)=b,
g(b)=a,f(a)=a,
h(f(a))=d,h(b)=c
```

Q

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

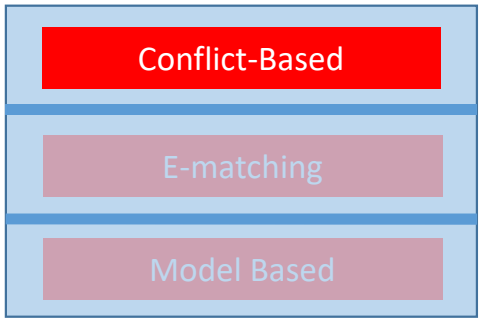
CBQI



$E, f(g(b))=h(f(b)) \models_E$ **a=c**

Instance is *not conflicting*, but *propagates* an equality between two existing terms in E

Conflict-Based Instantiation: EUF



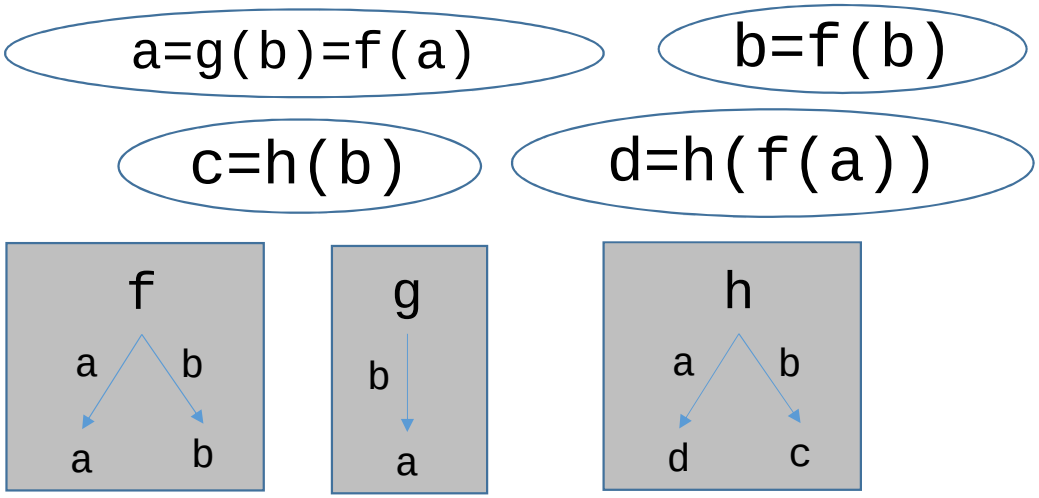
E

```
...,f(b)=b,
g(b)=a,f(a)=a,
h(f(a))=d,h(b)=c
```

Q

```
∀x.f(g(x))=h(f(x))
```

CBQI

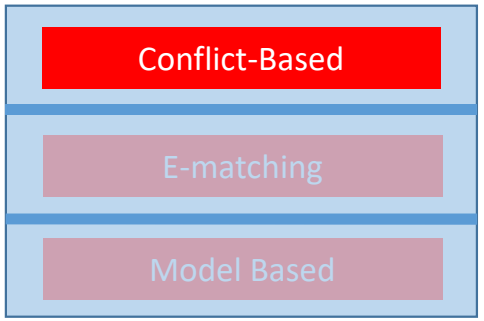


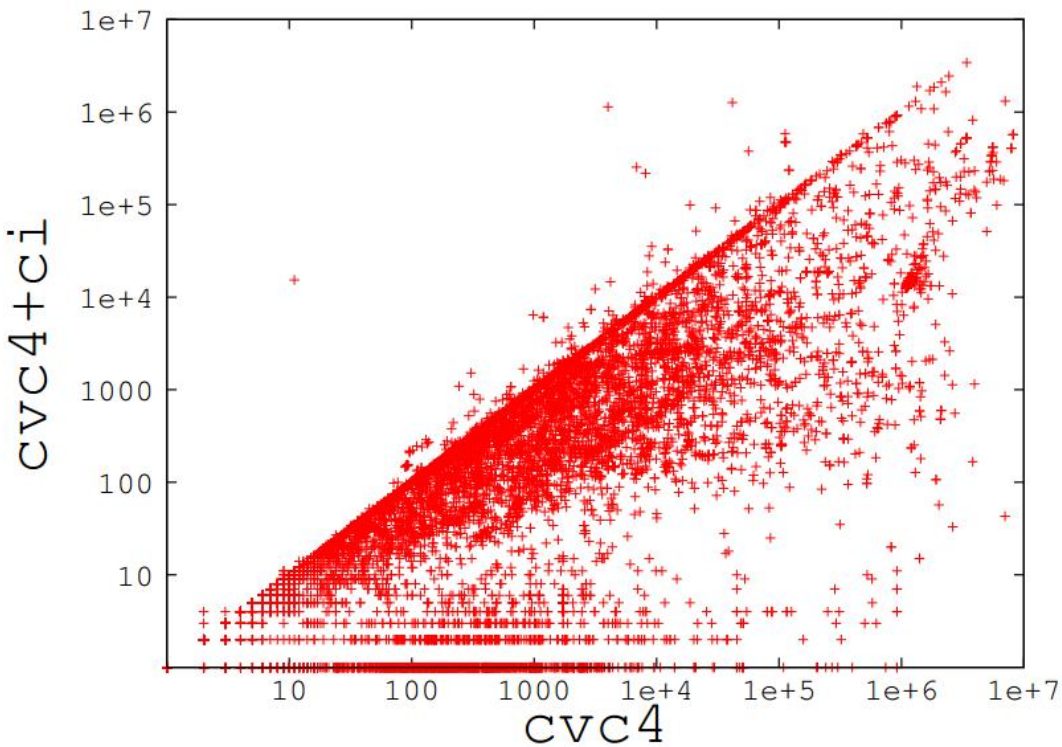
`f(g(b))=h(f(b))` is a propagating instance for `(E,Q)`

`E,f(g(b))=h(f(b))` $\models_E$ `a=c`

Ĝ *These are also useful*

Conflict-Based Instantiation: Impact



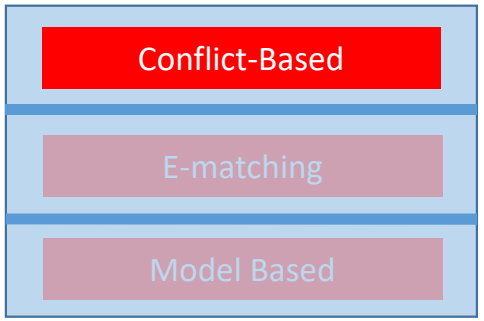


Reported number of instances.

- Using conflict-based instantiation (**`cvc4+ci`**), require an order of magnitude fewer instances for showing "UNSAT" wrt E-matching alone

(taken from **[Reynolds et al FMCAD14]**, evaluation On SMTLIB, TPTP, Isabelle benchmarks)

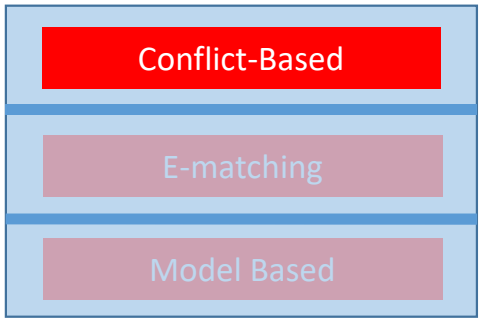
Conflict-Based Instantiation: Impact



- Conflicting instances found on ~75% of rounds (IR)
- Configuration **`cvc4+ci`**:
  - Calls E-matching 1.5x fewer times overall
  - As a result, returns 5x fewer instantiations

			E-matching		Conflict Inst.		Propagating Inst.	
		IR	% IR	# Inst	% IR	# Inst	% IR	# Inst
TPTP	**cvc4**	71,634	100.0	878,957,688				
	cvc4+ci	208,970	20.3	150,351,384	76.4	159,696	3.3	415,772
Isabelle	**cvc4**	6,969	100.0	119,008,834				
	cvc4+ci	21,756	22.4	28,196,846	64.0	13,932	13.6	130,864
SMT-LIB	**cvc4**	14,032	100.0	60,650,746				
	cvc4+ci	58,003	20.0	32,305,788	71.6	41,531	8.4	51,454

Conflict-Based Instantiation: Impact

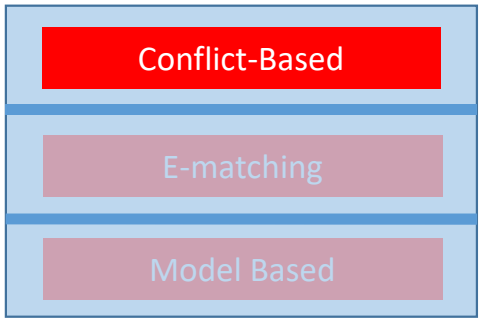


- CVC4 with conflicting instances **`cvc4+ci`**
  - Solves the most benchmarks for TPTP and Isabelle
  - Requires almost an order of magnitude fewer instantiations

	TPTP		Isabelle		SMT-LIB	
	Solved	Inst	Solved	Inst	Solved	Inst
cvc3	5,245	627.0M	3,827	186.9M	3,407	42.3M
z3	6,269	613.5M	3,506	67.0M	**3,983**	6.4M
cvc4	6,100	879.0M	3,858	119.0M	3,680	60.7M
cvc4+ci	**6,616**	150.9M	**4,082**	28.2M	3,747	32.4M

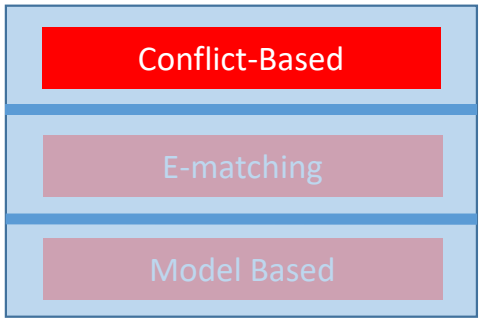
Ĝ *A number of hard benchmarks can be solved without resorting to E-matching at all*

Challenge : Finding Conflicting Instances



- How do we *find* conflicting instances?
  - Idea: construct instances via a stronger version of matching
    - Intuition: for $\forall x . P(x) \vee Q(x)$, will only match $P(x)$ where $P(t) \Leftrightarrow \bot$
      (see **[Reynolds et al FMCAD2014])**
  - Formalized as calculus based on E-ground (dis)unification **[Barbosa et al 2017]**

Challenge : Theory Symbols

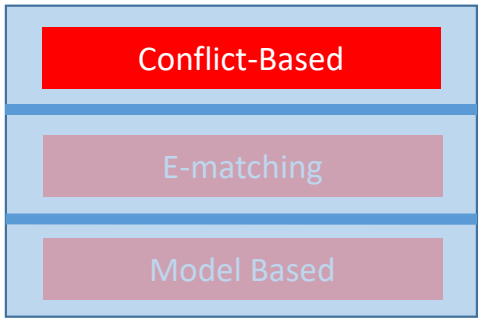


- Difficult for quantified formulas that contain *theory symbols:*

E `f(1)=5`

Q `∀xy.f(x+y)>x+2*y`

Challenge : Theory Symbols



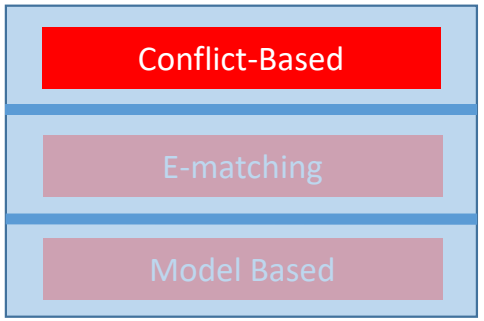
- Difficult for quantified formulas that contain *theory symbols:*

E: `f(1)=5`

Q: `∀xy.f(x+y)>x+2*y`

→ `∀xy.f(x+y)>x+2*y⇒ f(-3+4)>-3+2*4`

Challenge : Theory Symbols



- Difficult for quantified formulas that contain *theory symbols:*

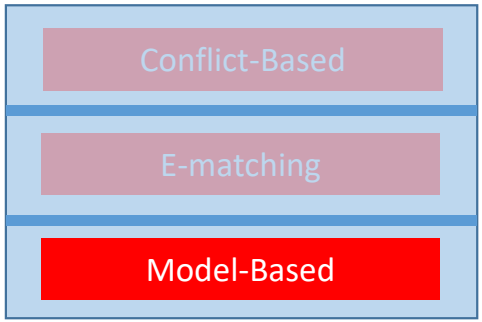
E: `f(1)=5`

Q: `∀xy.f(x+y)>x+2*y`

→ `∀xy.f(x+y)>x+2*y⇒`
`f(1)>5`

⇒ Generally, use fast and incomplete procedure for ∀+theories

Model-based Instantiation

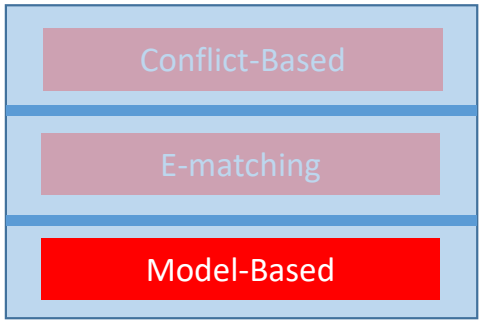


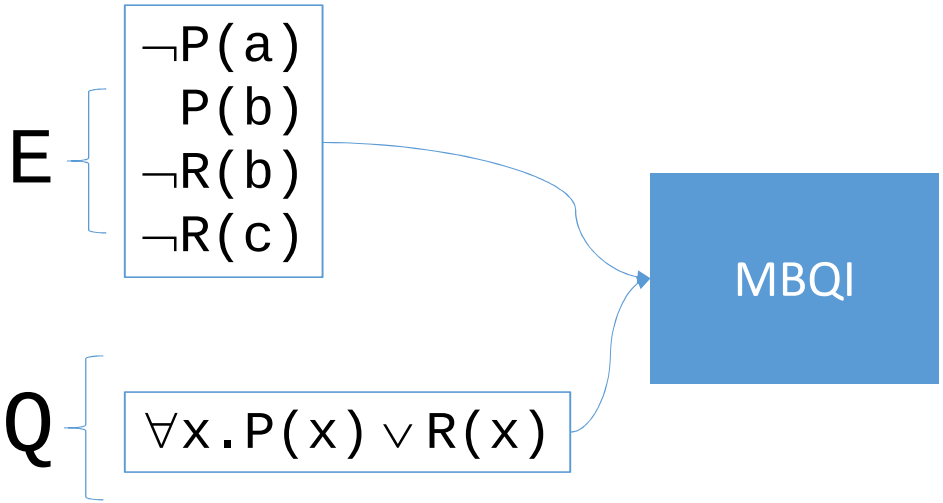
- Basic idea:
  - If E-matching saturates, build "candidate model" $\mathscr{M}$ satisfying E
    - Check if $\mathscr{M}$ also satisfies Q

      (using a quantifier-free satisfiability query)

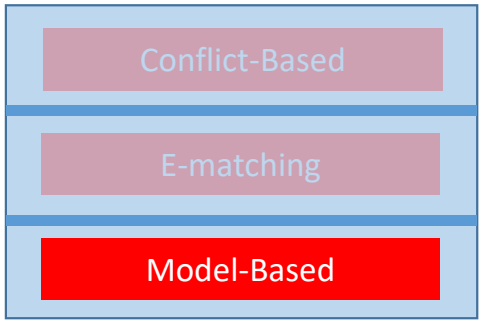
Ĝ *Ability to answer* sat

Model-based Instantiation





Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}$

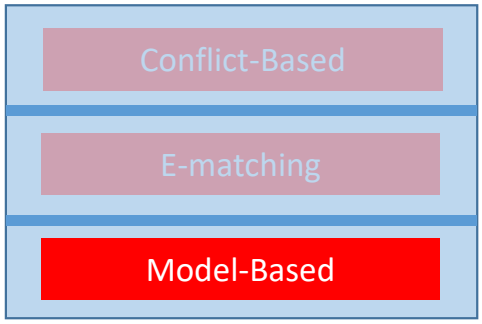
$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$\ldots)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\ldots)))$

Build interpretation $\mathscr{M}$ of predicates

- This interpretation must satisfy E

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}$

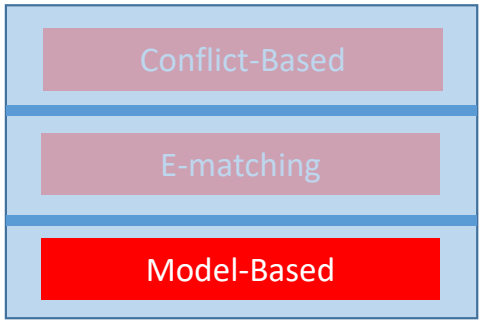
$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$T)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

Build interpretation $\mathscr{M}$ of predicates

- This interpretation must satisfy E
- Missing values may be filled in arbitrarily

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}$

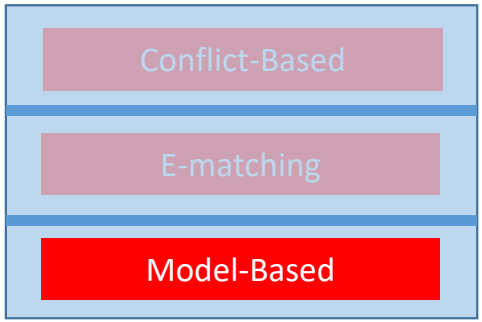
$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$T)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

$\Rightarrow$ Does $\mathscr{M}$ satisfy Q?

- Check (un)satisfiability of: $\exists x.\neg(P^{\mathscr{M}}(x) \vee R^{\mathscr{M}}(x))$

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

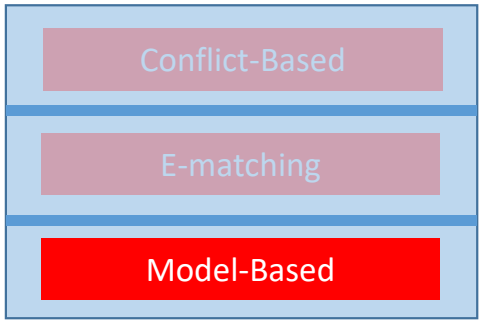
$\mathscr{M}$

$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$T)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

Check: $\exists x.\neg(P^{\mathscr{M}}(x) \vee R^{\mathscr{M}}(x))$

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}$

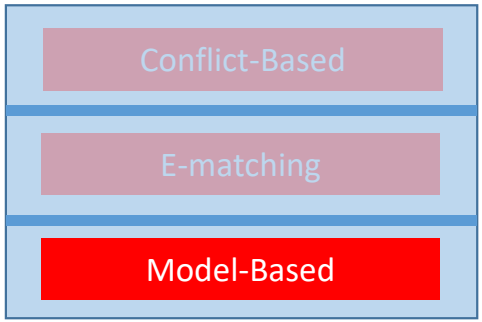
$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$T)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

Check: $\neg(P^{\mathscr{M}}(\mathbf{k}) \vee R^{\mathscr{M}}(\mathbf{k}))$

$\Rightarrow$ Skolemize

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}$

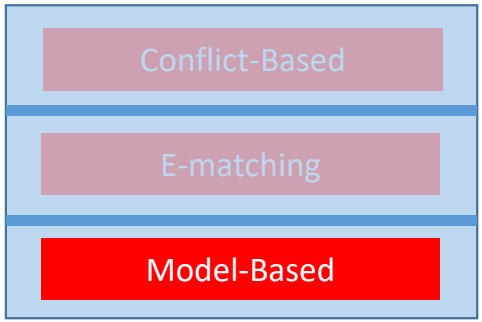
$P^{\mathscr{M}} \Leftrightarrow$
`|x. ite(x=a,{, ite(x=b,T, ,T)))`

$R^{\mathscr{M}} \Leftrightarrow$
`|x. ite(x=b,{, ite(x=c,{, ,{)))`

Check: $\neg$(`ite(k=a,{,ite(k=b,T,T)))`$\vee$
`ite(k=b,{,ite(k=c,{,{)))`)

$\Rightarrow$ Substitute

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}$

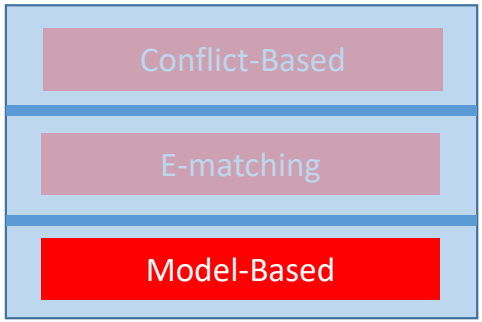
$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$T)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

Check: $\neg(k \neq a \vee \bot)$

$\Rightarrow$ Simplify

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

Check: k=a

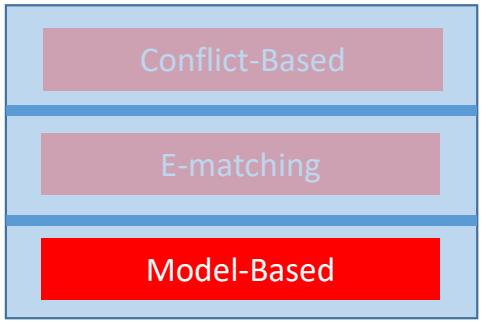
$\mathscr{M}$

$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$T)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

$\Rightarrow$ Simplify

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}$

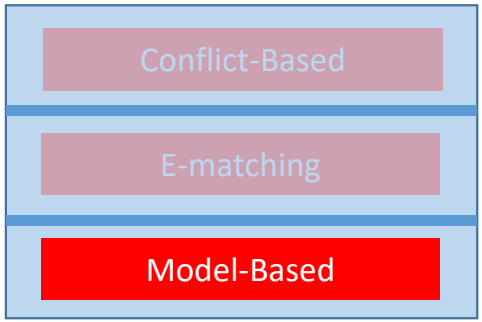
$P^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=a, \{,$
$ite(x=b, T,$
$T)))$

$R^{\mathscr{M}} \Leftrightarrow$
$\lambda x.$
$ite(x=b, \bot,$
$ite(x=c, \bot,$
$\{)))$

Check: **k=a**

$\Rightarrow$ *Satisfiable! There are values k for which $\mathscr{M}$ does not satisfy Q*

Model-based Instantiation



E

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

Q

$\forall x.P(x) \vee R(x)$

MBQI

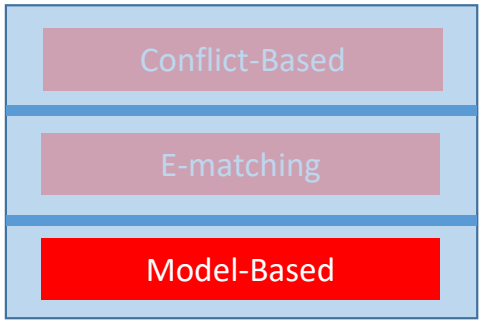
return

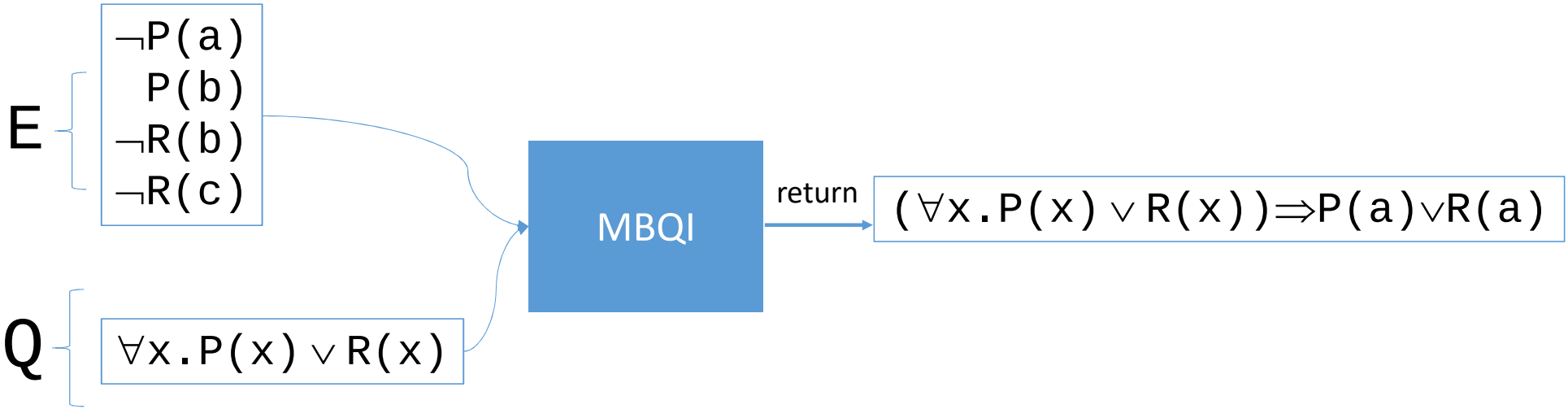
$(\forall x.P(x) \vee R(x)) \Rightarrow P(\mathbf{a}) \vee R(\mathbf{a})$

Check: **k=a**

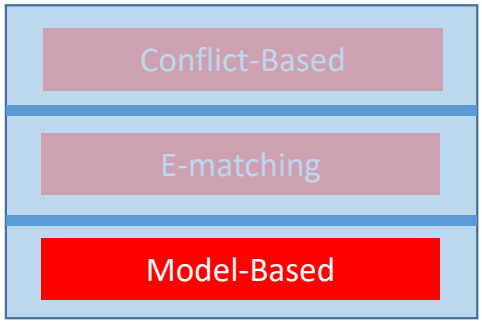
$\Rightarrow$ Add one instance
for one such value of k
for which $\mathscr{M}$ did satisfy Q

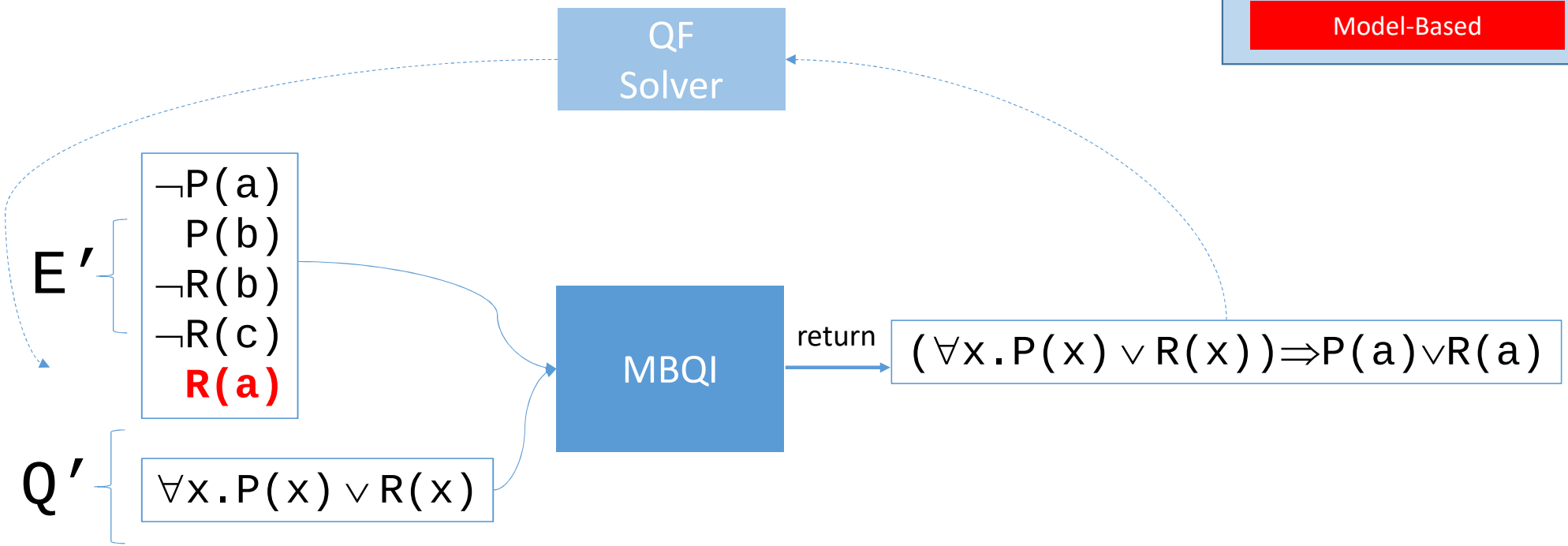
Model-based Instantiation





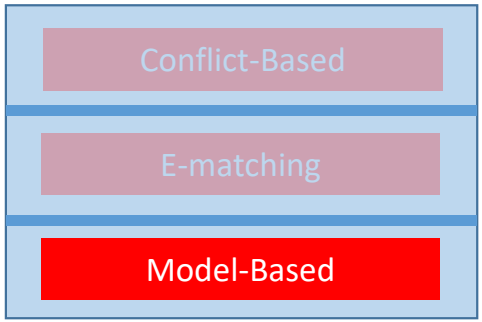
Model-based Instantiation

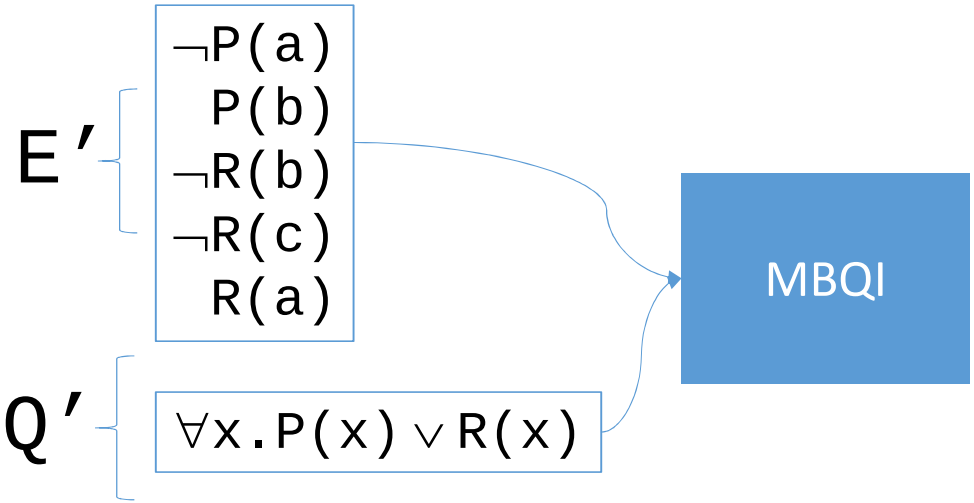




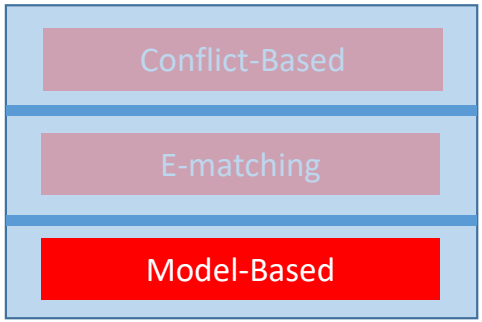
$\Rightarrow$ Subsequent models must satisfy $P(a) \vee \mathbf{R(a)}$

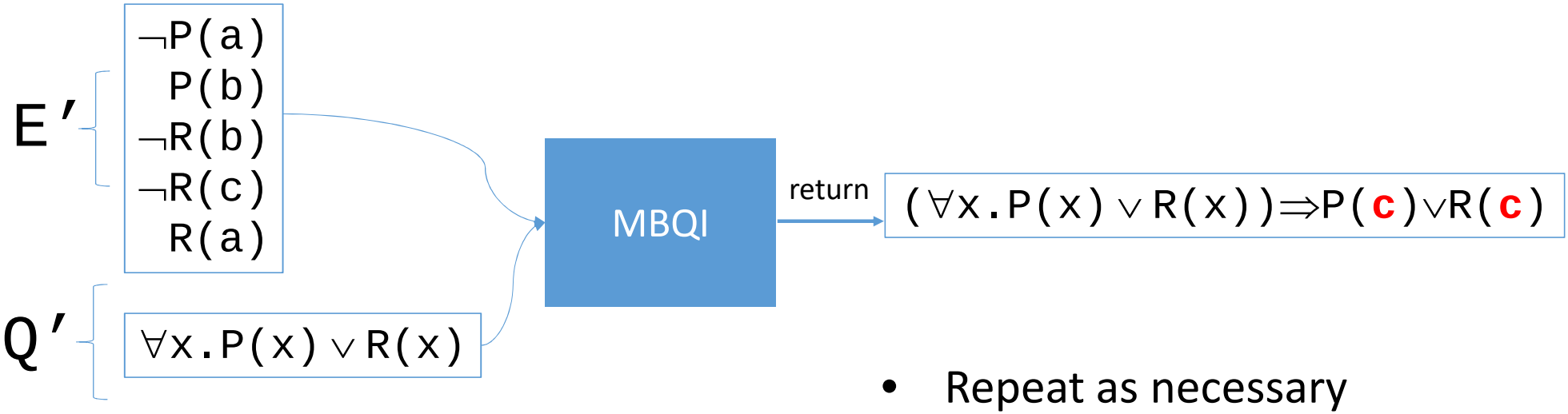
Model-based Instantiation





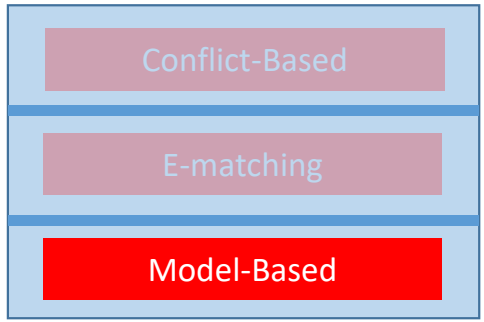
Model-based Instantiation

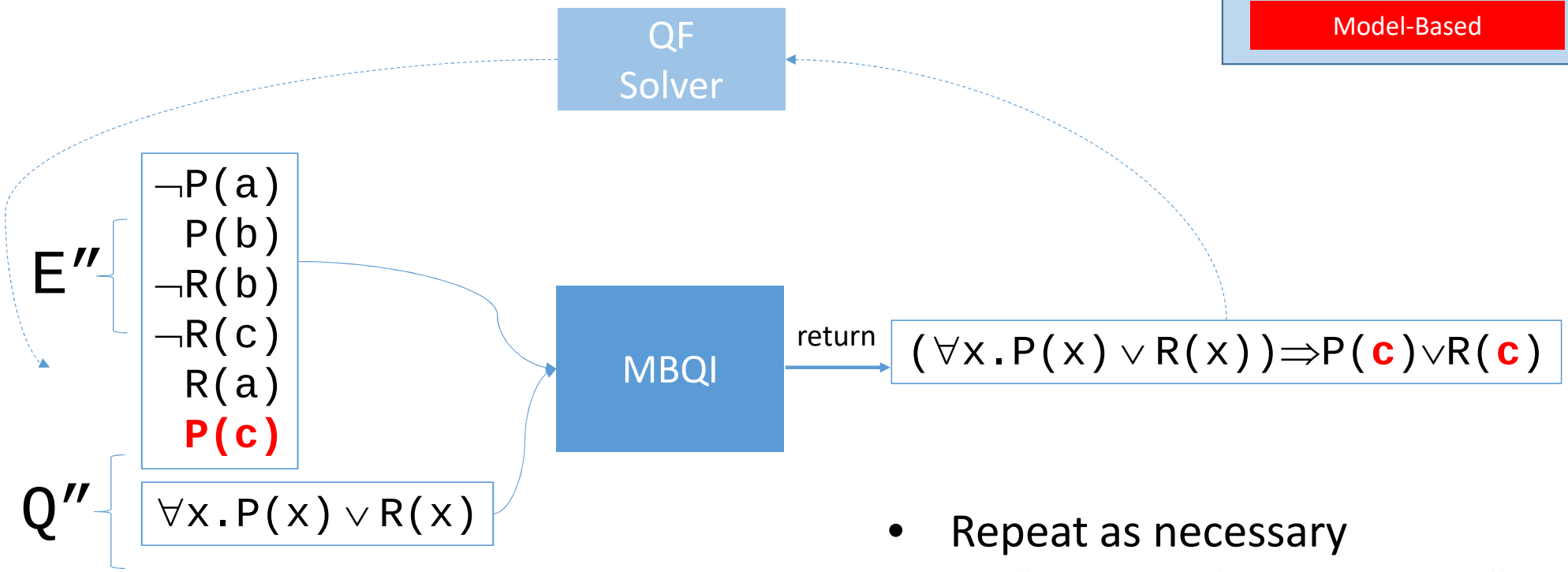




- Repeat as necessary
  ⇒ “Model refinement loop”

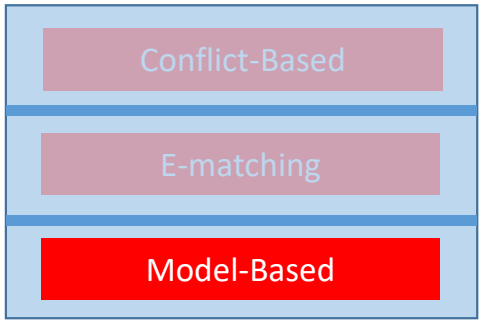
Model-based Instantiation





- Repeat as necessary
  ⇒ "Model refinement loop"

Model-based Instantiation



E''

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

Q''

$\forall x.P(x) \vee R(x)$

MBQI

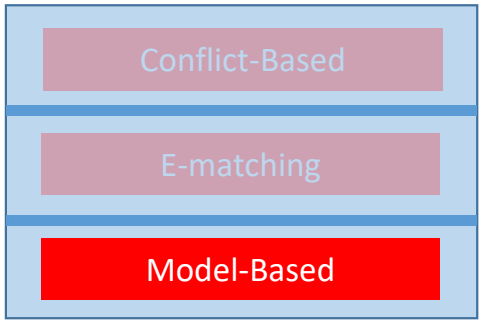
$\mathscr{M}''$

$P^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$ite(x=c,T,$
$T)))$

$R^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a,T,$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

Check: $\exists x.\neg(P^{\mathscr{M}''}(x) \vee R^{\mathscr{M}''}(x))$

Model-based Instantiation



E''

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

Q''

$\forall x.P(x) \vee R(x)$

MBQI

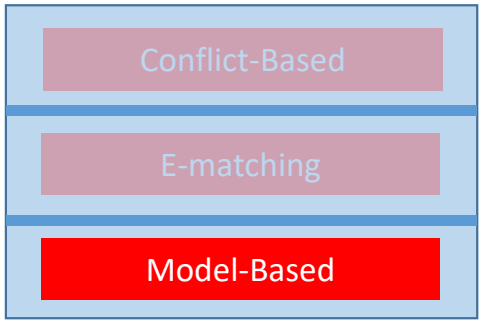
Check: $k=a \wedge k \neq a$

$\mathscr{M}''$

$P^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a, \{,$
$ite(x=b, T,$
$ite(x=c, T,$
$T)))$

$R^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a, T,$
$ite(x=b, \{,$
$ite(x=c, \{,$
$\{)))$

Model-based Instantiation



E''

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

Q''

$\forall x.P(x) \vee R(x)$

MBQI

$\mathscr{M}''$

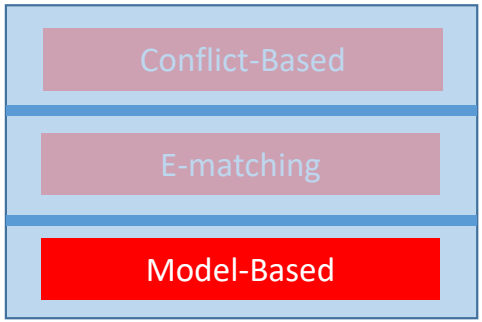
$P^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$ite(x=c,T,$
$T)))$

$R^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a,T,$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

Check: $k=a \wedge k\neq a$

$\Longrightarrow$ Unsatisfiable, there are no values k for which $\mathscr{M}''$ does not satisfy Q

Model-based Instantiation



E''

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

Q''

$\forall x.P(x) \vee R(x)$

MBQI

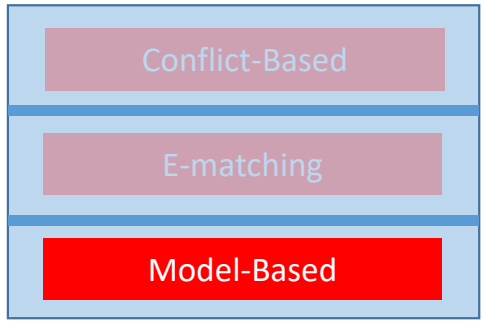
$\mathscr{M}''$

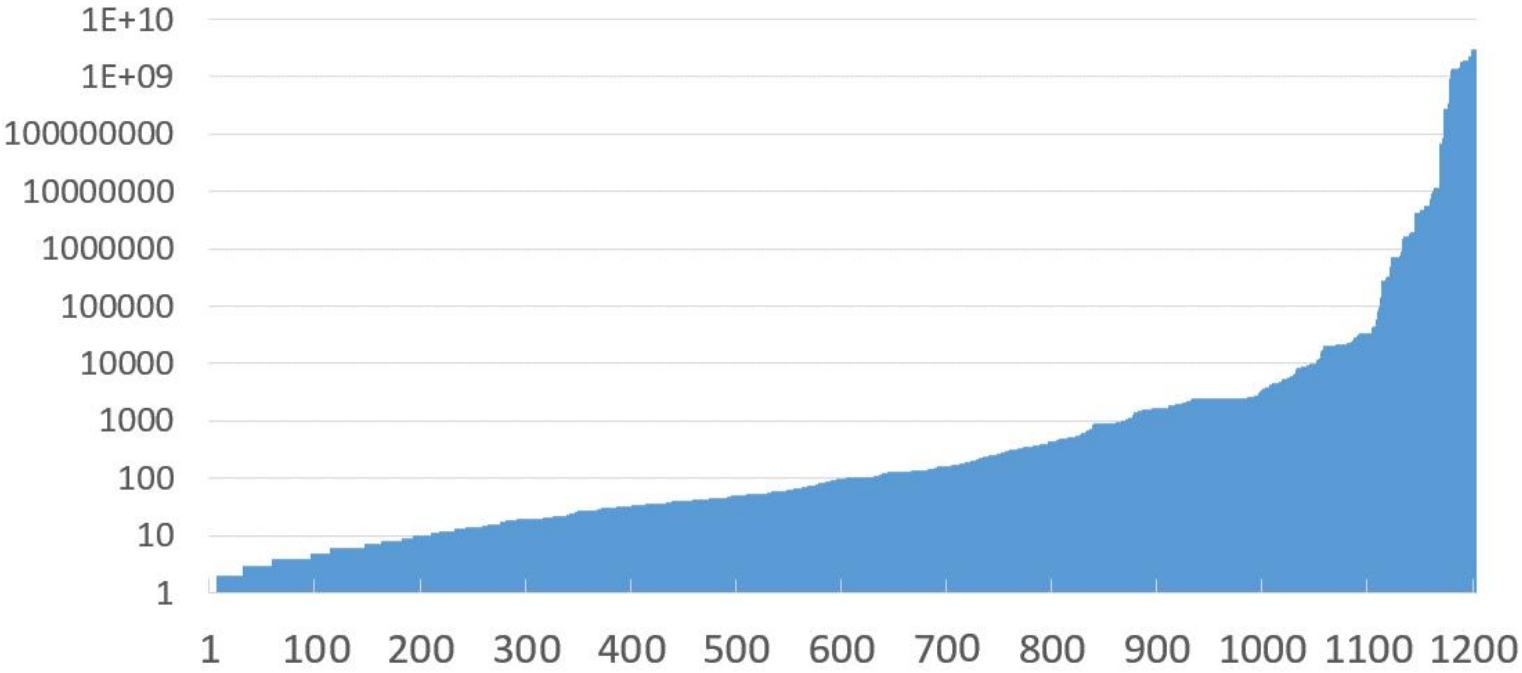
$P^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a,\{,$
$ite(x=b,T,$
$ite(x=c,T,$
$T)))$

$R^{\mathscr{M}''} \Leftrightarrow$
$\lambda x.$
$ite(x=a,T,$
$ite(x=b,\{,$
$ite(x=c,\{,$
$\{)))$

sat, model $\mathscr{M}''$

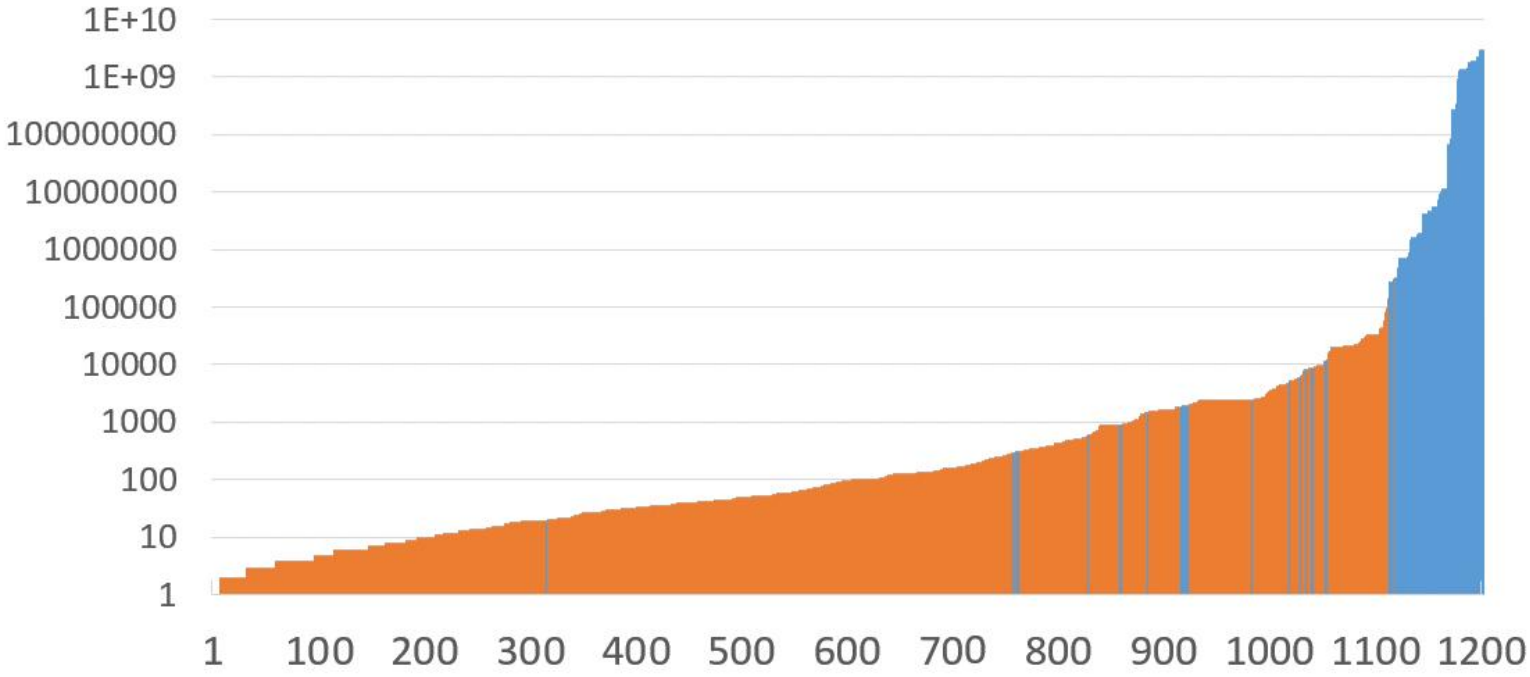
Model-based Instantiation: Impact

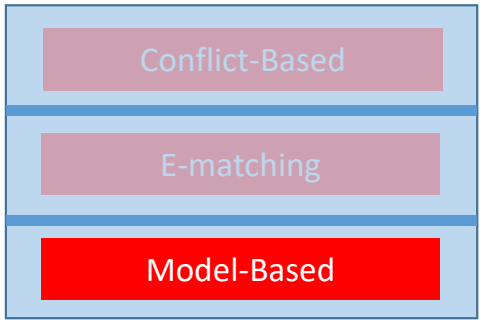




- 1203 satisfiable benchmarks from the TPTP library
  - Graph shows # instances required by exhaustive instantiation
    - E.g. $\forall xyz:U.P(x,y,z)$, if $|U|=4$, requires $4^3=64$ instances

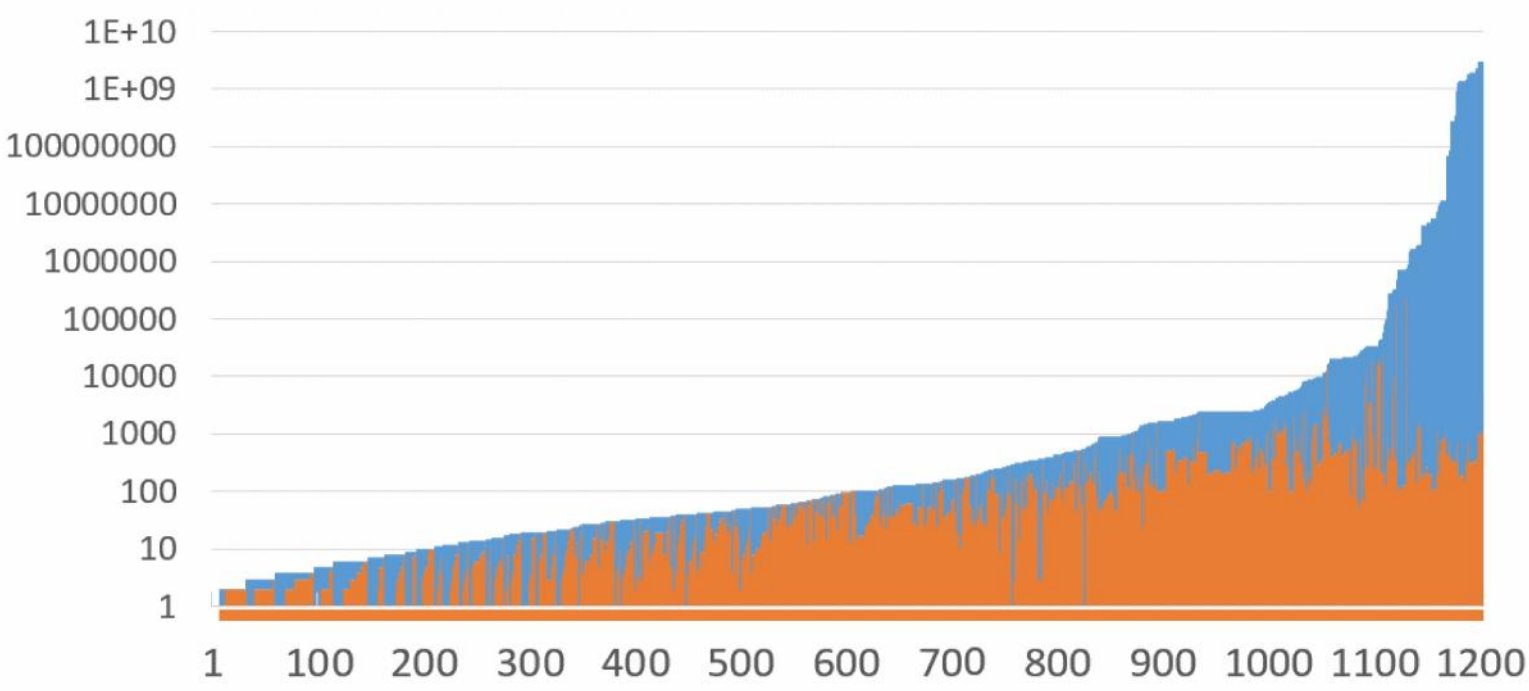
Model-based Instantiation: Impact

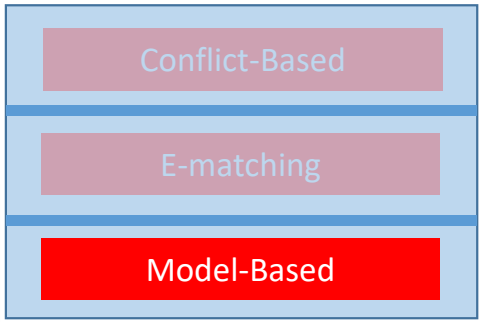




- CVC4 Finite Model Finding + Exhaustive instantiation
  - Scales only up to ~150k instances with a 30 sec timeout

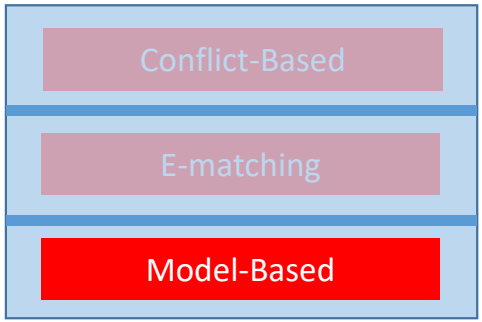
Model-based Instantiation: Impact





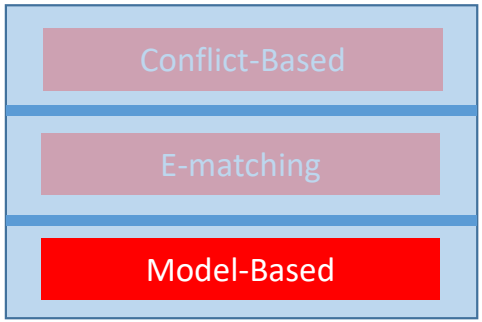
- CVC4 Finite Model Finding + Model-Based instantiation **[Reynolds et al CADE13]**
  - Scales to >2 billion instances with a 30 sec timeout, only adds fraction of possible instances

Model-based Instantiation: Challenges



- How do we build interpretations $\mathscr{M}$?
  - Typically, build interpretations $f^{\mathscr{M}}$ that are almost constant:
    - e.g. $f^{\mathscr{M}} := \lambda x.\mathtt{ite}(x=t_1, v_1, \mathtt{ite}(x=t_2, v_2, \ldots, \mathtt{ite}(x=t_n, v_n, v_{def})\ldots))$

Model-based Instantiation: Challenges



- How do we build interpretations $\mathscr{M}$?
  - Typically, build interpretations $f^{\mathscr{M}}$ that are almost constant:
    - e.g. $f^{\mathscr{M}} := \lambda x.\mathtt{ite}(x=t_1, v_1, \mathtt{ite}(x=t_2, v_2, \ldots, \mathtt{ite}(x=t_n, v_n, v_{def})\ldots))$

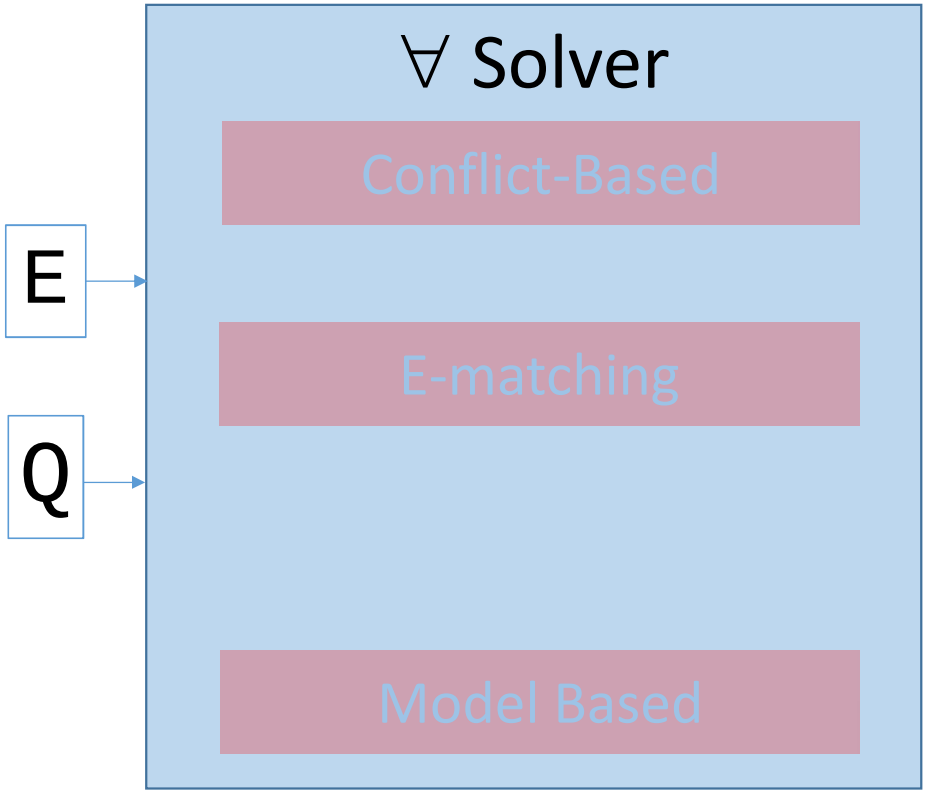
…but models may need to be more complex when *theories are present*:

$\forall xy{:}\mathtt{Int}.(f(x,y)\geq x \wedge f(x,y)\geq y)$ ⇢ $f^{\mathscr{M}} := \lambda xy.\mathtt{ite}(x\geq y, x, y)$

$\forall x{:}\mathtt{Int}.3*g(x)+5*h(x)=x$ ⇢ $g^{\mathscr{M}} := \lambda x.-3*x$, $h^{\mathscr{M}} := \lambda x.2*x$

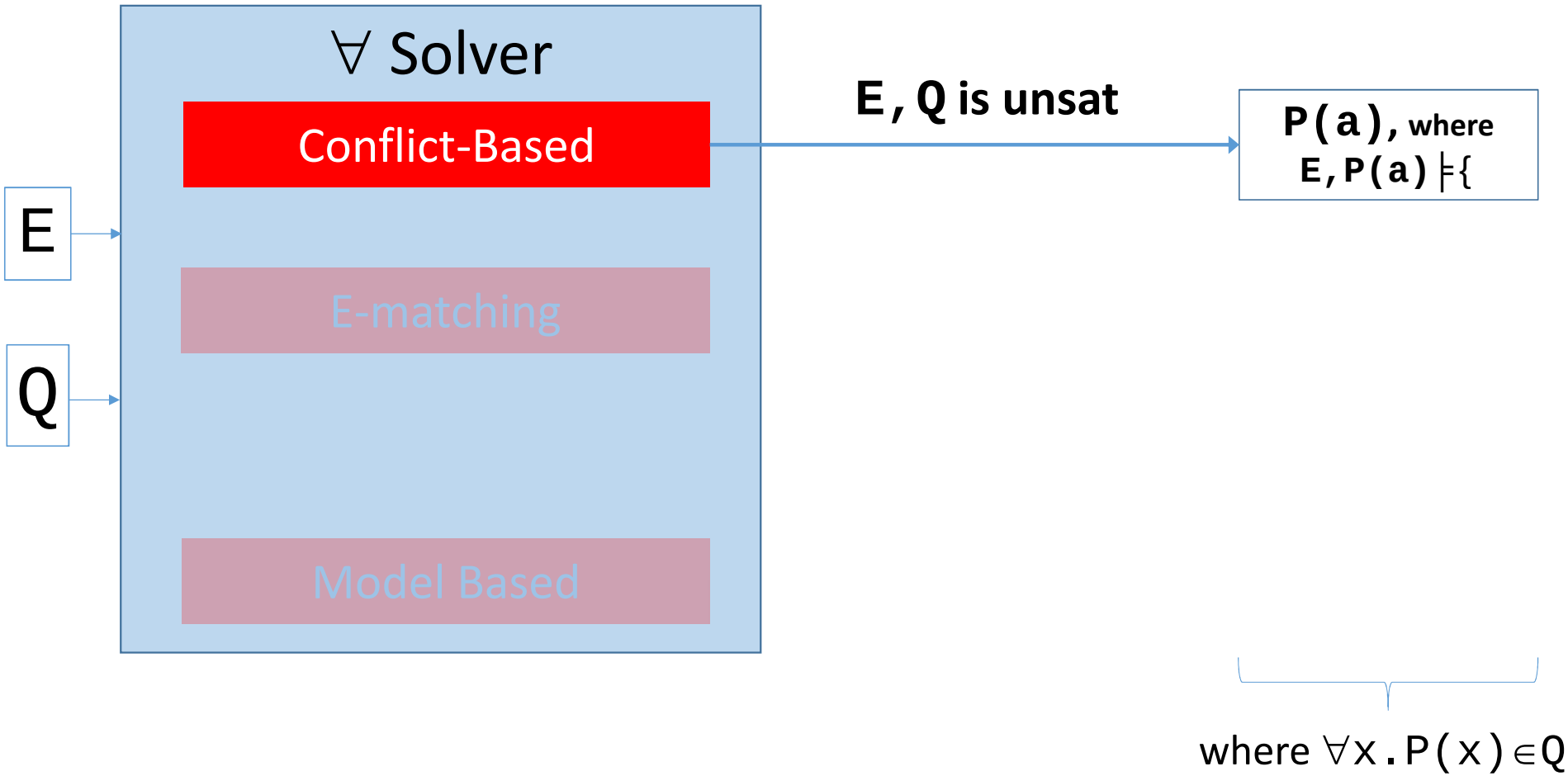
$\forall xy{:}\mathtt{Int}.u(x+y)+11*v(w(x))=x+y$ ⇢ ???

Putting it Together

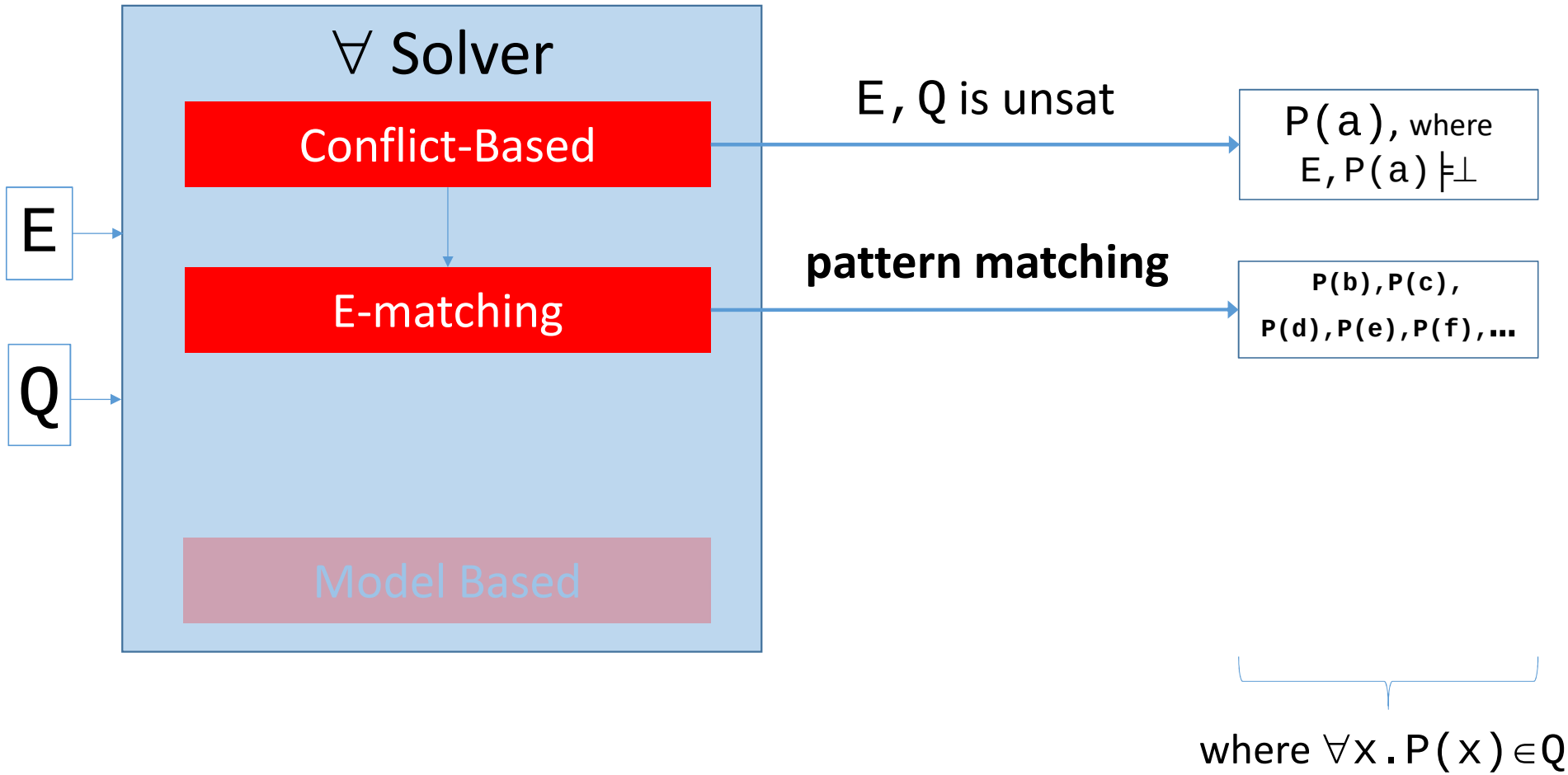


- Input:
  - Ground literals E
  - Quantified formulas Q

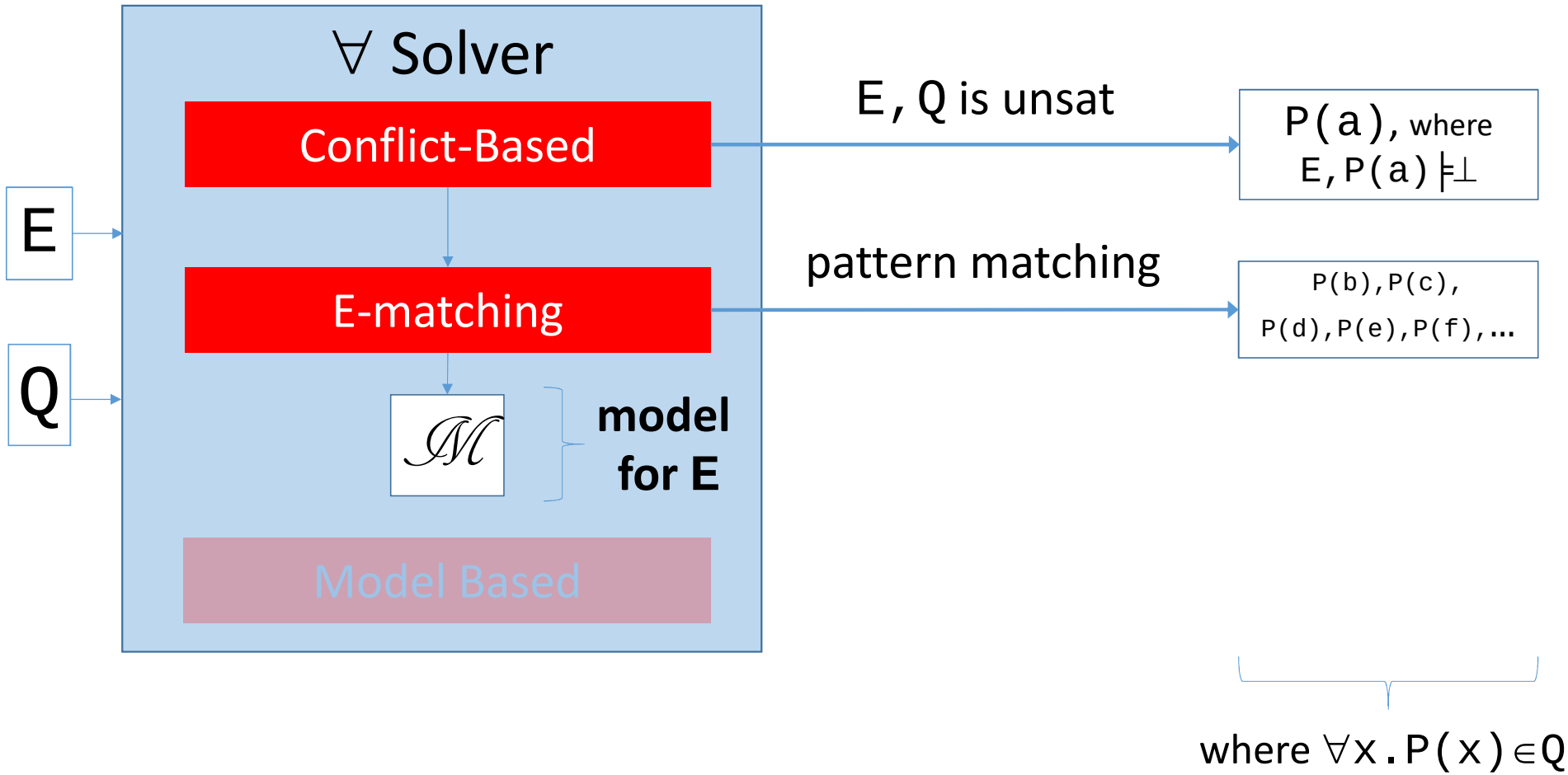
Putting it Together



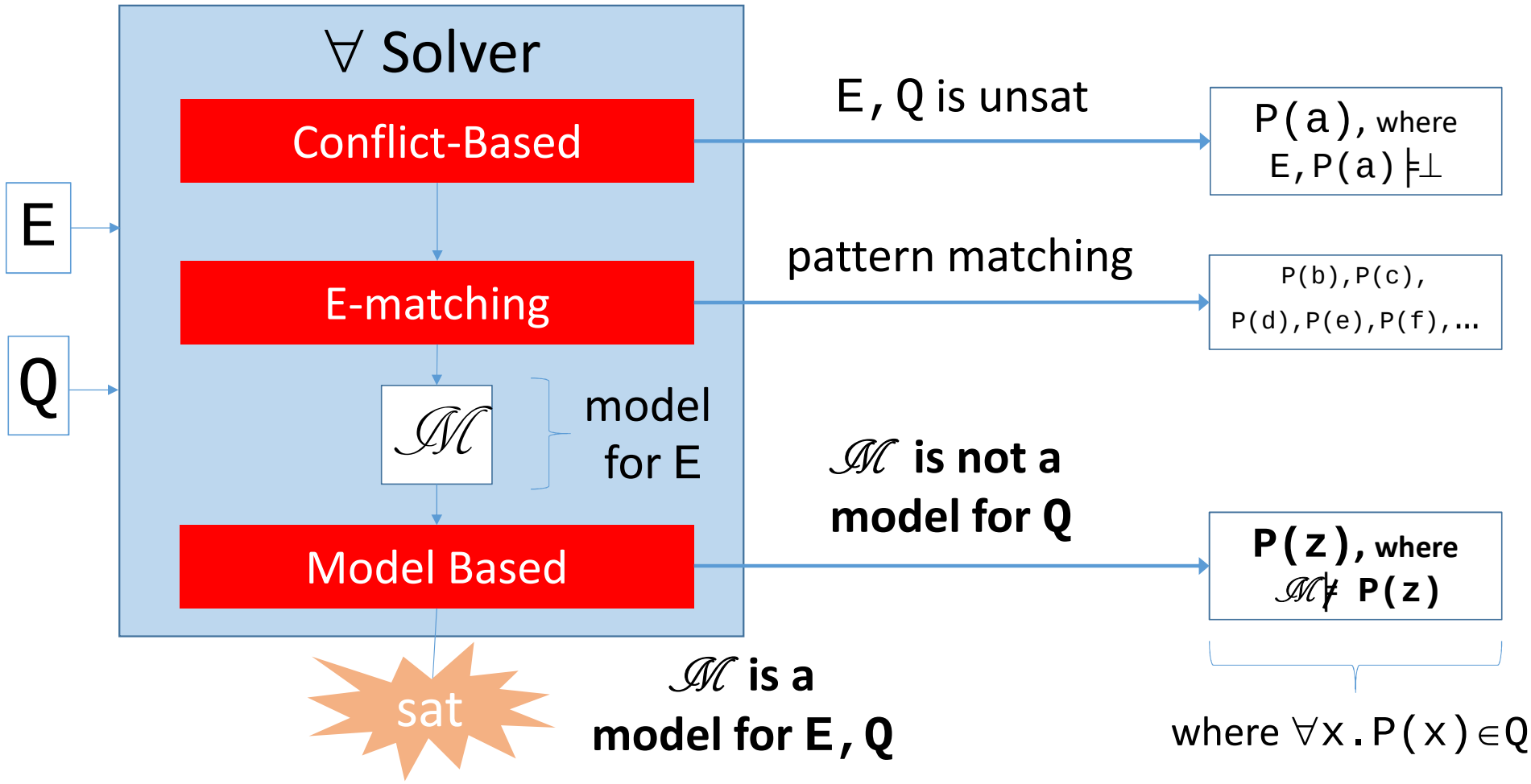
Putting it Together



Putting it Together



Putting it Together



E-matching, Conflict-Based, Model-based:

- **Common thread:** satisfiability of $\forall$ + UF + theories is hard!
  - E-matching:
    - Pattern selection, matching modulo theories
  - Conflict-based:
    - Matching is incomplete, entailment tests are expensive
  - Model-based:
    - Models are complex, interpreted domains (e.g. Int) may be infinite

E-matching, Conflict-Based, Model-based:

- **Common thread:** satisfiability of $\forall$ + UF + theories is hard!
  - E-matching:
    - Pattern selection, matching modulo theories
  - Conflict-based:
    - Matching is incomplete, entailment tests are expensive
  - Model-based:
    - Models are complex, interpreted domains (e.g. Int) may be infinite

$\Rightarrow$ But reasoning about $\forall$ + theories without UF isn't as bad:

- Classic $\forall$-elimination algorithms are decision procedures for $\forall$ in:
  - LRA [Ferrante+Rackoff 79, Loos+Wiespfenning 93] , LIA [Cooper 72], datatypes, …

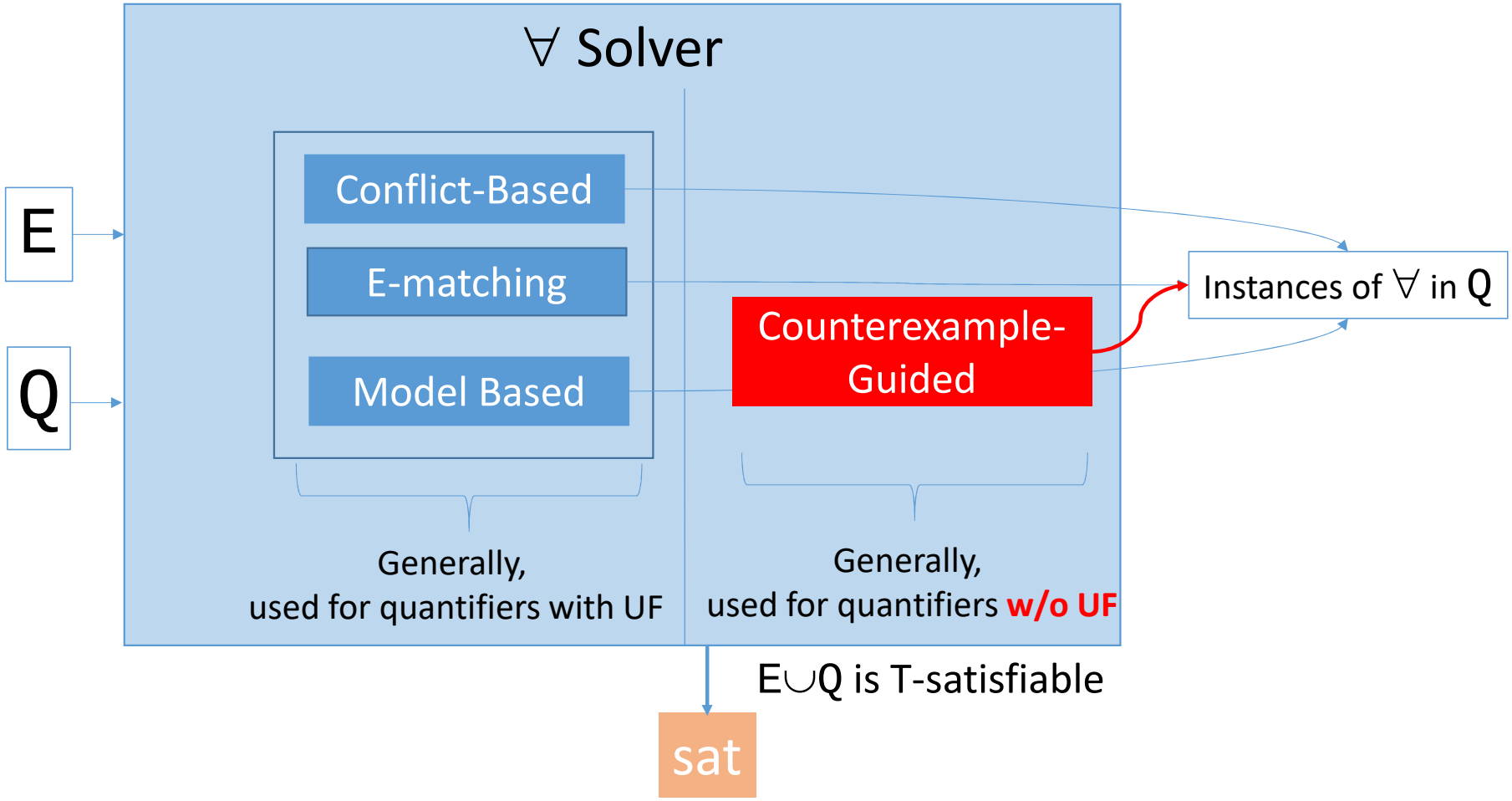
E-matching, Conflict-Based, Model-based:

- **Common thread:** satisfiability of $\forall$ + UF + theories is hard!
  - E-matching:
    - Pattern selection, matching modulo theories
  - Conflict-based:
    - Matching is incomplete, entailment tests are expensive
  - Model-based:
    - Models are complex, interpreted domains (e.g. Int) may be infinite

$\Rightarrow$ But reasoning about $\forall$ + theories without UF isn't as bad:

- Classic $\forall$-elimination algorithms are decision procedures for $\forall$ in:
  - LRA [Ferrante+Rackoff 79, Loos+Wiespfenning 93] , LIA [Cooper 72], datatypes, …
- Can classic $\forall$-elimination algorithms be leveraged in an DPLL(T) context?
  - Yes: [Monniaux 2010, Bjorner 2012, Reynolds et al 2015, Bjorner/Janota 2016]

Techniques for Quantifier Instantiation



Counterexample-Guided Instantiation

- Variants implemented in number of tools:
  - **Z3** [Bjorner 2012, Bjorner/Janota 2016]
  - Tools using Z3 as backend: **SPACER** [Komuravelli et al 2014] **UFO** [Fedyukovich et al 2016]
  - **Yices** [Dutertre 2015]
  - **CVC4** [Reynolds et al 2015]
  - **Boolector** [Preiner et al 2017]

Counterexample-Guided Instantiation

- High-level idea:
  - Quantifier elimination (e.g. for LIA) says:

$$\exists x \,.\, \psi[x] \Leftrightarrow \psi[t_1] \vee \ldots \vee \psi[t_n] \text{ for finite } n$$

Counterexample-Guided Instantiation

- High-level idea:
  - Quantifier elimination (e.g. for LIA) says:

$$?\mathbf{x}\,.\,\dot{E}y\,[\mathbf{x}] \;\; \dot{e} \;\; \dot{E}y\,[\mathbf{t_1}]\;\dot{e}\;\ldots\;\dot{e}\;\dot{E}y\,[\mathbf{t_n}] \text{ for finite } n$$

(consider the dual)

Counterexample-Guided Instantiation

- High-level idea:
  - Quantifier elimination (e.g. for LIA) says:

$$\forall x . \neg\psi[x] \Leftrightarrow \neg\psi[\mathbf{t_1}] \wedge \ldots \wedge \neg\psi[\mathbf{t_n}] \text{ for finite } n$$

  - Enumerate these instances via a counterexample-guided loop, that is:
    - Terminating: enumerate at most n instances
    - Efficient in practice: typically terminates after <<n instances

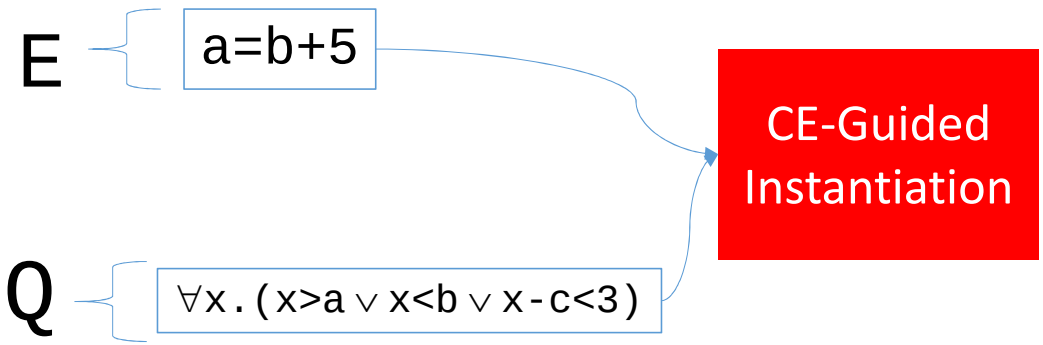
Counterexample-Guided Instantiation

E $\{$ $a=b+5$

Q $\{$ $\forall x.(x>a \vee x<b \vee x-c<3)$

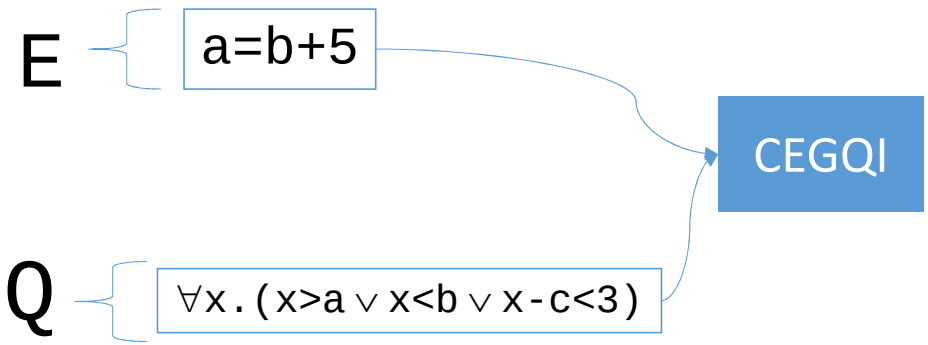
E, Q contain no uninterpreted functions, only linear arithmetic symbols

Counterexample-Guided Instantiation



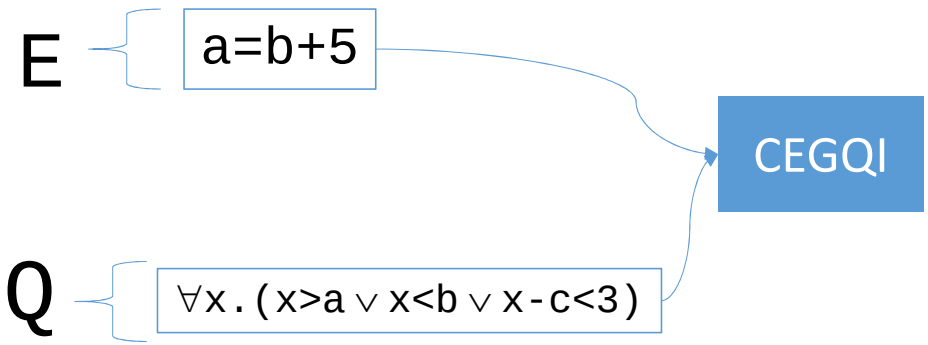
⇒Use *counterexample-guided instantiation*

Counterexample-Guided Instantiation



⇒Use *counterexample-guided instantiation*

Counterexample-Guided Instantiation



⇒With respect to *model-based instantiation*:

- Similar: based on finding models for Q's negation

Counterexample-Guided Instantiation

E

$a=b+5$

Extend context to include counterexample

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

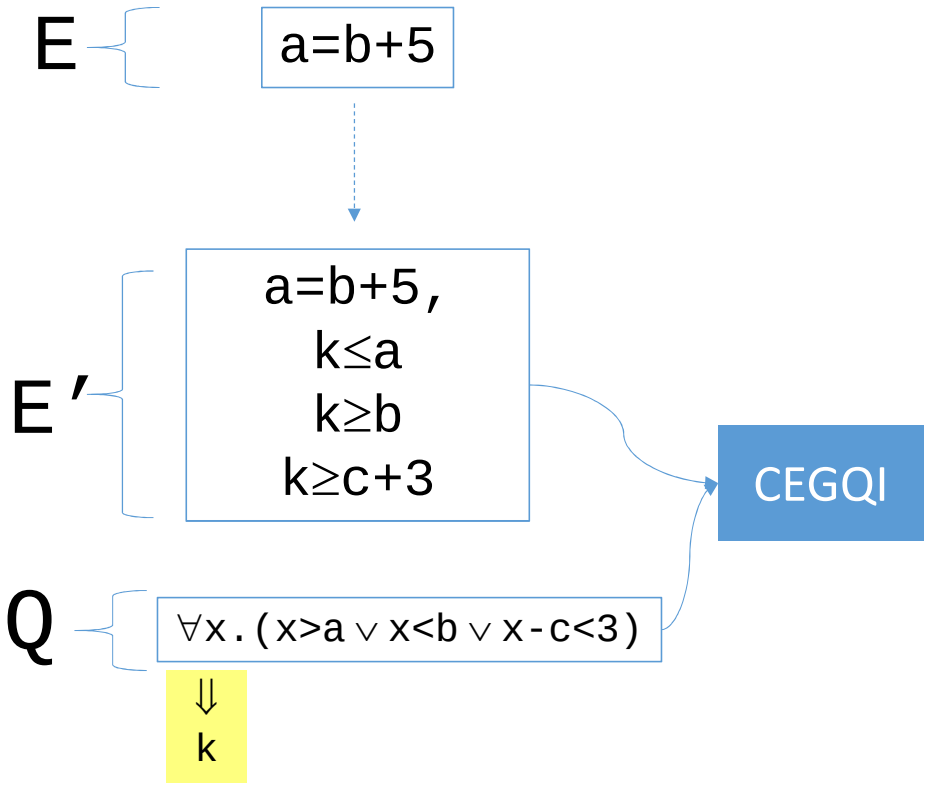
CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

…has counterexample **k** iff $\exists k.\neg(k>a \vee k<b \vee k-c<3)$

Counterexample-Guided Instantiation



One of two cases…

Counterexample-Guided Instantiation

E

$a=b+5$

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

sat

1 If E′ is T-unsatisfiable,
all models of E also satisfy Q

(since $E' \equiv E \cup \neg Q \models_T \bot$ implies $E \models_T Q$)

Counterexample-Guided Instantiation

E

$a=b+5$

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

sat

2 Return an instance based on model for E′

1 If E′ is T-unsatisfiable, all models of E also satisfy Q

(since $E' \equiv E \cup \neg Q \models_T \bot$ implies $E \models_T Q$)

Counterexample-Guided Instantiation

E

$a=b+5$

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

Counterexample-Guided Instantiation

E

$a=b+5$

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathscr{M}}=5$

$b^{\mathscr{M}}=0$

$c^{\mathscr{M}}=0$

$k^{\mathscr{M}}=3$

Build model $\mathscr{M}$ for E′

Counterexample-Guided Instantiation

E

a=b+5

E′

a=b+5,
k≤a
k≉b
k≉c+3

Q

∀x.(x>a ∨ x<b ∨ x-c<3)

⇓
k

CEGQI

$a^{\mathcal{M}}=5$

$b^{\mathcal{M}}=0$

$c^{\mathcal{M}}=0$

$k^{\mathcal{M}}=3$

k≉b
k≉c+3

Take lower bounds of k in E′

Counterexample-Guided Instantiation

E

$a=b+5$

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathcal{M}}=5$

$b^{\mathcal{M}}=0$

$c^{\mathcal{M}}=0$

$k^{\mathcal{M}}=3$

in $\mathcal{M}$

$k \geq b$	$=0$
$k \geq c+3$	$=3$

Compute their value in $\mathcal{M}$

Counterexample-Guided Instantiation

E

$a=b+5$

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathscr{M}}=5$

$b^{\mathscr{M}}=0$

$c^{\mathscr{M}}=0$

$k^{\mathscr{M}}=3$

in $\mathscr{M}$

$k \geq b$	$=0$
$k \geq \mathbf{c+3}$	$=\mathbf{3}$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$\mathbf{c+3}>a \vee \mathbf{c+3}<b \vee \mathbf{c+3}-c<3$

Add instance for lower bound that is maximal in $\mathscr{M}$

Counterexample-Guided Instantiation

E

$a=b+5$

E′

$a=b+5,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathcal{M}}=5$

$b^{\mathcal{M}}=0$

$c^{\mathcal{M}}=0$

$k^{\mathcal{M}}=3$

in $\mathcal{M}$

$k \geq b$	$=0$
$k \geq c+3$	$=3$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$c+3>a$ Ę $c+3<b$

Simplify

Counterexample-Guided Instantiation

E

$a=b+5, \mathbf{c+3<b}$

QF Solver

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

CEGQI

$a^{\mathcal{M}}=5$

$b^{\mathcal{M}}=0$

$c^{\mathcal{M}}=0$

$k^{\mathcal{M}}=3$

in $\mathcal{M}$

$k \geq b$	$=0$
$k \geq c+3$	$=3$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$c+3>a \vee \mathbf{c+3<b}$

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

Extend context to include counterexample

E′

$a=b+5, c+3<b,$
káa
kñb
kñc+3

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

ĝ
k

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

E′

$a=b+5, c+3<b,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathscr{M}}=5$

$b^{\mathscr{M}}=0$

$c^{\mathscr{M}}=-4$

$k^{\mathscr{M}}=3$

Build model $\mathscr{M}$ for E′

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

E′

$a=b+5, c+3<b,$
$k \leq a$
$k \tilde{n} b$
$k \tilde{n} c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathcal{M}}=5$

$b^{\mathcal{M}}=0$

$c^{\mathcal{M}}=-4$

$k^{\mathcal{M}}=3$

$k \tilde{n} b$
$k \tilde{n} c+3$

Take lower bounds of k in E′

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

E′

$a=b+5, c+3<b,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathscr{M}}=5$

$b^{\mathscr{M}}=0$

$c^{\mathscr{M}}=-4$

$k^{\mathscr{M}}=3$

in $\mathscr{M}$

$k \geq b$	$=0$
$k \geq c+3$	$=-1$

Compute their value in $\mathscr{M}$

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

E′

$a=b+5, c+3<b,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathscr{M}}=5$

$b^{\mathscr{M}}=0$

$c^{\mathscr{M}}=-4$

$k^{\mathscr{M}}=3$

in $\mathscr{M}$

$k \geq \mathbf{b}$	$=\mathbf{0}$
$k \geq c+3$	$=-1$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$\mathbf{b}>a \vee \mathbf{b}<b \vee \mathbf{b}-c<3$

Add instance for lower bound that is maximal in $\mathscr{M}$

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

E′

$a=b+5, c+3<b,$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$a^{\mathscr{M}}=5$

$b^{\mathscr{M}}=0$

$c^{\mathscr{M}}=-4$

$k^{\mathscr{M}}=3$

in $\mathscr{M}$

$k \geq b$	$=0$
$k \geq c+3$	$=-1$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$b>a$ Ę $b<c+3$

Simplify

Counterexample-Guided Instantiation

E

$a=b+5, c+3<b$

QF Solver

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$b>a$ Ę $b<c+3$

Counterexample-Guided Instantiation

E

$a=b+5, \underline{c+3<b}$

QF Solver

Backtrack previous decision $c+3>a \vee \underline{c+3<b}$

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$b>a \vee b<c+3$

Counterexample-Guided Instantiation

E

$a=b+5, \underline{c+3>a}$

QF Solver

Backtrack previous decision $\underline{c+3>a} \vee c+3<b$

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$b>a \vee b<c+3$

Counterexample-Guided Instantiation

E

$a=b+5,\ c+3>a,$
$b<c+3$

QF Solver

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$b>a$ Ę $b<c+3$

Counterexample-Guided Instantiation

E

$a=b+5,\ c+3>a,\ b<c+3$

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

Counterexample-Guided Instantiation

E

```
a=b+5, c+3>a,
b<c+3
```

Extend context to include counterexample

E′

```
a=b+5, c+3>a,
b<c+3
```

káa
kñb
kñc+3

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

ĝ
k

Counterexample-Guided Instantiation

E

$a=b+5,\ c+3>a,\ b<c+3$

E′

$a=b+5,\ c+3>a,\ b<c+3$

$k \leq a$

$k \geq b$

$k \geq c+3$

CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$

k

Counterexample-Guided Instantiation

E

$a=b+5,\ c+3>a,\ b<c+3$

E′

$a=b+5,\ c+3>a,\ b<c+3$

$k \leq a$

$k \geq b$

$k \geq c+3$

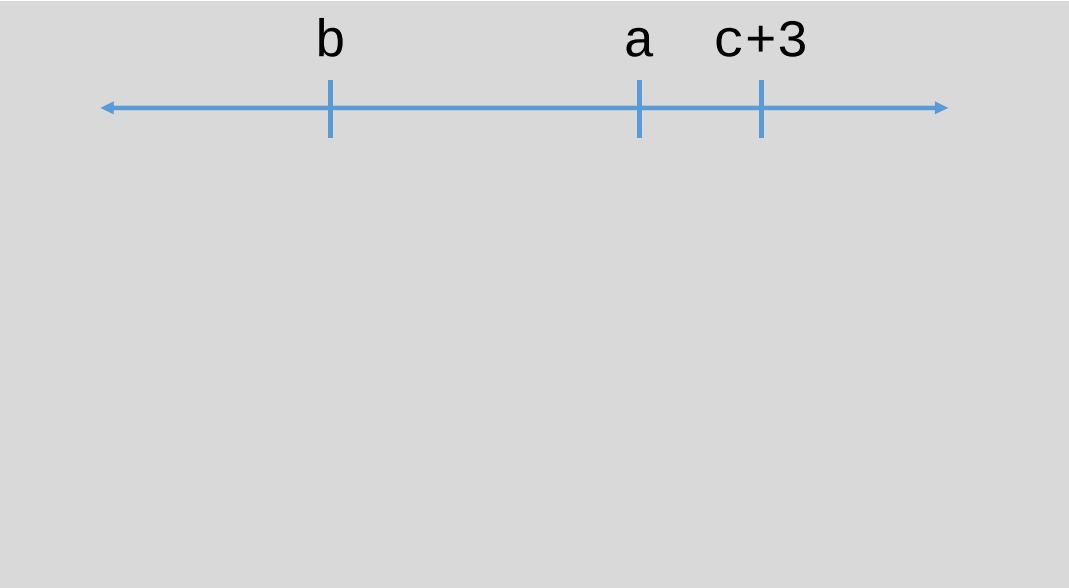
CEGQI

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$

k



Counterexample-Guided Instantiation

E

a=b+5, c+3>a,
b<c+3

E′

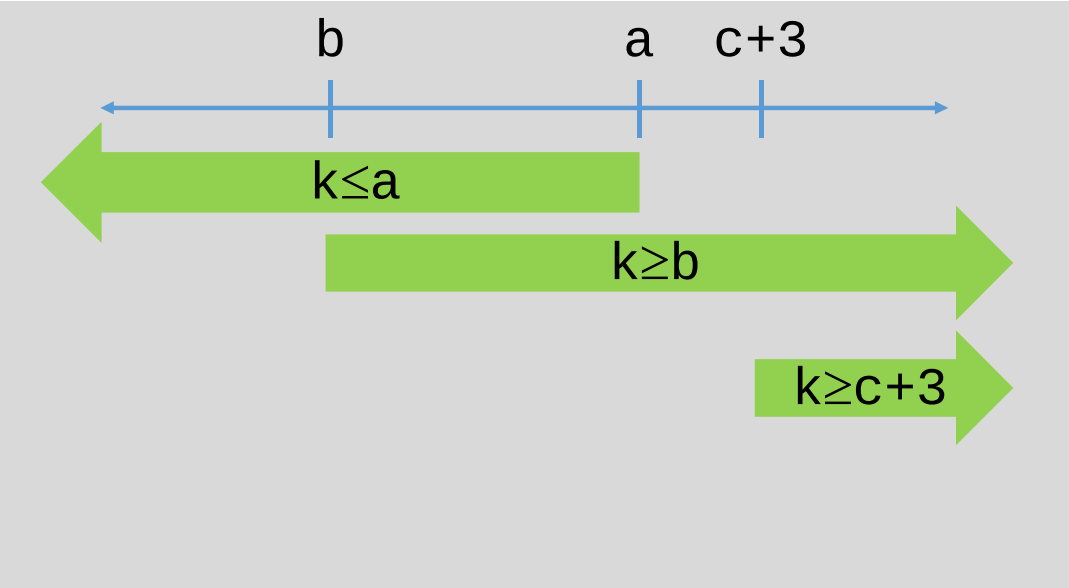
a=b+5, c+3>a,
b<c+3
káa
kñb
kñc+3

Q

∀x.(x>a ∨ x<b ∨ x-c<3)

⇓
k

CEGQI



Counterexample-Guided Instantiation

E

$a=b+5,\ c+3>a,\ b<c+3$

E′

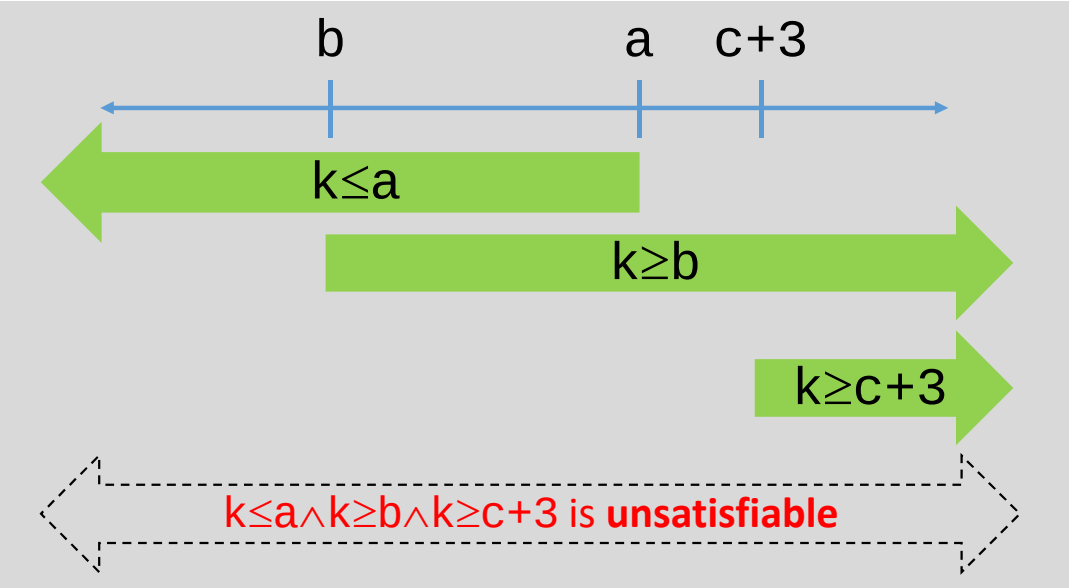
$a=b+5,\ c+3>a,\ b<c+3$
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI



Counterexample-Guided Instantiation

E

$a=b+5,\ c+3>a,\ b<c+3$

E′

$a=b+5,\ c+3>a,\ b<c+3,\ k\le a,\ k\ge b,\ k\ge c+3$

Q

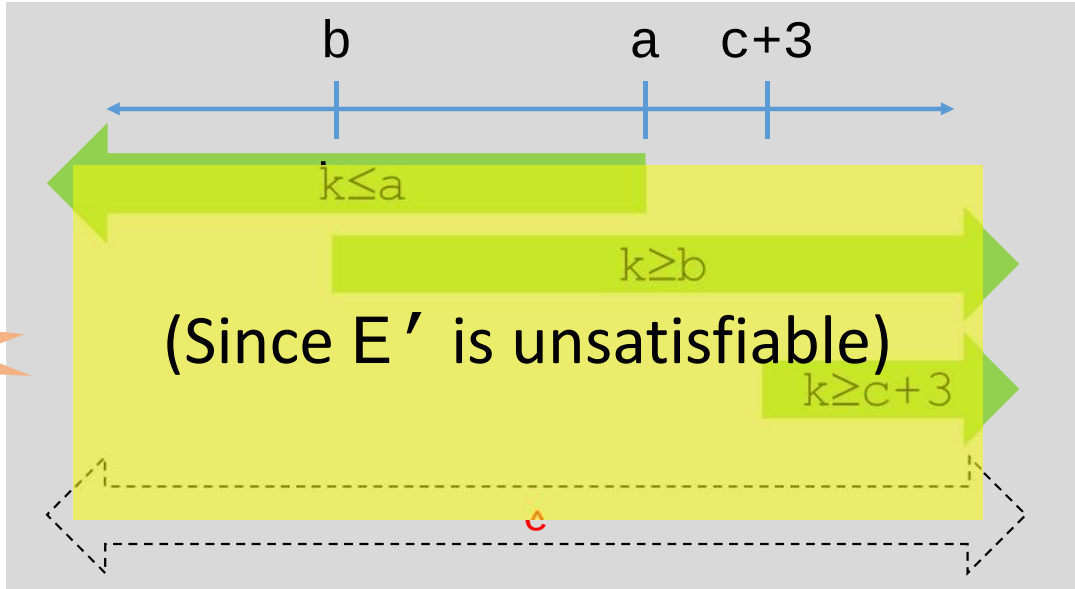
$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$

k

CEGQI

sat



Counterexample-Guided Instantiation

E

a=b+5, c+3>a,
b<c+3

E′

a=b+5, c+3>a,
b<c+3
k≤a
k≥b
k≥c+3

Q

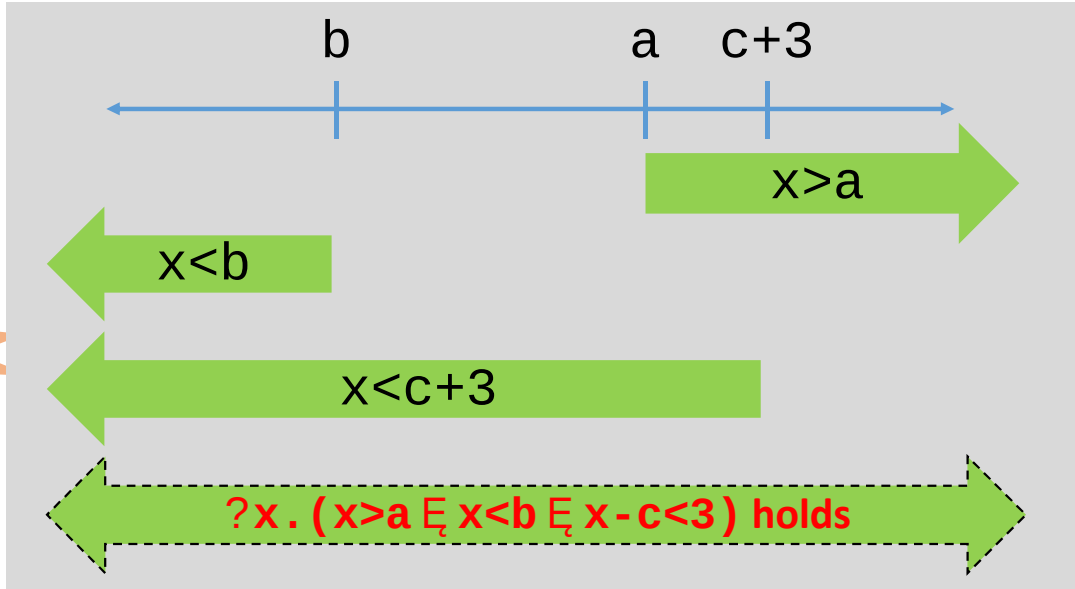
?x.(x>a Ę x<b Ę x-c<3)

⇓
k

CEGQI

sat

Dually:



Summary

E { ...

Q { $\forall x.(x>a \vee x<b \vee x-c<3)$ → CEGQI

Summary

E

. . .

Extend context to include counterexample

E′

. . .
$k \leq a$
$k \geq b$
$k \geq c+3$

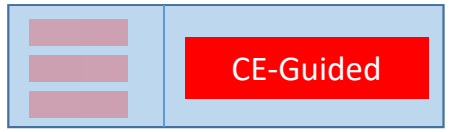
CEGQI

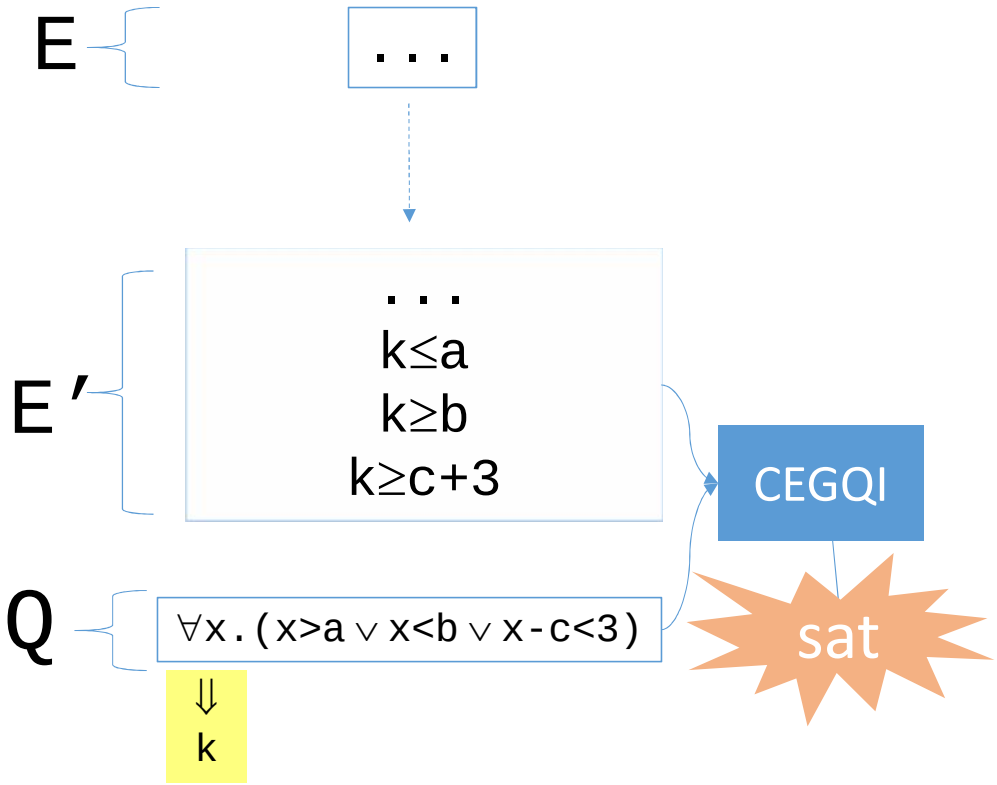
Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

Summary





…if E′ is unsatisfiable

Summary

E

. . .

E′

. . .
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=…

$b^{\mathscr{M}}$=…

$c^{\mathscr{M}}$=…

$k^{\mathscr{M}}$=…

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$t>a \vee t<b \vee t-c<3$

…select an instance to return otherwise

Summary

E

...

E′

...
$k \leq a$
$k \geq b$
$k \geq c+3$

Q

$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$
k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=...

$b^{\mathscr{M}}$=...

$c^{\mathscr{M}}$=...

$k^{\mathscr{M}}$=...

$\forall x.(x>a \vee x<b \vee x-c<3) \Rightarrow$
$t>a \vee t<b \vee t-c<3$

$\Rightarrow$ *In general:*

Requires a

selection function

$\mathtt{Sel}_T : E', \mathscr{M}, k \rightarrow t$

Selection Functions

- Selection function gives decision procedure for $\forall$ in various theories:
  - Linear real arithmetic (LRA)
    - Maximal lower (minimal upper) bounds $\quad l_1<k, \ldots, l_n<k \rightarrow \{x \rightarrow l_{max}+\delta\}$

      [Loos+Wiespfenning 93] $\quad$ *...may involve virtual terms* $d, \tilde{a}$
    - Interior point method: $\quad l_{max}<k<u_{min} \rightarrow \{x \rightarrow (l_{max}+u_{min})/2\}$

      [Ferrante+Rackoff 79]
  - Linear integer arithmetic (LIA)
    - Maximal lower (minimal upper) bounds (+c) $\quad l_1<k, \ldots, l_n<k \rightarrow \{x \rightarrow l_{max}+c\}$

      [Cooper 72]
  - Bitvectors/finite domains
    - Value instantiations $\quad \ldots \rightarrow \{x \rightarrow k^{\mathcal{M}}\}$
  - Datatypes, …

Ĝ ***Termination argument for each****: enumerate at most a finite number of instances*

Current Work

- In current work **[Reynolds/King/Kuncak FMSD 2017]**
  - Finite selection functions for LRA, LIA, LIRA
  - Extension to arbitrary quantifier alternations

  - Instantiation can be used for quantifier elimination:

    CEGQI terminates with $\psi[t_1] \wedge \ldots \wedge \psi[t_n] \Rightarrow$

    $\exists x . \neg\psi[x]$ is equivalent to $\neg\psi[t_1] \vee \ldots \vee \neg\psi[t_n]$

  - Instantiation can be used as basis of synthesis procedures:

    CEGQI finds $\psi[t_1] \wedge \ldots \wedge \psi[t_n]$ is unsat $\Rightarrow$

    $\lambda x . \mathtt{ite}(\psi[t_1], t_1, \ldots, \mathtt{ite}(\psi[t_{n-1}], t_{n-1}, t_n) \ldots)$ is a solution for f in $\forall x . \psi[f(x)]$

    $\Rightarrow$Used in CVC4's synthesis solver **[Reynolds et al CAV 2015]**

CEGQI for Bit-Vectors

E { empty

CEGQI

Q { ∀x.bvshl(x,#0001)≠bvand(a,#FFFE)

E, Q contain only bit-vector symbols

CEGQI for Bit-Vectors

E { empty

CEGQI

Q { ∀x.bvshl(x,#0001)≠bvand(a,#FFFE)

CEGQI for Bit-Vectors

E { empty

Extend context to include counterexample

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

CEGQI

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI for Bit-Vectors

E

empty

E ′

`bvshl(k,#0001)=bvand(a,#FFFE)`

Q

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

Build model $\mathscr{M}$ for E ′

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathcal{M}$

$a^{\mathcal{M}}$=#0000

$k^{\mathcal{M}}$=#0000

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒`
`bvshl(#0000,#0001)≠bvand(a,#FFFE)`

Add instance for value of $k^{\mathcal{M}}$

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒`
`#0000÷bvand(a,#FFFE)`

Simplify

CEGQI for Bit-Vectors

E

`#0000÷bvand(a,#FFFE)`

QF Solver

CEGQI

Q

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒`
`#0000÷bvand(a,#FFFE)`

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
```

CEGQI

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
```

Extend context to include counterexample

E’

```
#0000≠bvand(a,#FFFE)
bvshl(k,#0001)=bvand(a,#FFFE)
```

CEGQI

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

ĝ
k

CEGQI for Bit-Vectors

E

`#0000≠bvand(a,#FFFE)`

E′

`#0000≠bvand(a,#FFFE)`
`bvshl(k,#0001)=bvand(a,#FFFE)`

Q

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0002

$k^{\mathscr{M}}$=#0004

Build model $\mathscr{M}$ for E′

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
```

E′

```
#0000≠bvand(a,#FFFE)
bvshl(k,#0001)=bvand(a,#FFFE)
```

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

⇓
k

CEGQI

$\mathcal{M}$

$a^{\mathcal{M}}$=#0002

$k^{\mathcal{M}}$=#0004

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
bvshl(#0004,#0001)≠bvand(a,#FFFE)
```

Add instance for value of $k^{\mathcal{M}}$

CEGQI for Bit-Vectors

E

`#0000≠bvand(a,#FFFE)`

E′

`#0000≠bvand(a,#FFFE)`
`bvshl(k,#0001)=bvand(a,#FFFE)`

Q

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0002

$k^{\mathscr{M}}$=#0004

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒`
`#0002÷bvand(a,#FFFE)`

Simplify

CEGQI for Bit-Vectors

E

#0000≠bvand(a,#FFFE)
#0002÷bvand(a,#FFFE)

QF Solver

CEGQI

Q

∀x.bvshl(x,#0001)≠bvand(a,#FFFE)

∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
#0002÷bvand(a,#FFFE)

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
```

CEGQI

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
```

Extend context to include counterexample

E’

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
bvshl(k,#0001)=bvand(a,#FFFE)
```

CEGQI

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

ĝ
k

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
```

E′

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
bvshl(k,#0001)=bvand(a,#FFFE)
```

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

⇓
k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0004

$k^{\mathscr{M}}$=#0008

Build model $\mathscr{M}$ for E′

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
```

E′

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
bvshl(k,#0001)=bvand(a,#FFFE)
```

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

⇓
k

CEGQI

$\mathcal{M}$

$a^{\mathcal{M}}$=#0004

$k^{\mathcal{M}}$=#0008

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
bvshl(#0008,#0001)≠bvand(a,#FFFE)
```

Add instance for value of $k^{\mathcal{M}}$

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
```

E′

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
bvshl(k,#0001)=bvand(a,#FFFE)
```

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0004

$k^{\mathscr{M}}$=#0008

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
#0004÷bvand(a,#FFFE)
```

Simplify

CEGQI for Bit-Vectors

E

#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
#0004÷bvand(a,#FFFE)

QF Solver

CEGQI

Q

∀x.bvshl(x,#0001)≠bvand(a,#FFFE)

∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
#0004÷bvand(a,#FFFE)

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
#0004≠bvand(a,#FFFE)
```

CEGQI

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

CEGQI for Bit-Vectors

E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
#0004≠bvand(a,#FFFE)
#0006÷bvand(a,#FFFE)
#0008÷bvand(a,#FFFE)
#000A÷bvand(a,#FFFE)
...
#FFFE÷bvand(a,#FFFE)
```

QF Solver

...and repeat 2^{31} times

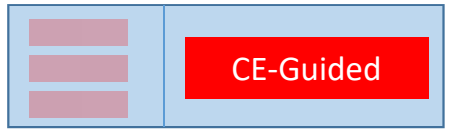
CEGQI

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
...
```

CEGQI for Bit-Vectors



E

```
#0000≠bvand(a,#FFFE)
#0002≠bvand(a,#FFFE)
#0004≠bvand(a,#FFFE)
#0006≠bvand(a,#FFFE)
#0008≠bvand(a,#FFFE)
#000A≠bvand(a,#FFFE)
...
#FFFE≠bvand(a,#FFFE)
```

unsat

QF Solver

...and repeat 2^{31} times

CEGQI

Q

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)
```

```
∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
...
```

CEGQI for Bit-Vectors

E { empty

But what if we had used a different selection function?

CEGQI

Q { ∀x.bvshl(x,#0001)≠bvand(a,#FFFE)

CEGQI for Bit-Vectors

E { empty

Extend context to include counterexample

E′ { **bvshl(k,#0001)=bvand(a,#FFFE)**

CEGQI

Q { ∀x.bvshl(x,#0001)≠bvand(a,#FFFE)

ĝ
k

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

Build model $\mathscr{M}$ for E′

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

$\mathscr{M} \models$ `bvshl(#0000,#0001)=bvand(a,#FFFE)`

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

$\mathscr{M} \models$ `bvshl(#0000,#0001)=bvand(a,#FFFE)`

Consider other terms whose value is **#0000** is in $\mathscr{M}$…

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

$\mathscr{M} \models$ `bvshl(#0000,#0001)=bvand(a,#FFFE)`

$a^{\mathscr{M}}$ = `bvmul(a,b)`$^{\mathscr{M}}$ = `bvshr(a,#0001)`$^{\mathscr{M}}$ = `bvand(a,#0001)`$^{\mathscr{M}}$ ...

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

$\mathscr{M} \models$ `bvshl(bvshr(a,#0001),#0001)=bvand(a,#FFFE)`

CEGQI for Bit-Vectors

E { empty

E′ { `bvshl(k,#0001)=bvand(a,#FFFE)`

Q { `∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

⇓

k

CEGQI

$\mathscr{M}$

$a^{\mathscr{M}}$=#0000

$k^{\mathscr{M}}$=#0000

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒`
`bvshl(bvshr(a,#0001),#0001)≠bvand(a,#FFFE)`

CEGQI for Bit-Vectors

E

bvshl(bvshr(a,#0001),#0001)÷bvand(a,#FFFE)

QF Solver

CEGQI

Q

∀x.bvshl(x,#0001)≠bvand(a,#FFFE)

∀x.bvshl(x,#0001)≠bvand(a,#FFFE)⇒
bvshl(bvshr(a,#0001),#0001)÷bvand(a,#FFFE)

CEGQI for Bit-Vectors

E

`bvshl(bvsh… 1)≠bvand(a,#FFFE)`

unsat

Ground Solver

⇒Single instance suffices to show this input is unsatisfiable

CEGQI

Q

`∀x.bvshl(x,#0001)≠bvand(a,#FFFE)`

…found using selection function $\mathtt{Sel}_T : E', \mathscr{M}, k \rightarrow t$
which returns terms whose interpretation is equal to k's in $\mathscr{M}$

CEGQI for Bit-Vectors

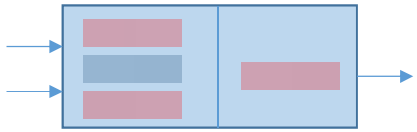
- Ongoing work: model-based selection functions for bit-vectors that:
  - Choose symbolic terms based on algebraic reasoning
    - E.g. many bit-vector operators are (partially) invertible:

```
bvadd(bvsub(a,n),n)=a
bvshl(bvshr(a,#0001),#0001)=a      when bvand(a,#FFFE)=#x0000
bvand(bvor(a,n),~n)=a              when bvand(a,n)=#x0000
```

- Related work on bit-vectors:
  - MBQI for ∀+BV **[Wintersteiger et al 2013]**
  - Incremental determinization for QBF **[Rabe et al SAT 2016]**
  - Syntax-guided synthesis for ∀+BV **[Preiner et al TACAS 2017]**

Summary

- Quantifier Instantiation Beyond E-matching, via:
  - **Conflict-based, Model-Based Instantiation**
    - Effective in practice for ∀ that highly involve UF
    - Conflict-Based is useful for “unsat”
    - Model-Based is useful for “sat”
  - **Counterexample-guided Instantiation**
    - Decision procedure for ∀+LRA, ∀+LIA, ∀+BV, …



Future Challenges

- How do we develop instantiations procedures for **?+new theories**?
  - Ongoing work on ∀+bit-vectors
  - Also interested in ∀+strings, ∀+floating-points, ∀+finite sets, etc.

- How do we **combine** instantiation procedures for **?+UF+theories**?
  - E.g. we have E-matching for **?+UF**, counterexample-guided for **?+ theories**
    …to what extent can these techniques be combined?

- How can we leverage first-order instantiation for **higher-order** problems?
  - E.g. synthesis conjectures, higher-order theorem proving

- Techniques in this talk implemented in CVC4
  - Open source
  - Available at : http://cvc4.cs.stanford.edu/web/

- …Thanks for listening!