Quantifiers in SMT: A View from Inside the Solver

Andrew Reynolds

SRI seminar

July 9, 2020

Overview

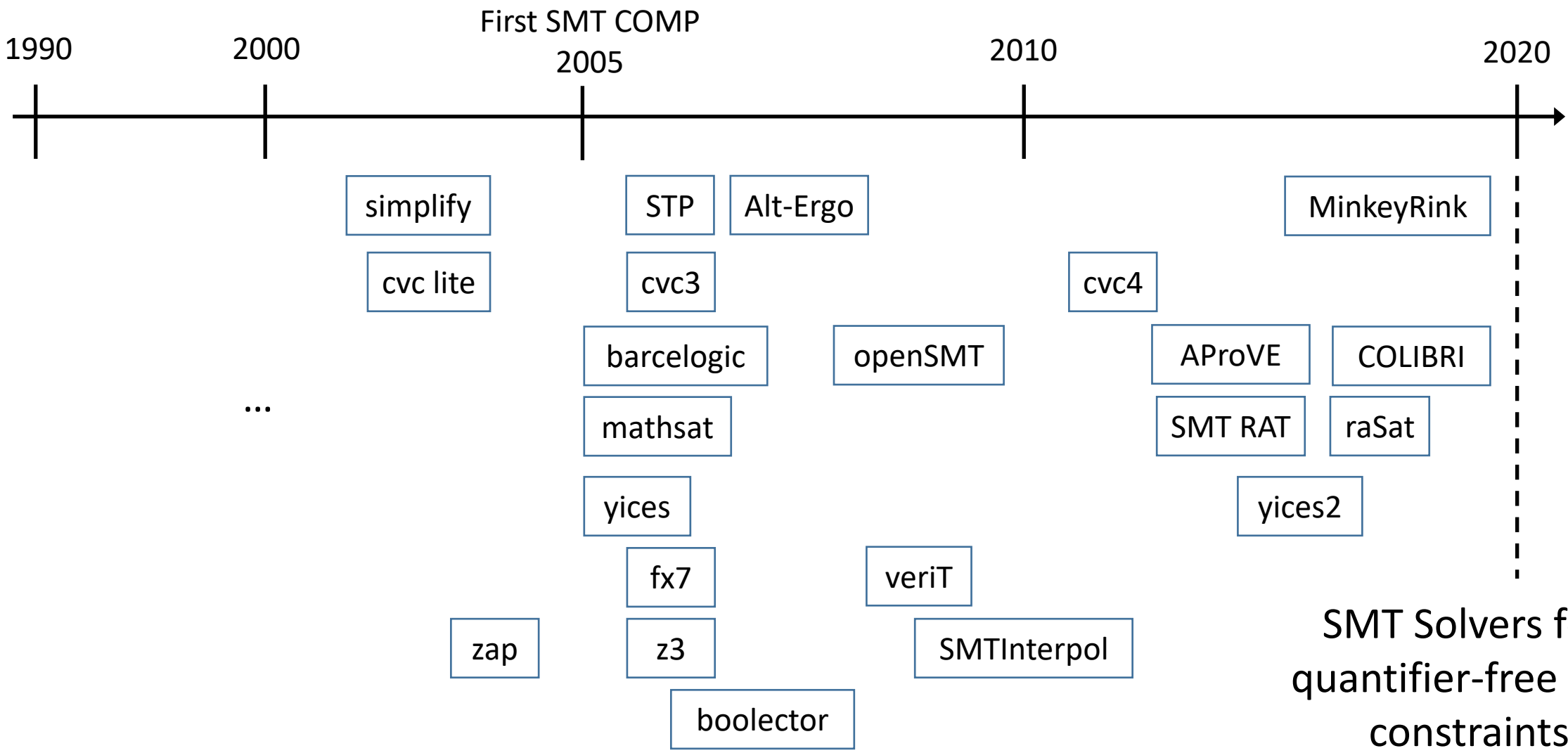
- Landscape of tools and approaches for handling quantifiers
  - ATP vs SMT, UF vs theories

$\Rightarrow$ ***Quantifier Instantiation in SMT***

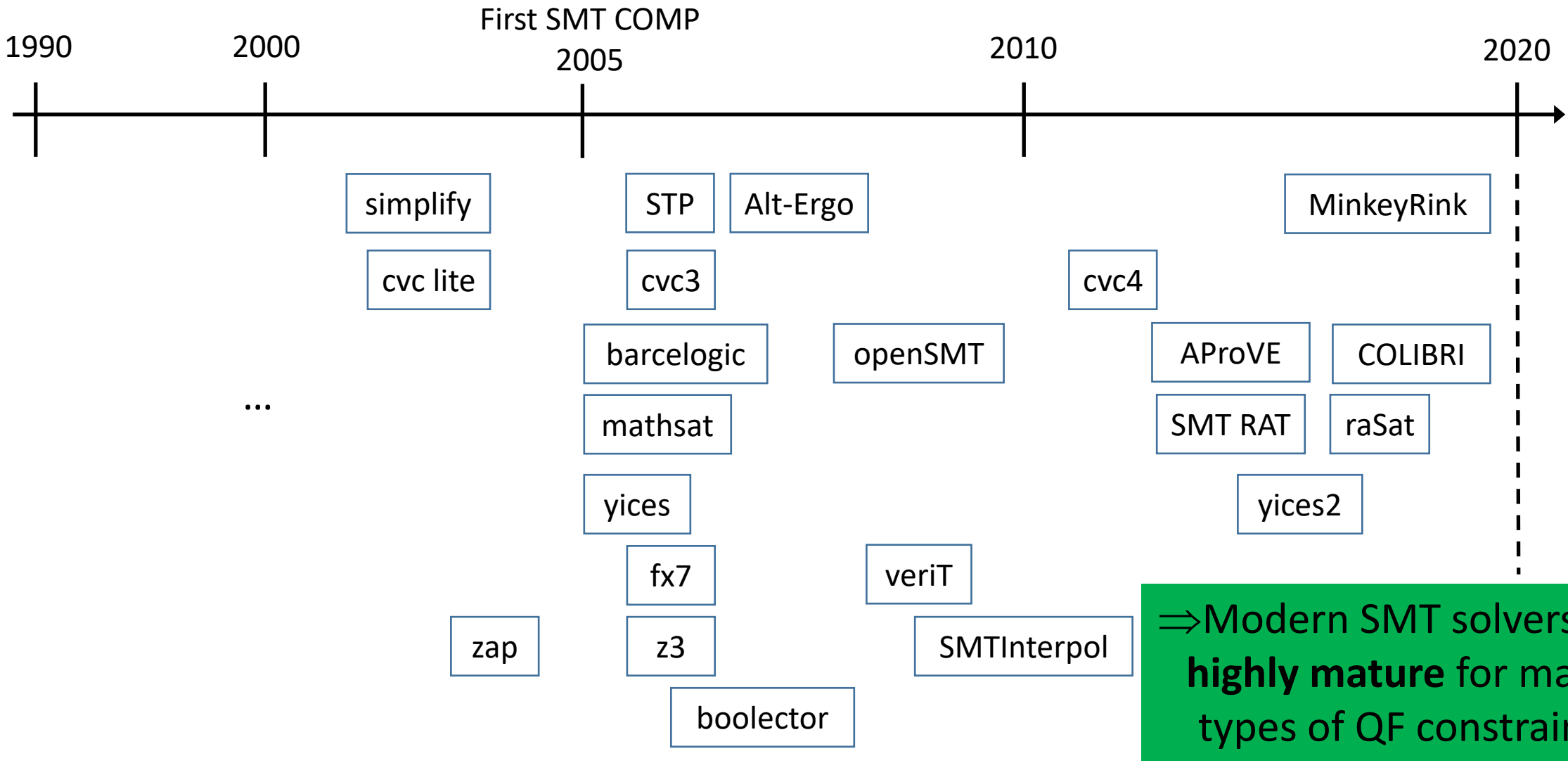
  - E-matching, Model-based, Conflict-based, Enumerative

- Advanced Topics
  - Counterexample-guided instantiation for $\forall$+theories

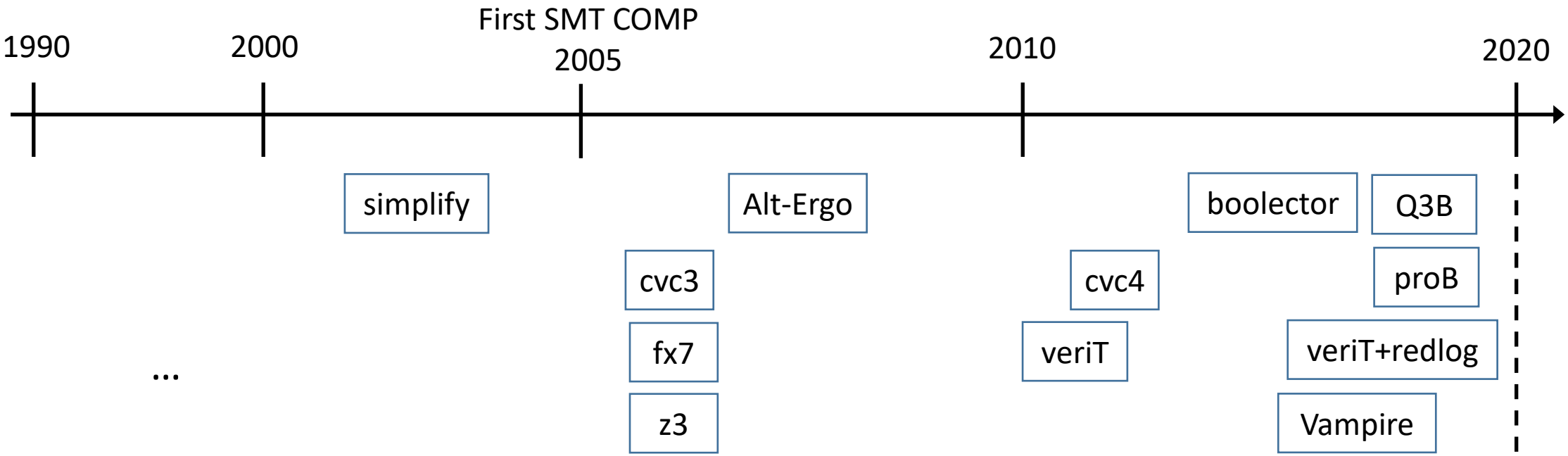
Timeline of Modern SMT Solvers for QF (Approximate)



Timeline of Modern SMT Solvers for QF (Approximate)

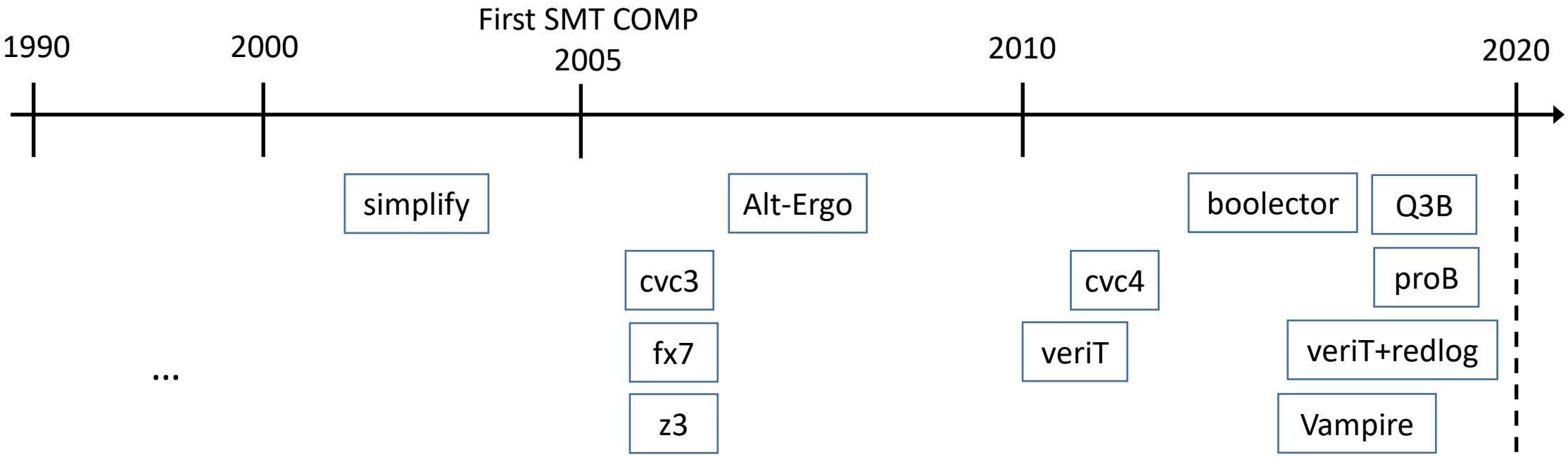


Timeline of Modern SMT Solvers for ∀ (Approximate)



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

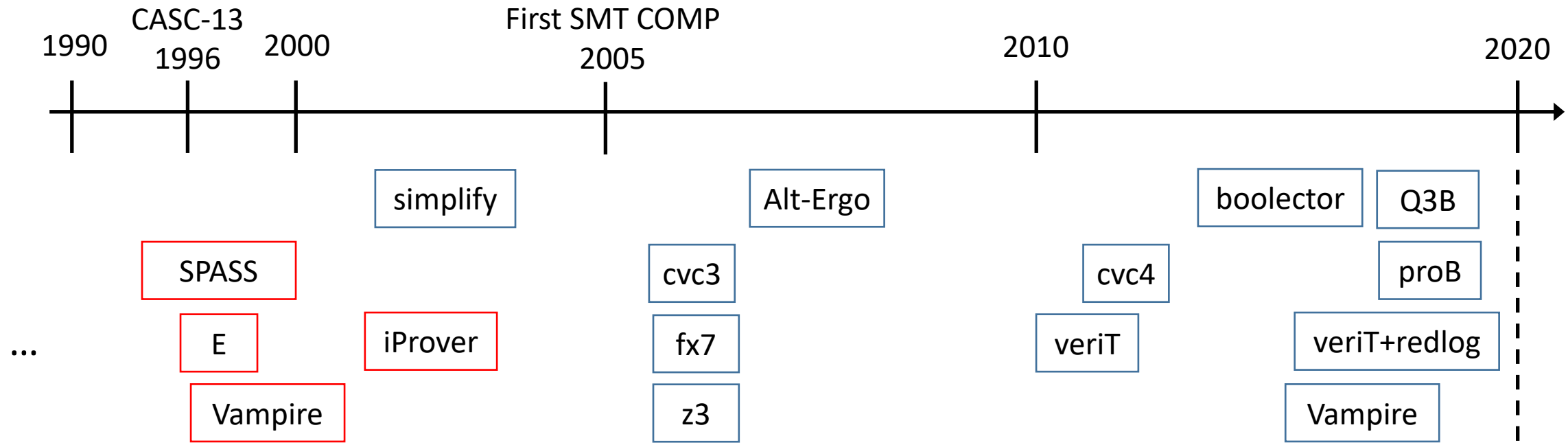
Timeline of Modern SMT Solvers for ∀ (Approximate)



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

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

Timeline of Modern SMT Solvers for ∀ (Approximate)



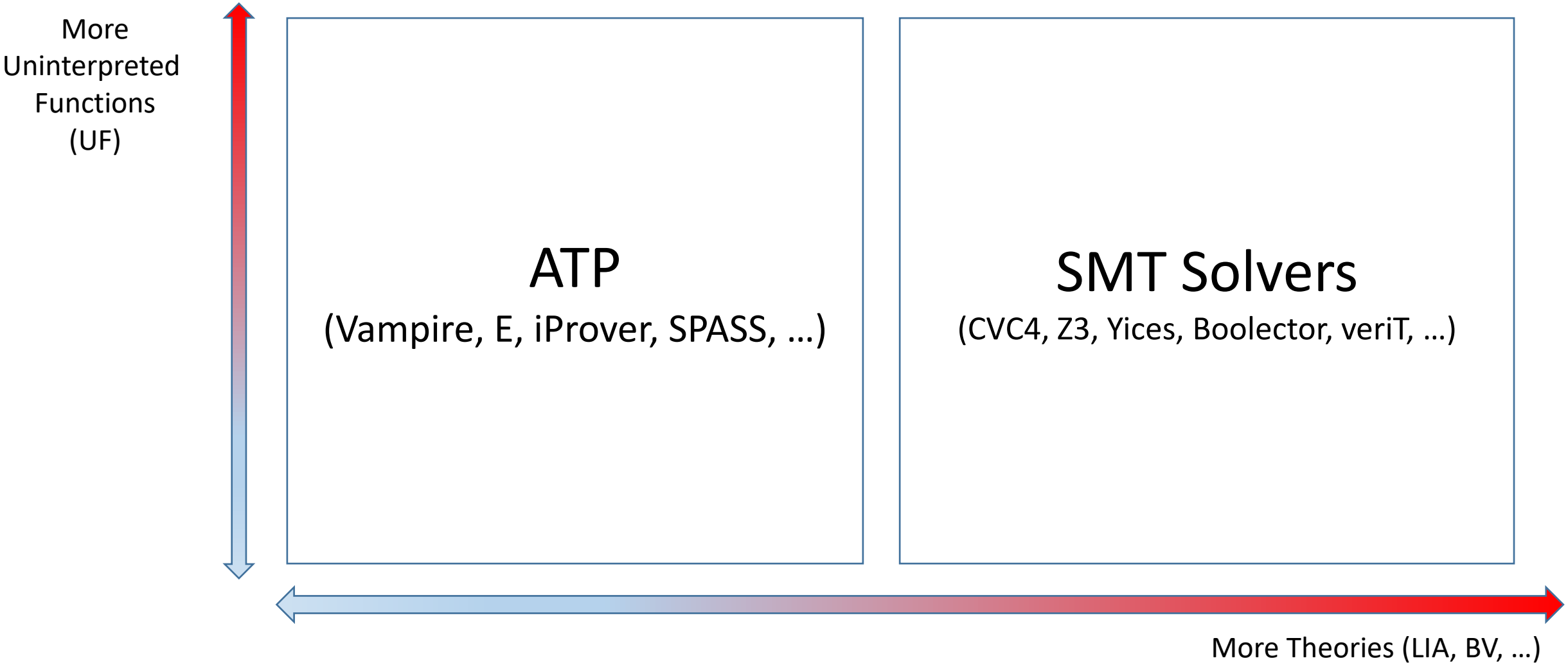
⇒ Build on earlier work from ATP community for ∀-constraints

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

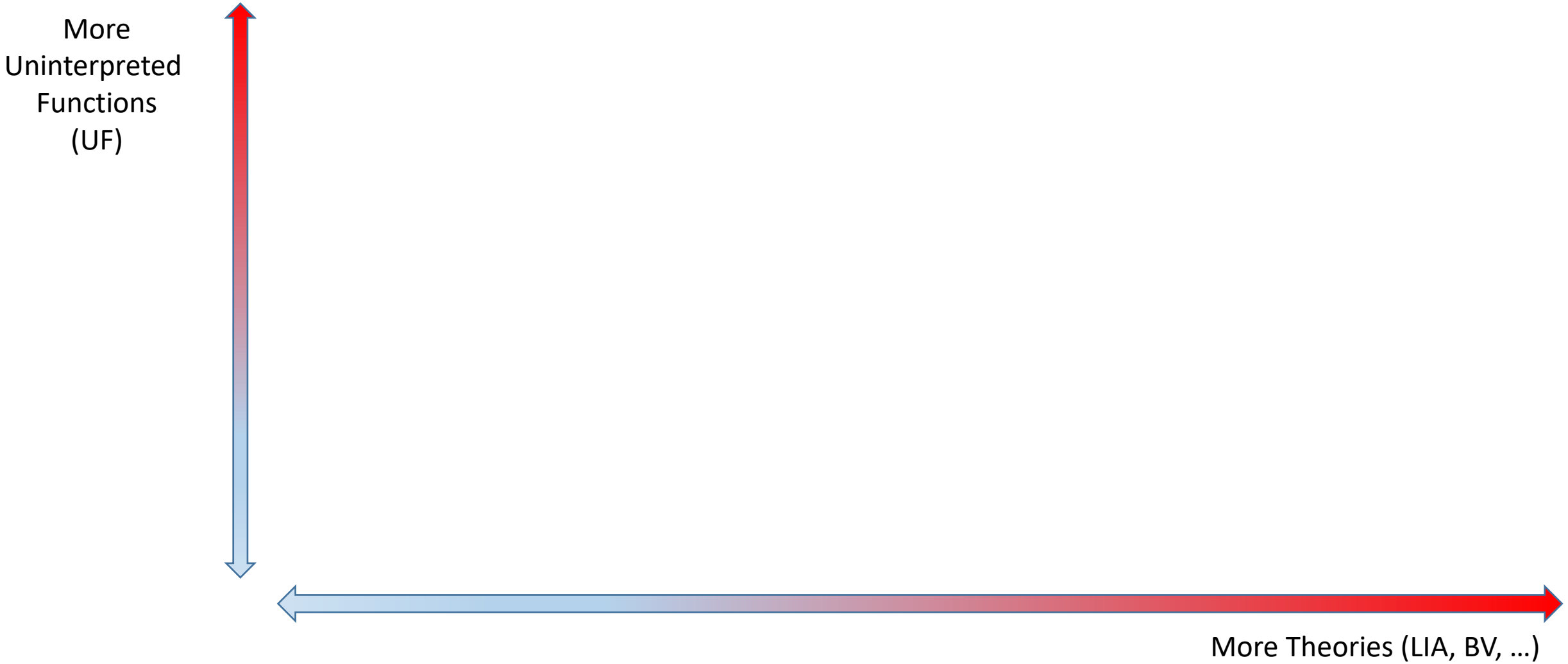
Applications of ∀ in SMT

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

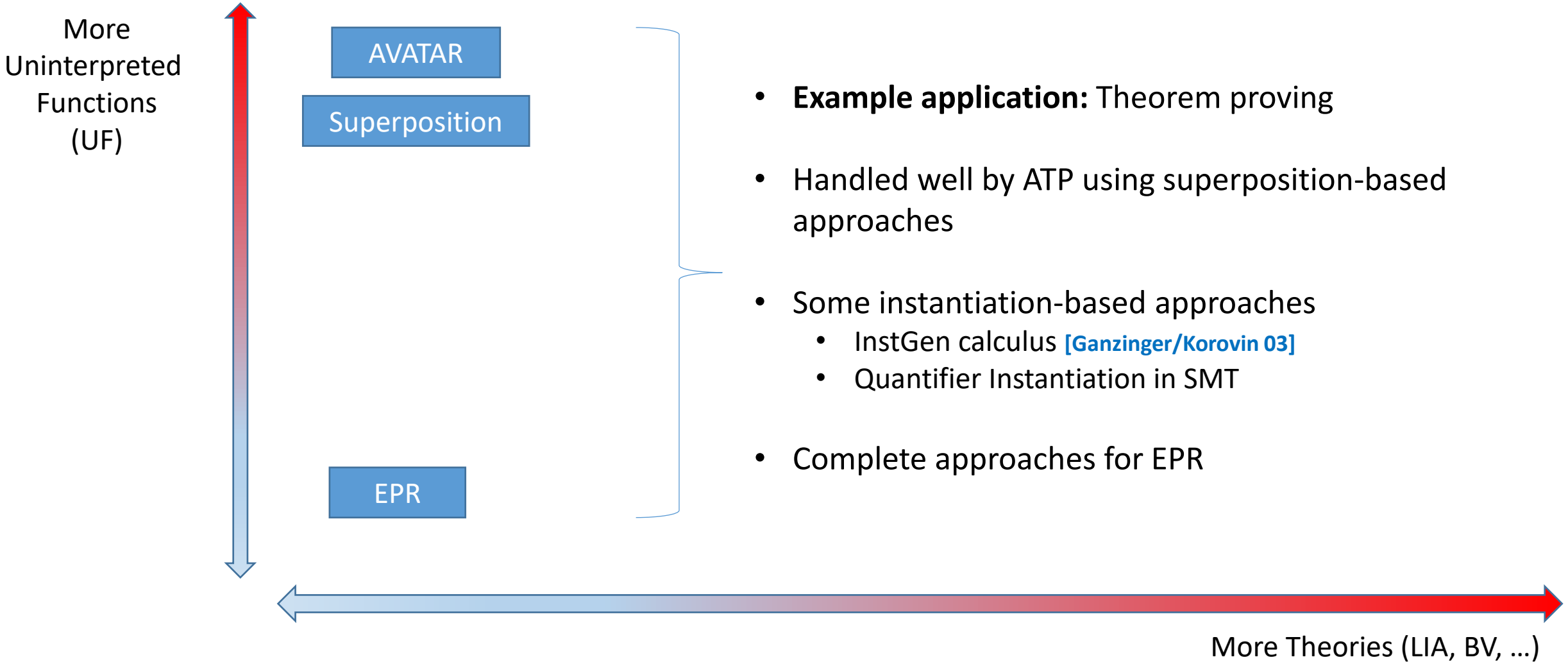
Landscape of Quantified Constraints in SMT



Landscape of Quantified Constraints in SMT

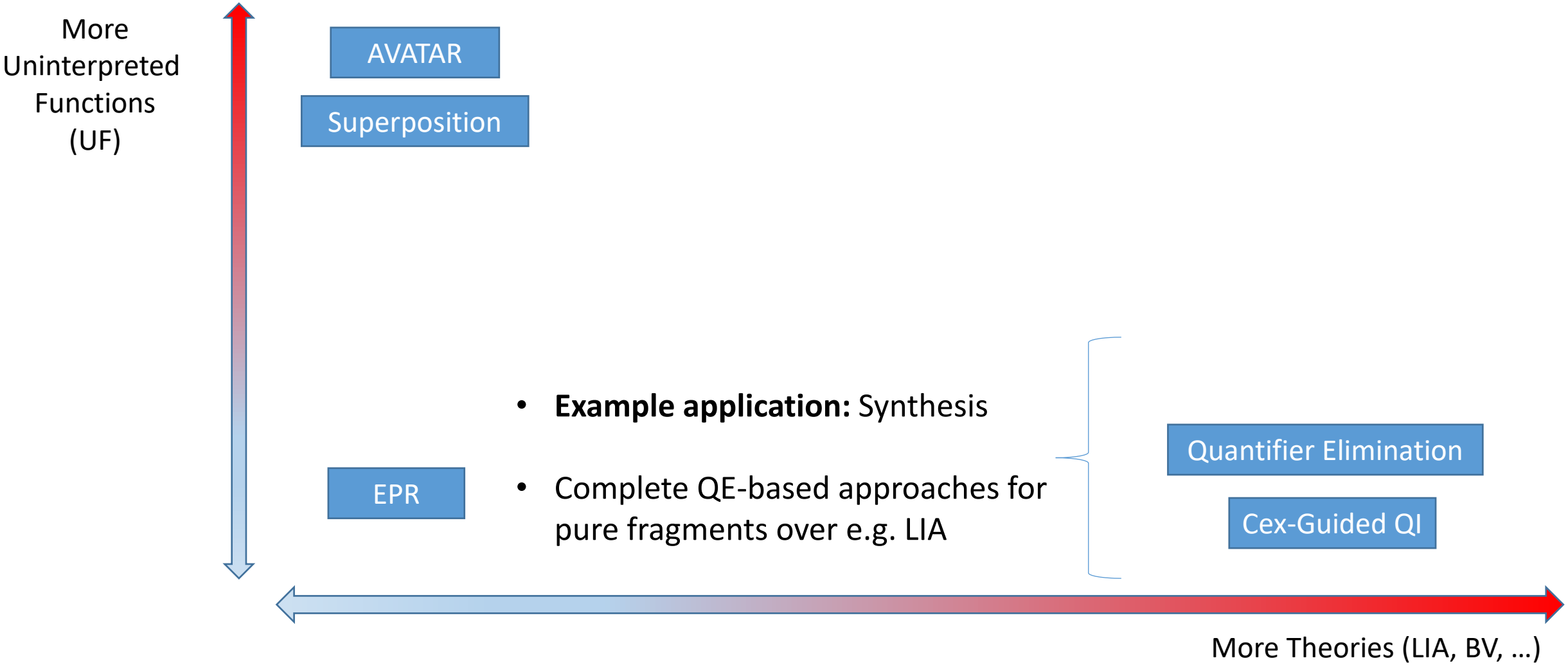


Landscape of Quantified Constraints in SMT

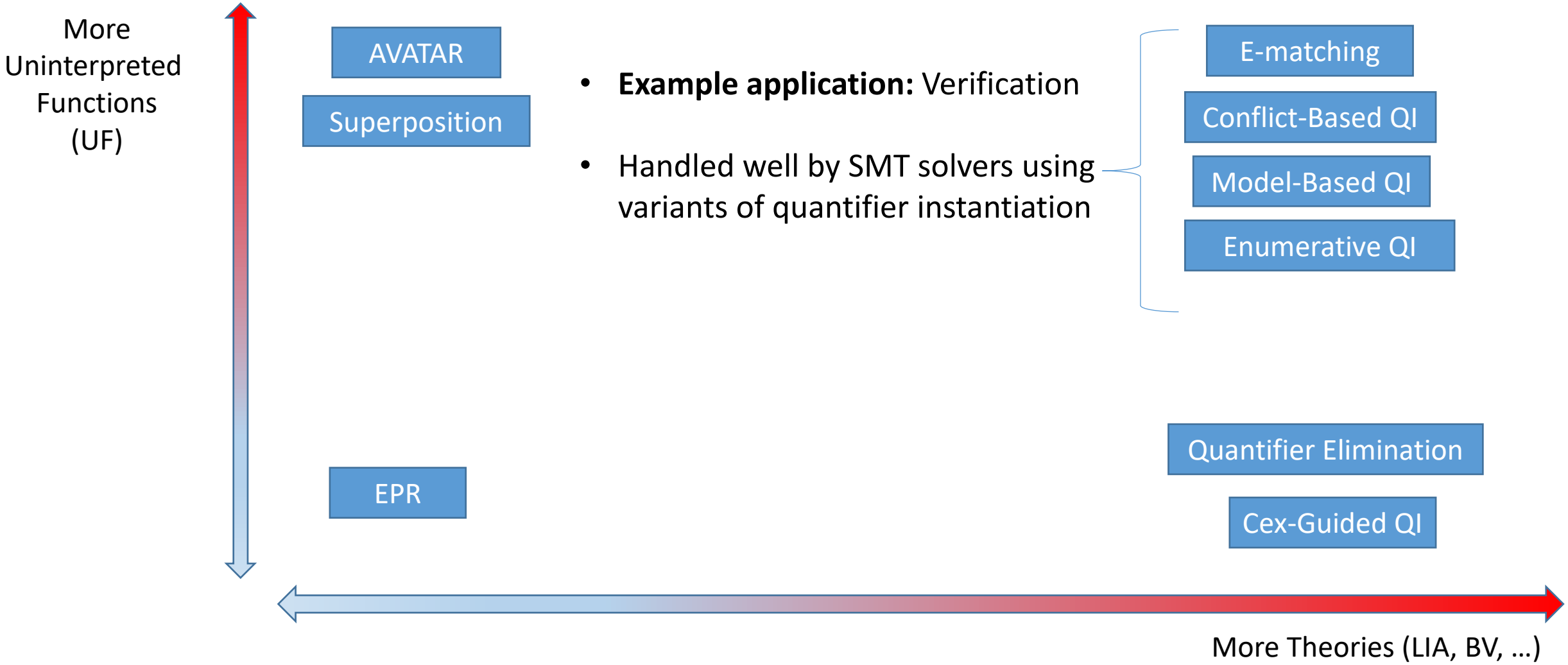


- **Example application:** Theorem proving
- Handled well by ATP using superposition-based approaches
- Some instantiation-based approaches
  - InstGen calculus [Ganzinger/Korovin 03]
  - Quantifier Instantiation in SMT
- Complete approaches for EPR

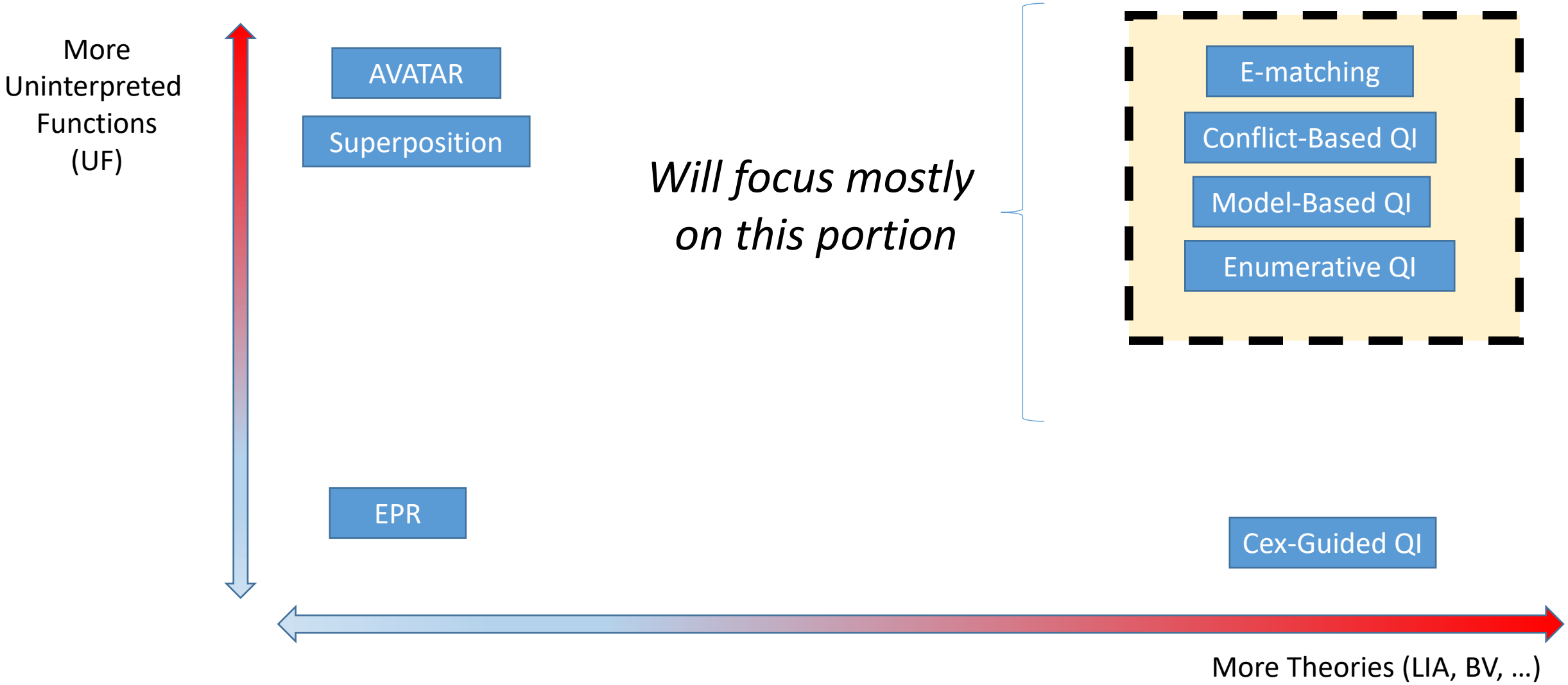
Landscape of Quantified Constraints in SMT



Landscape of Quantified Constraints in SMT



Landscape of Quantified Constraints in SMT



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, SMTInterpol

SMT Solvers for ∀ **using Quantifier Instantiation**

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

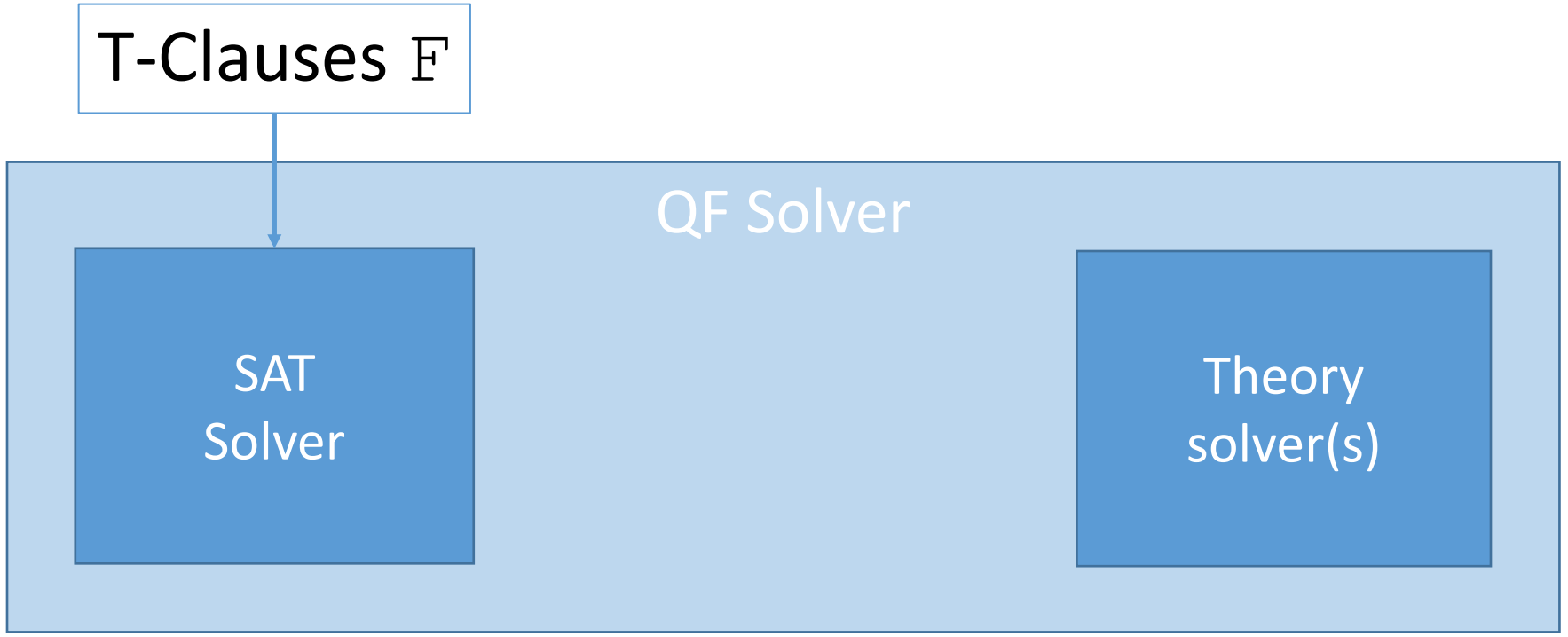
Implemented in

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

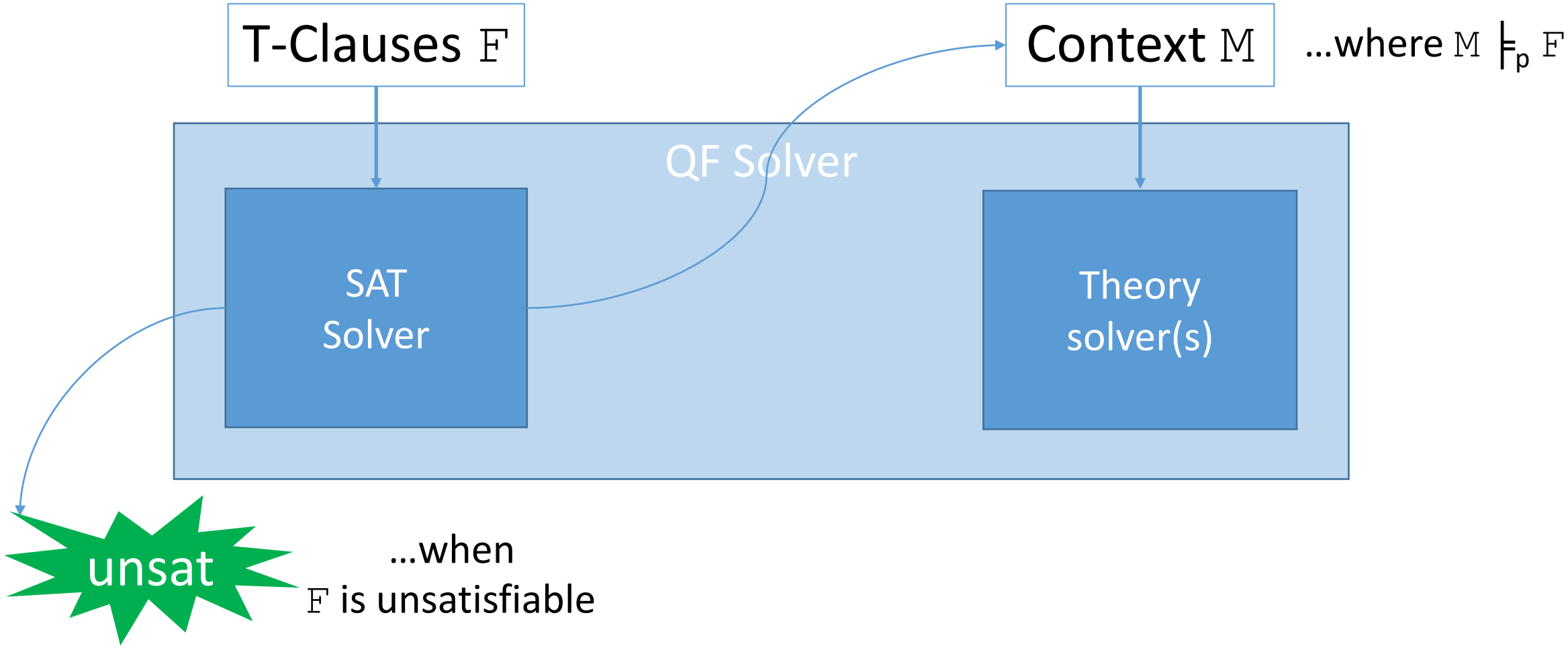
- More recently:
  - Model-Based Instantiation **[Ge et al 2009, Reynolds et al 2013]**
  - Conflict-Based Instantiation **[Reynolds et al 2014, Barbosa et al 2017]**
  - Enumerative Instantiation **[Reynolds et al 2018]**

z3, cvc4
cvc4, veriT, SMTInterpol
cvc4, veriT

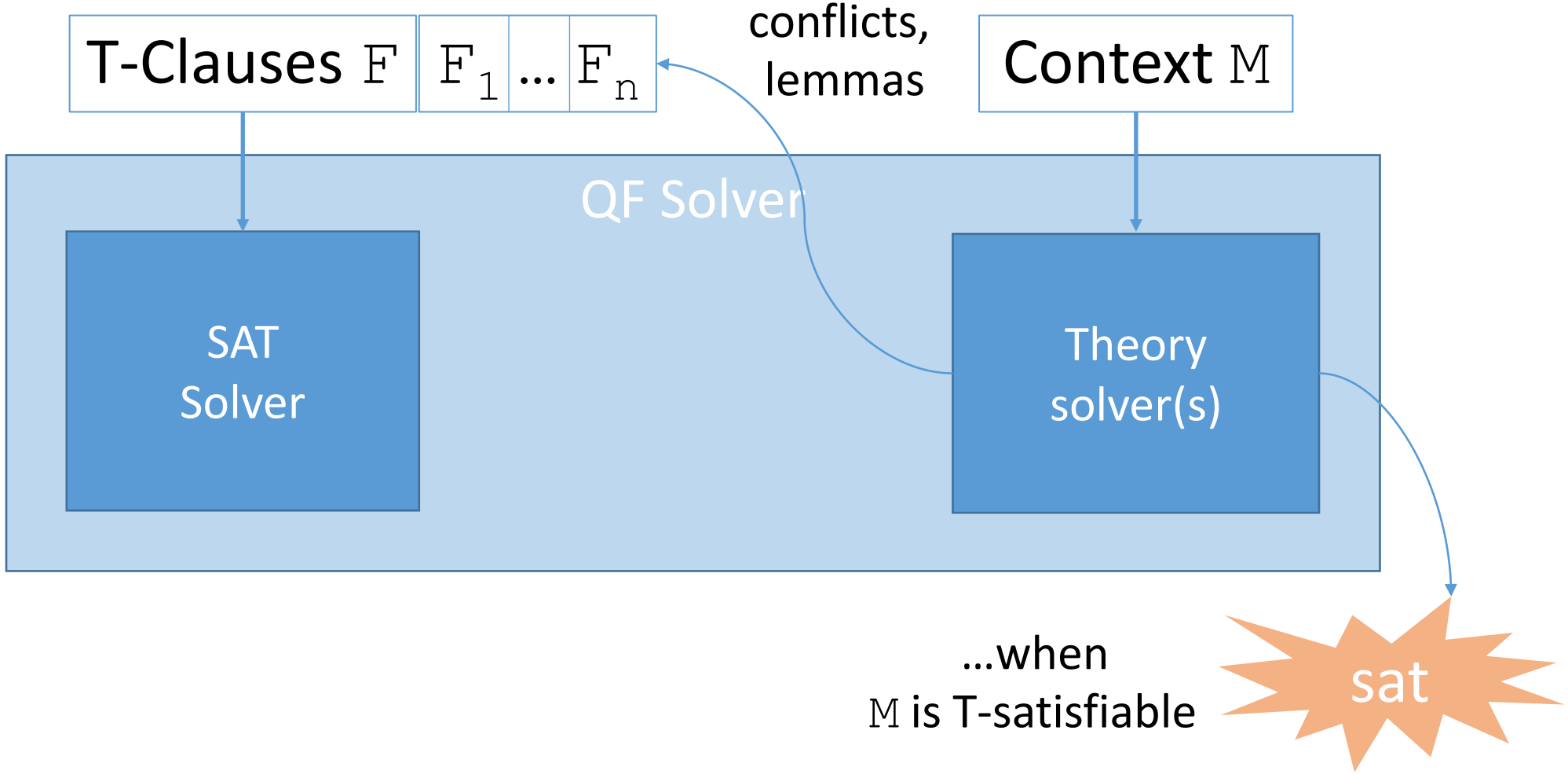
DPLL(T)-Based SMT Solvers (quantifier-free)



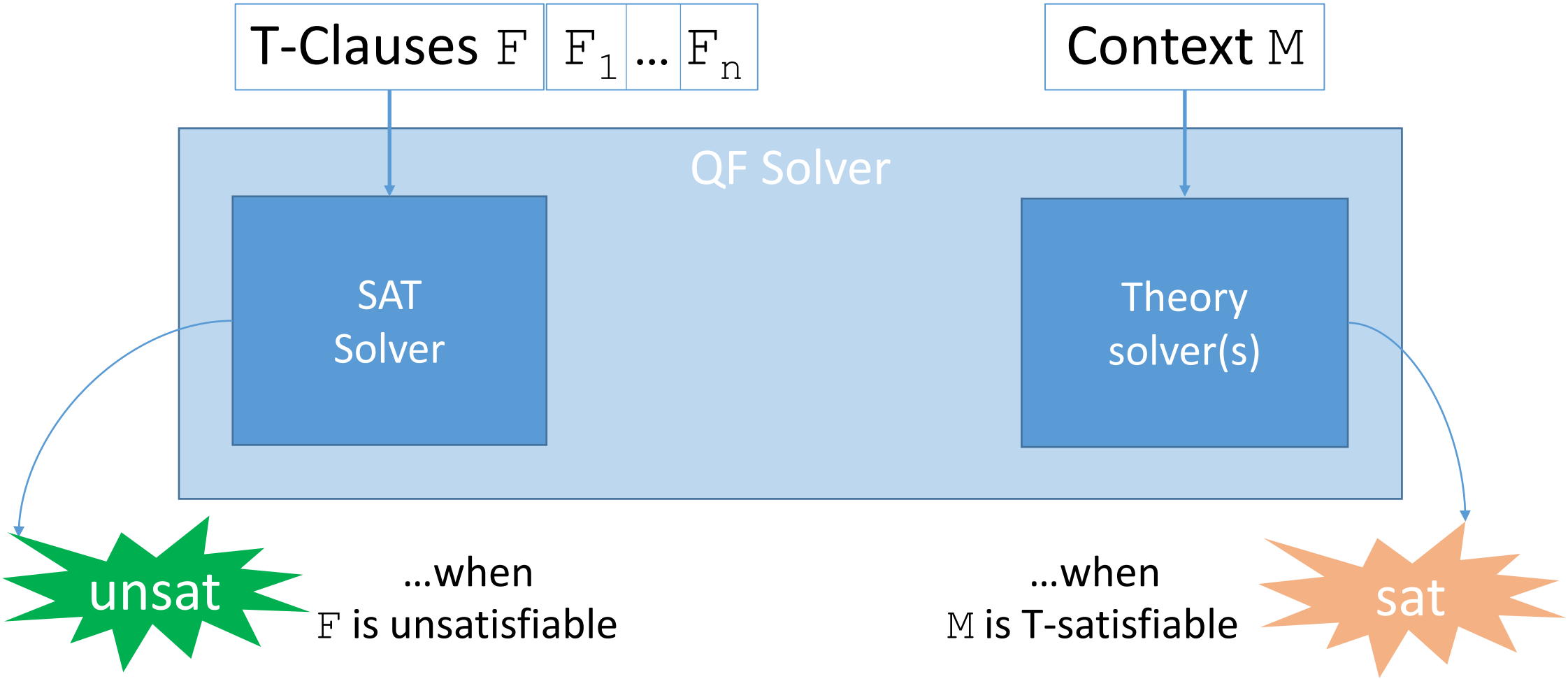
DPLL(T)-Based SMT Solvers



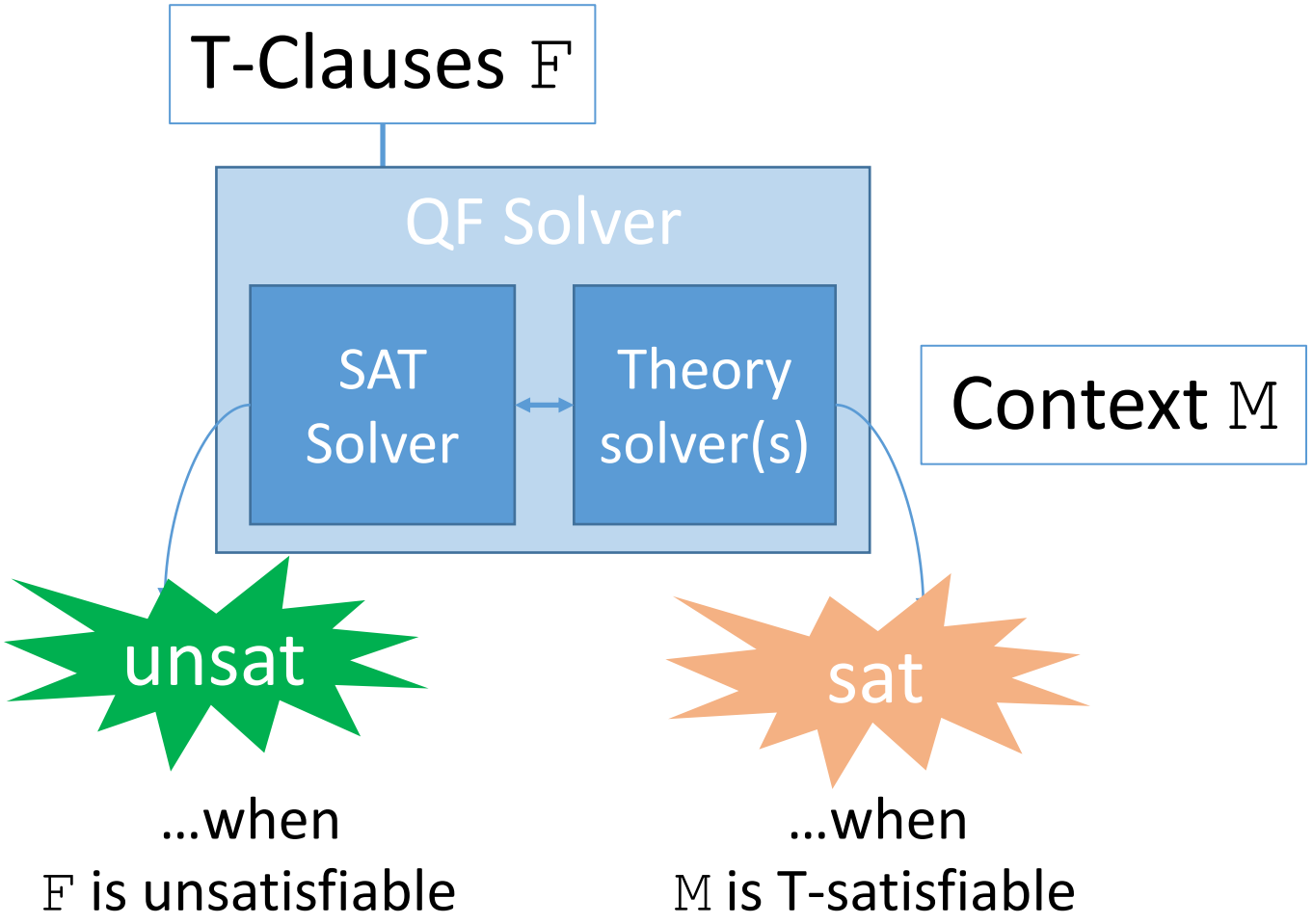
DPLL(T)-Based SMT Solvers



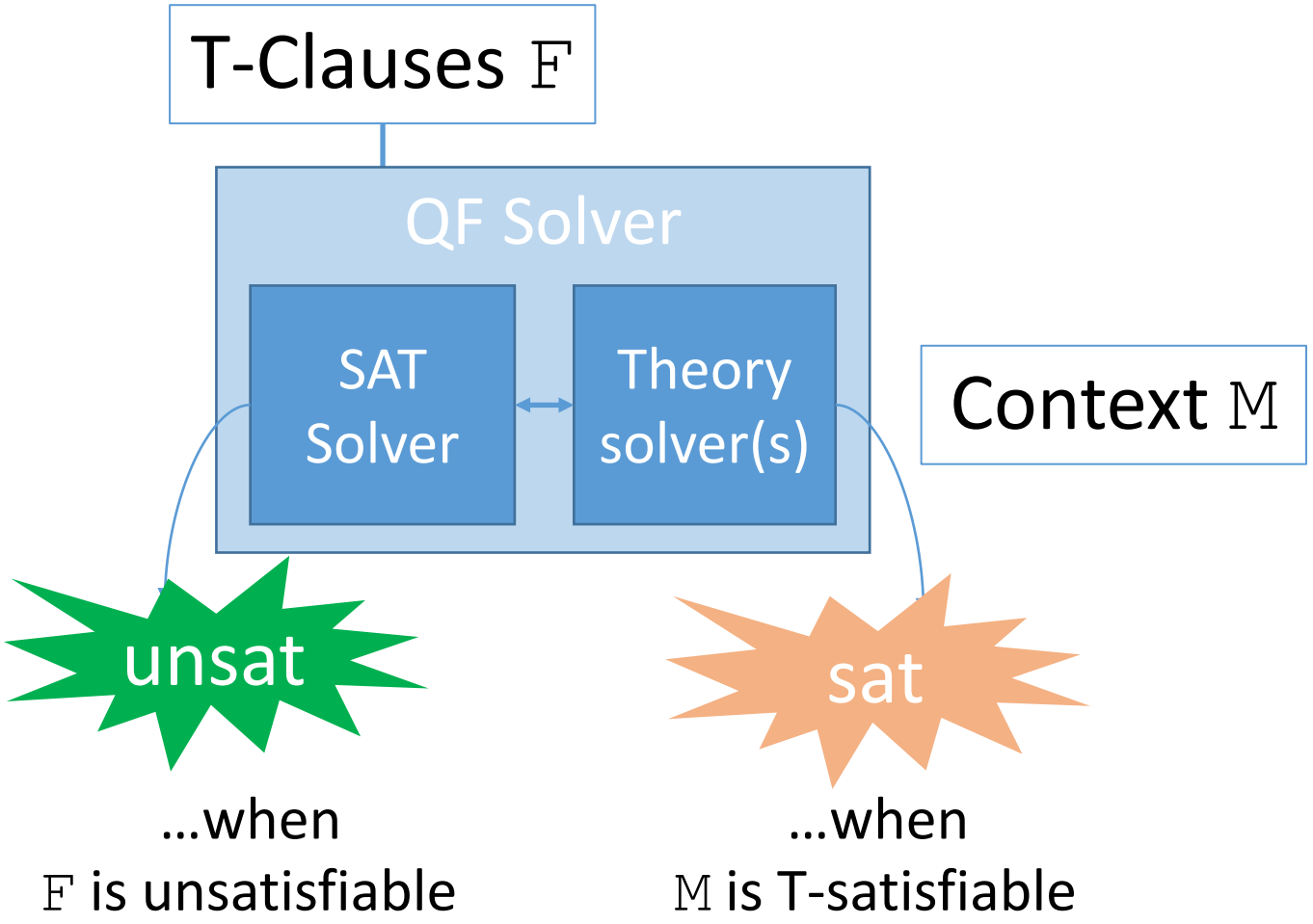
DPLL(T)-Based SMT Solvers



DPLL(T)-Based SMT Solvers + ∀ Instantiation

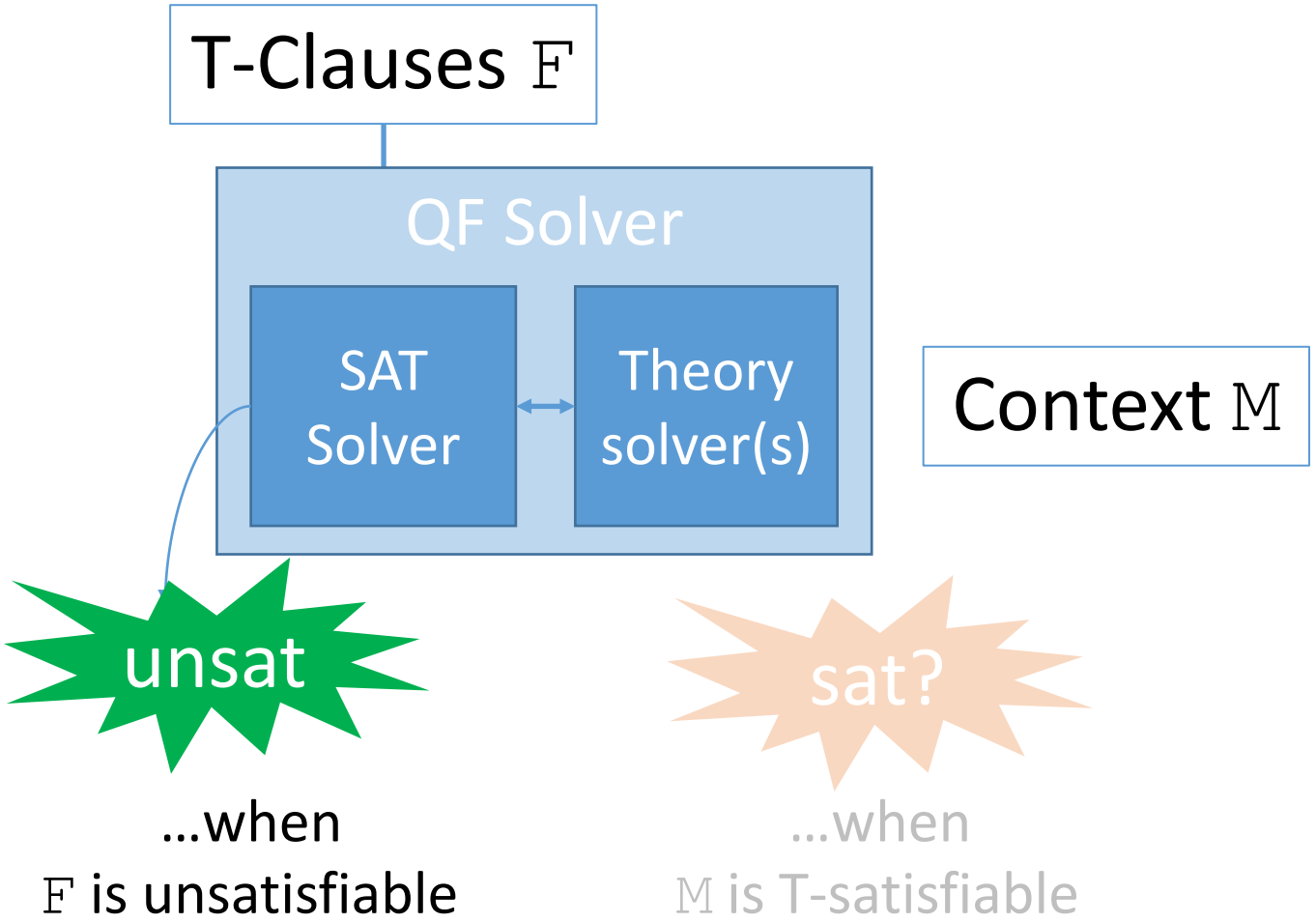


DPLL(T)-Based SMT Solvers + ∀ Instantiation



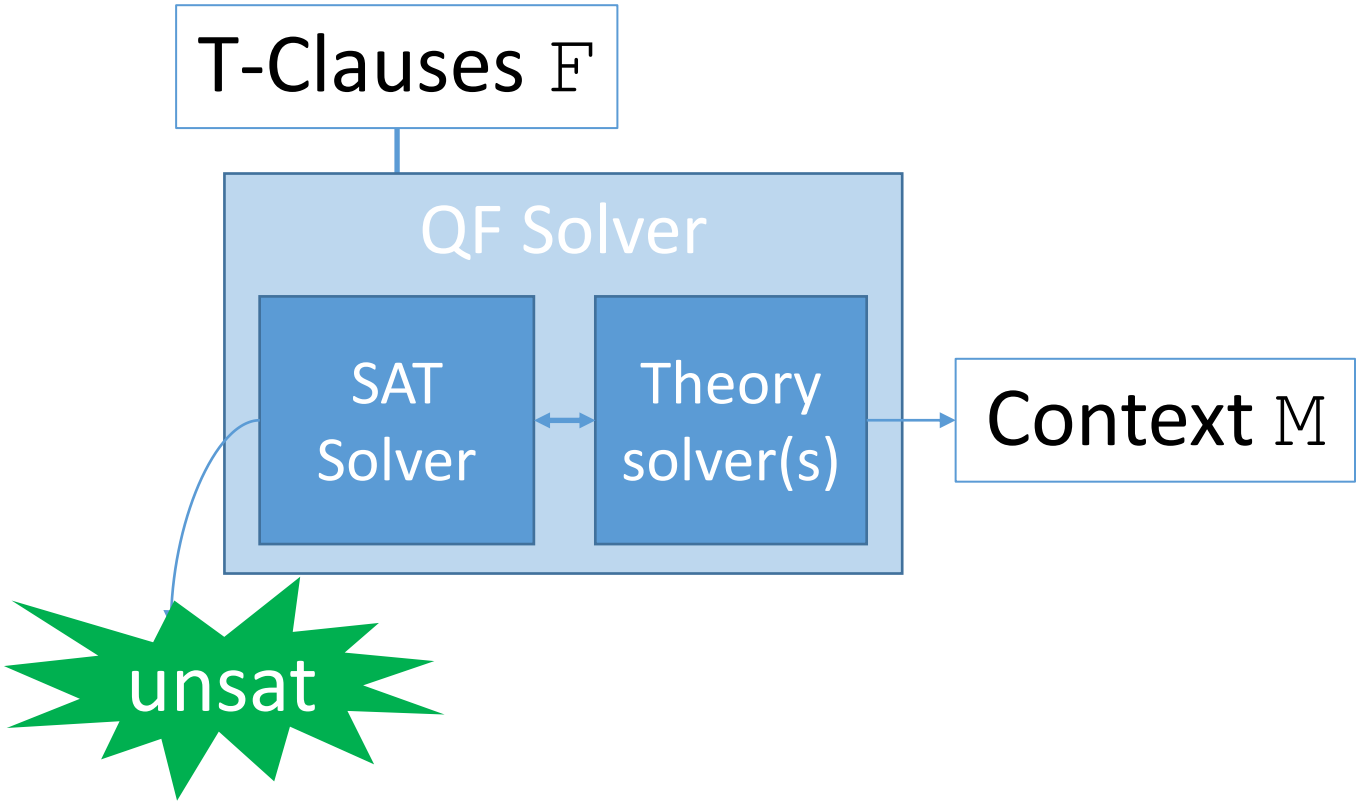
When M contains quantified formulas…

DPLL(T)-Based SMT Solvers + ∀ Instantiation



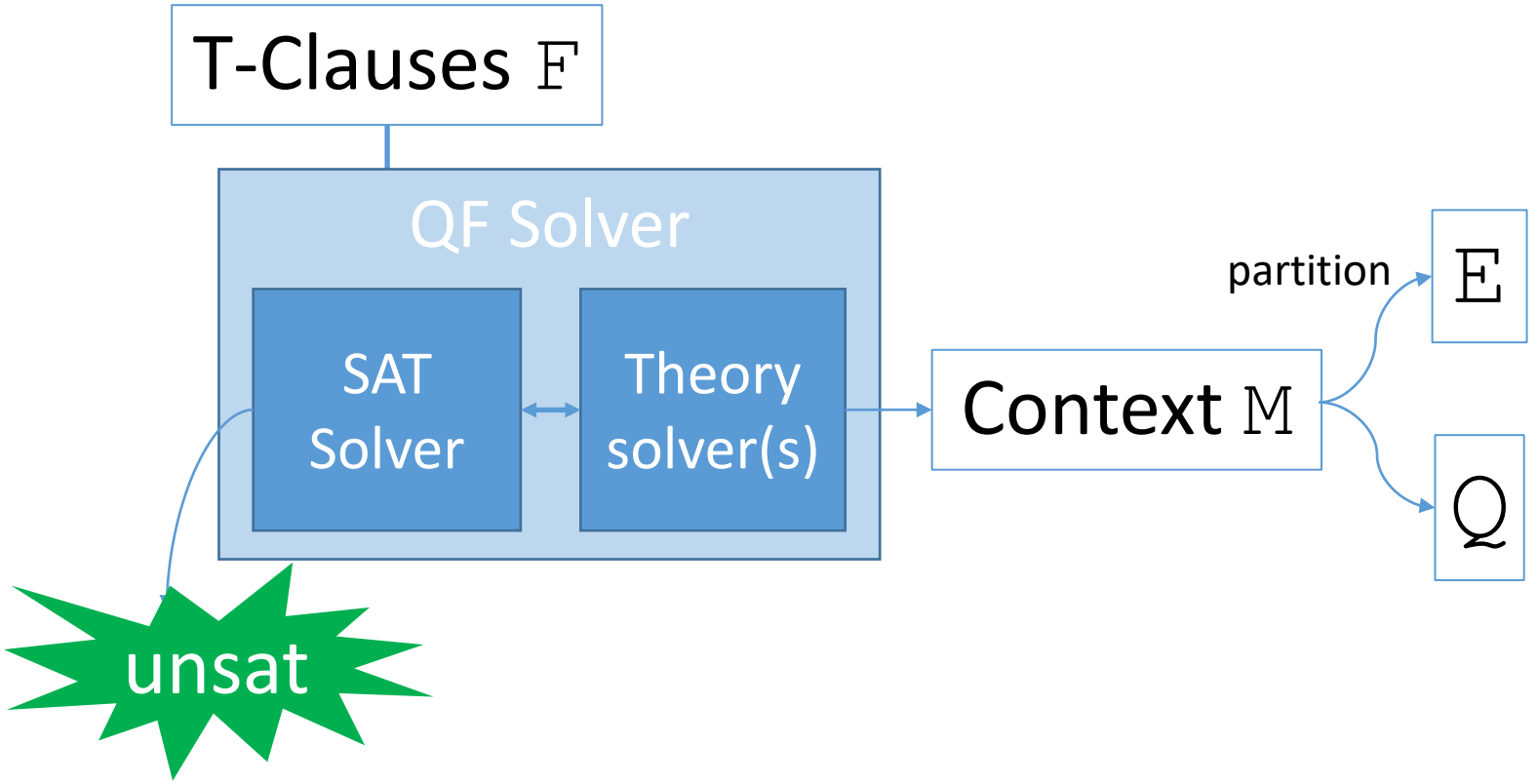
…cannot use QF procedure for establishing M is sat

DPLL(T)-Based SMT Solvers + ∀ Instantiation



…when
$\mathbb{F}$ is unsatisfiable

DPLL(T)-Based SMT Solvers + ∀ Instantiation



…when
F is unsatisfiable

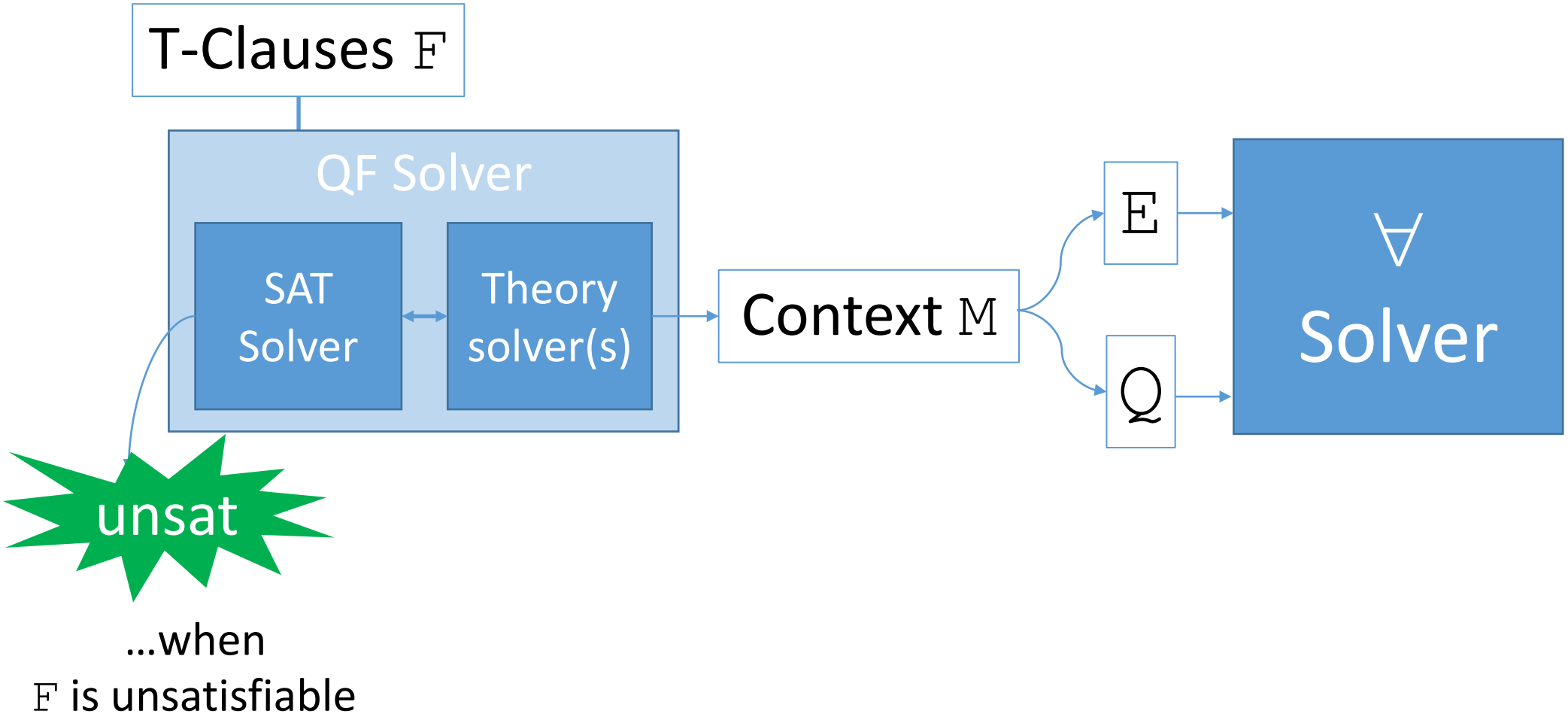
Set of ground equalities and disequalities

- {f(a)=b, P(a)=⊥, …}

Set of quantified formulas

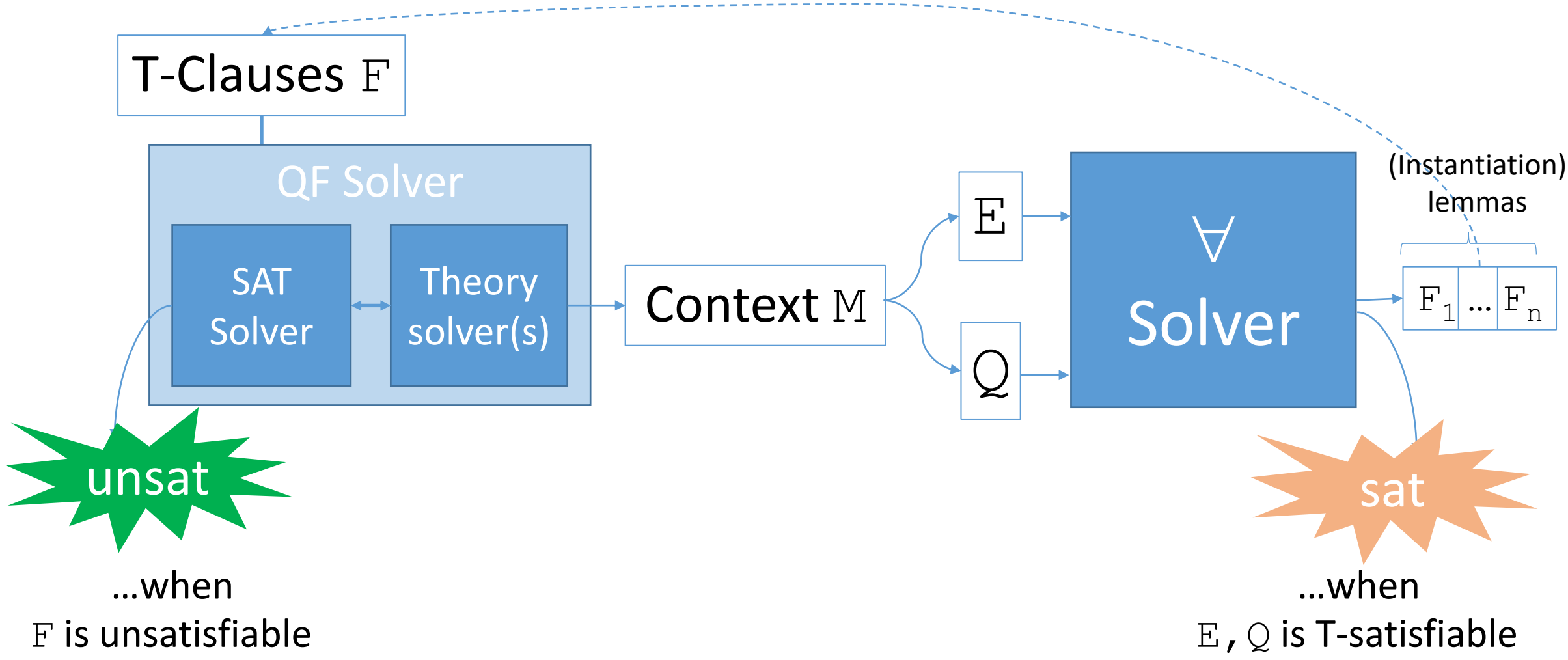
- {∀x.P(x), …}

DPLL(T)-Based SMT Solvers + ∀ Instantiation

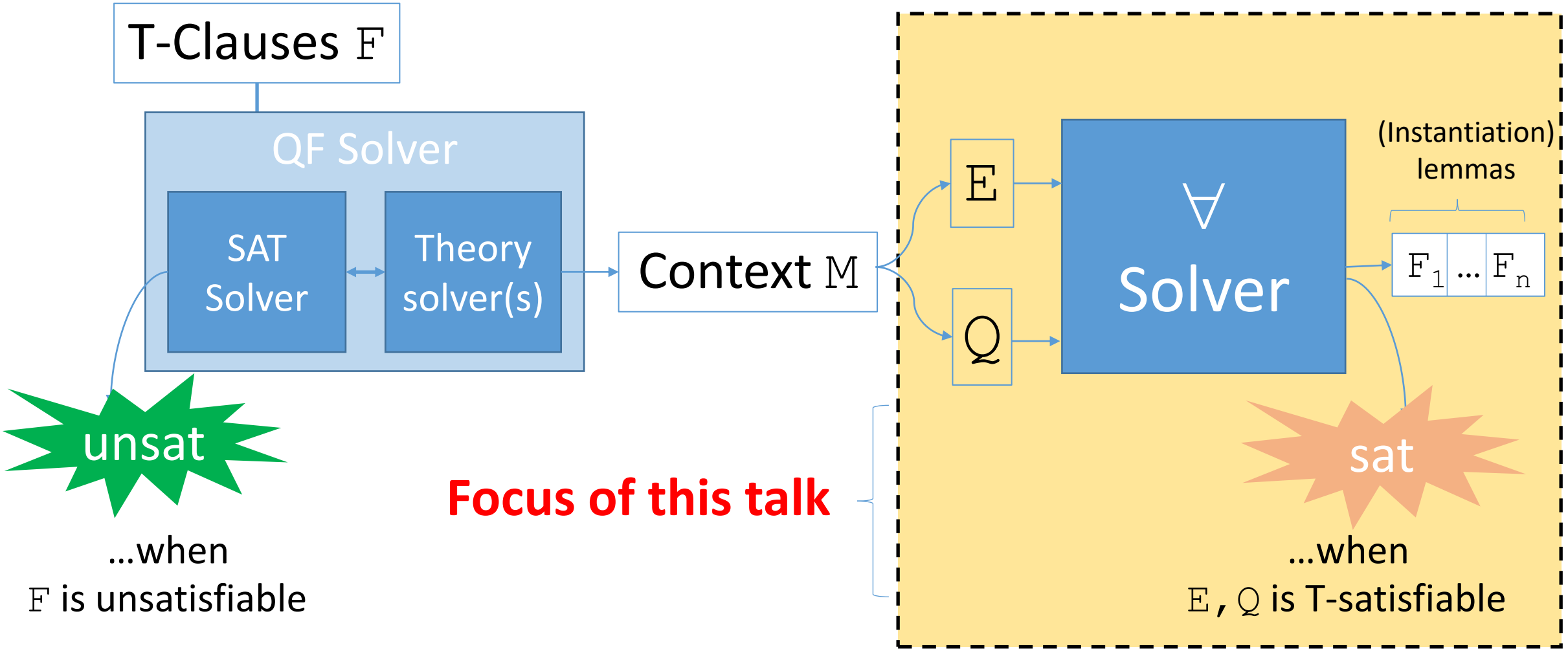


…when
F is unsatisfiable

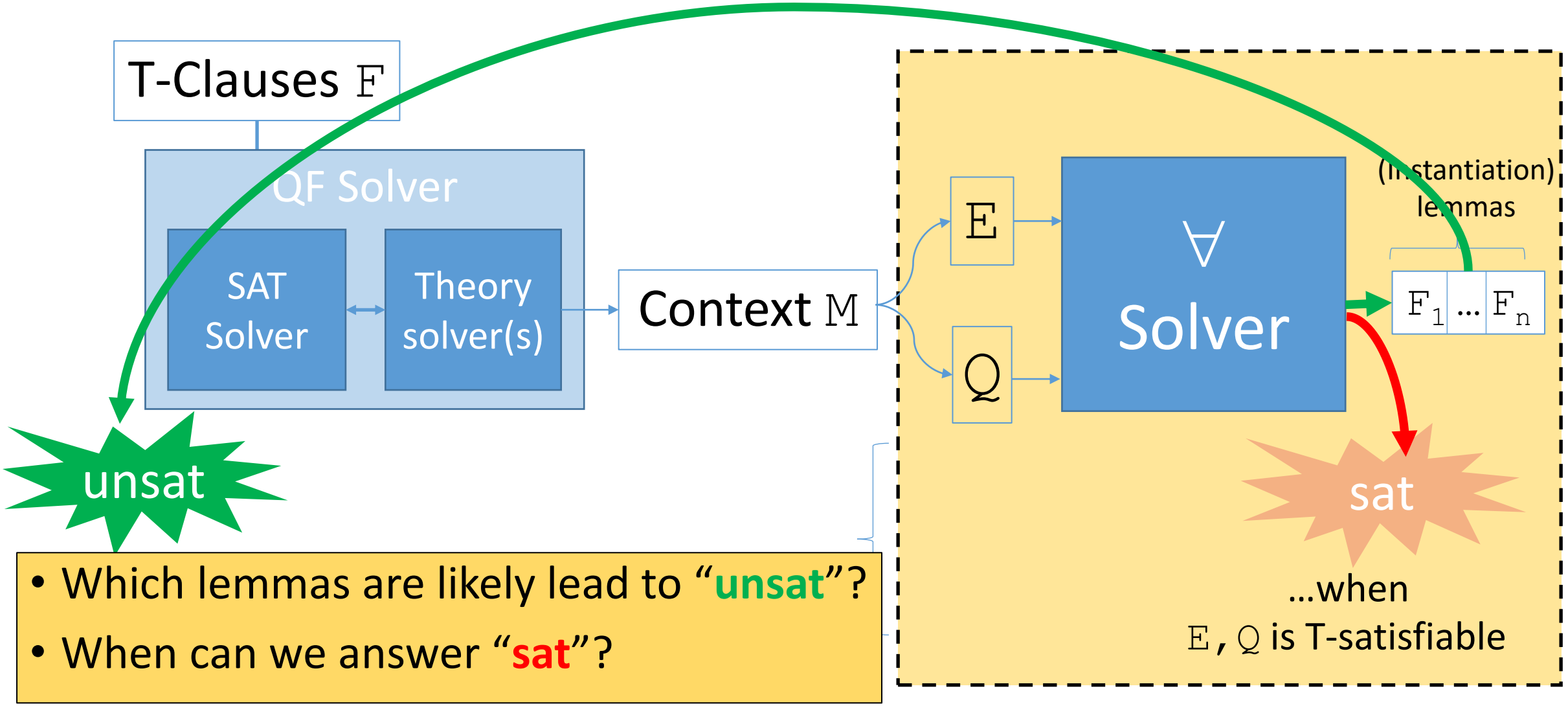
DPLL(T)-Based SMT Solvers + ∀ Instantiation



DPLL(T)-Based SMT Solvers + ∀ Instantiation

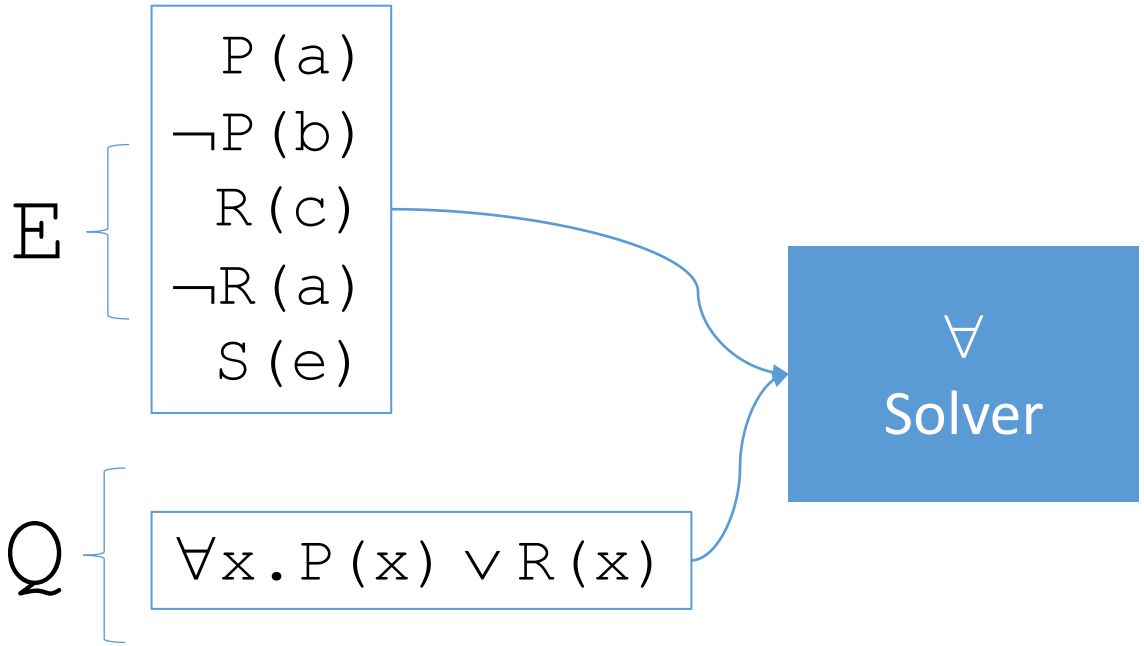


DPLL(T)-Based SMT Solvers + ∀ Instantiation

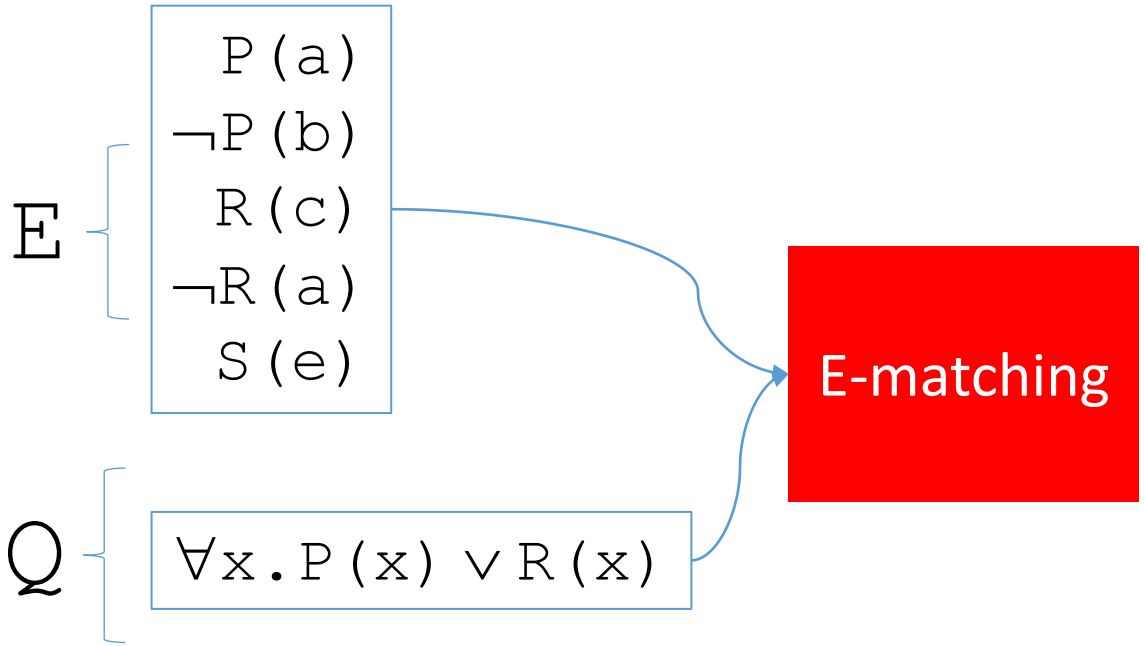


- Which lemmas are likely lead to "**unsat**"?
- When can we answer "**sat**"?

E-matching

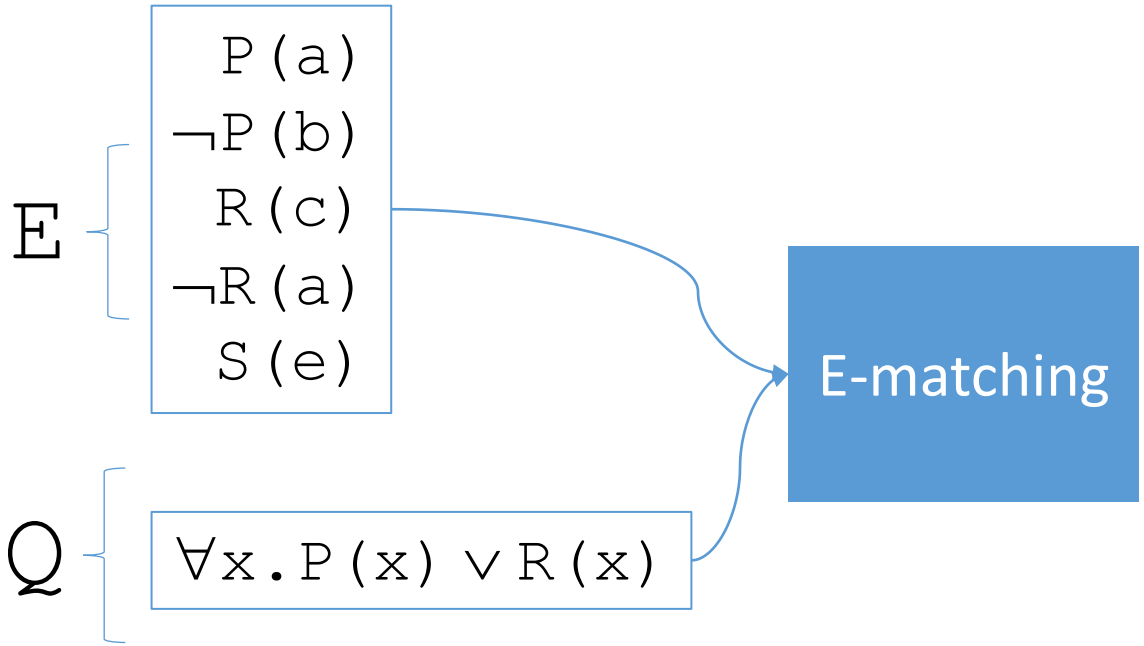


E-matching

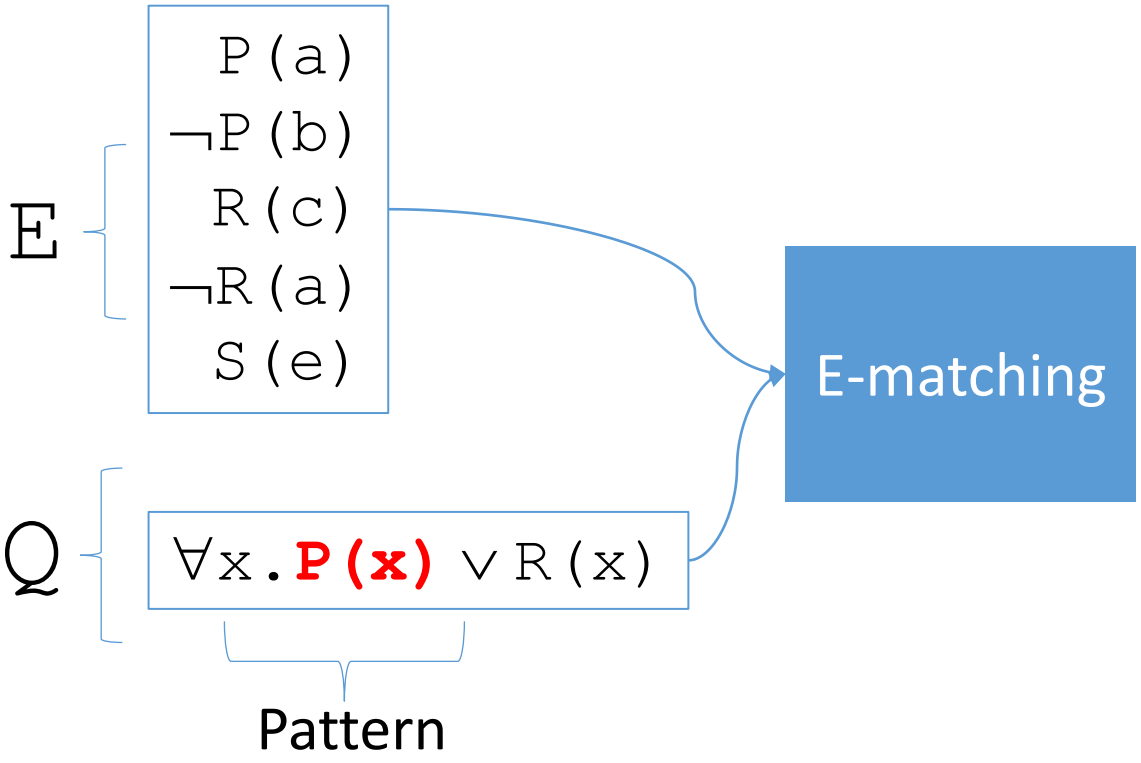


- Introduced in Nelson's Phd Thesis [Nelson 80]

E-matching

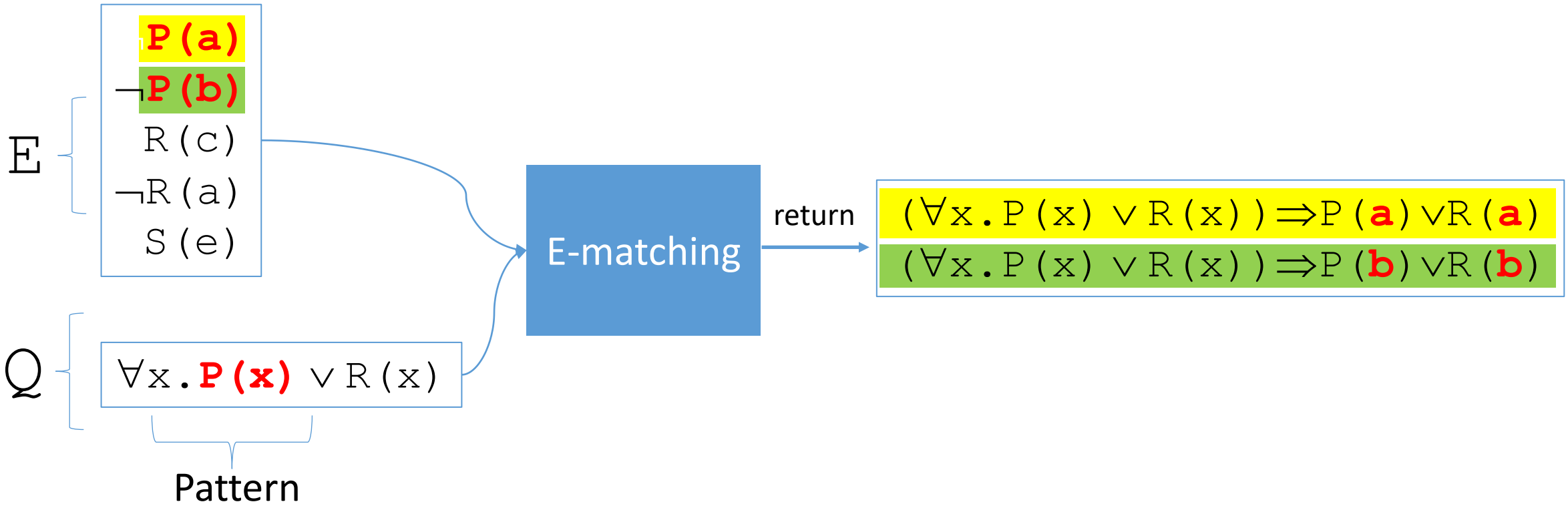


E-matching

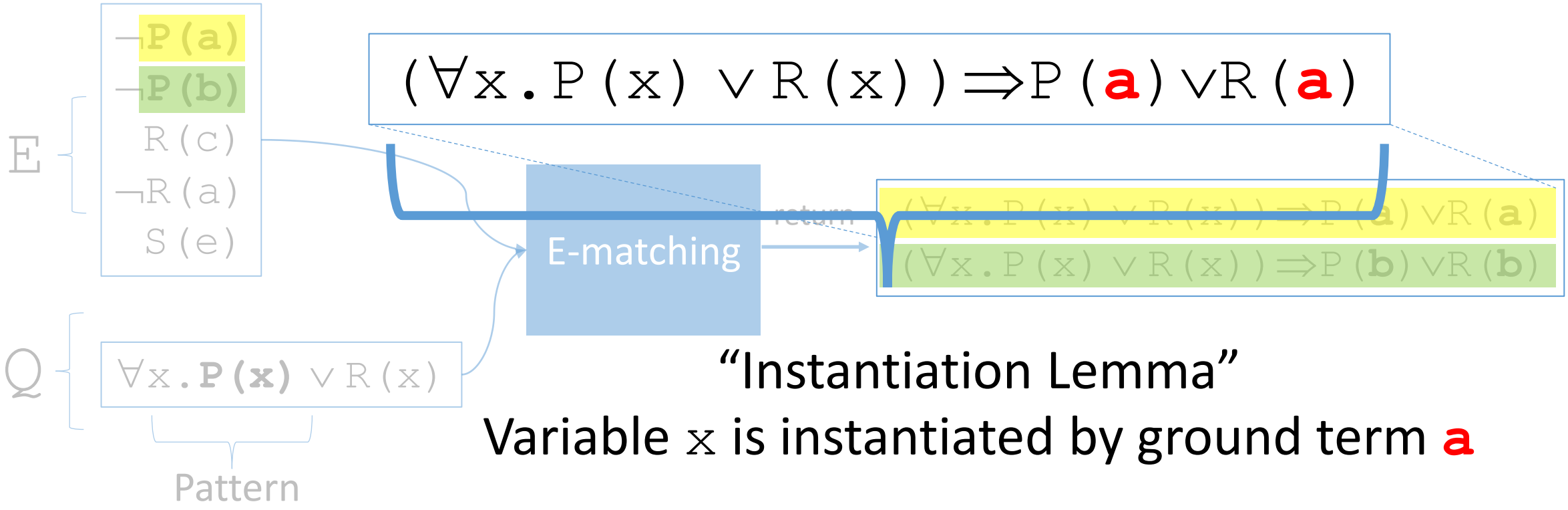


⇒ **Idea**: choose instances based on pattern matching

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)

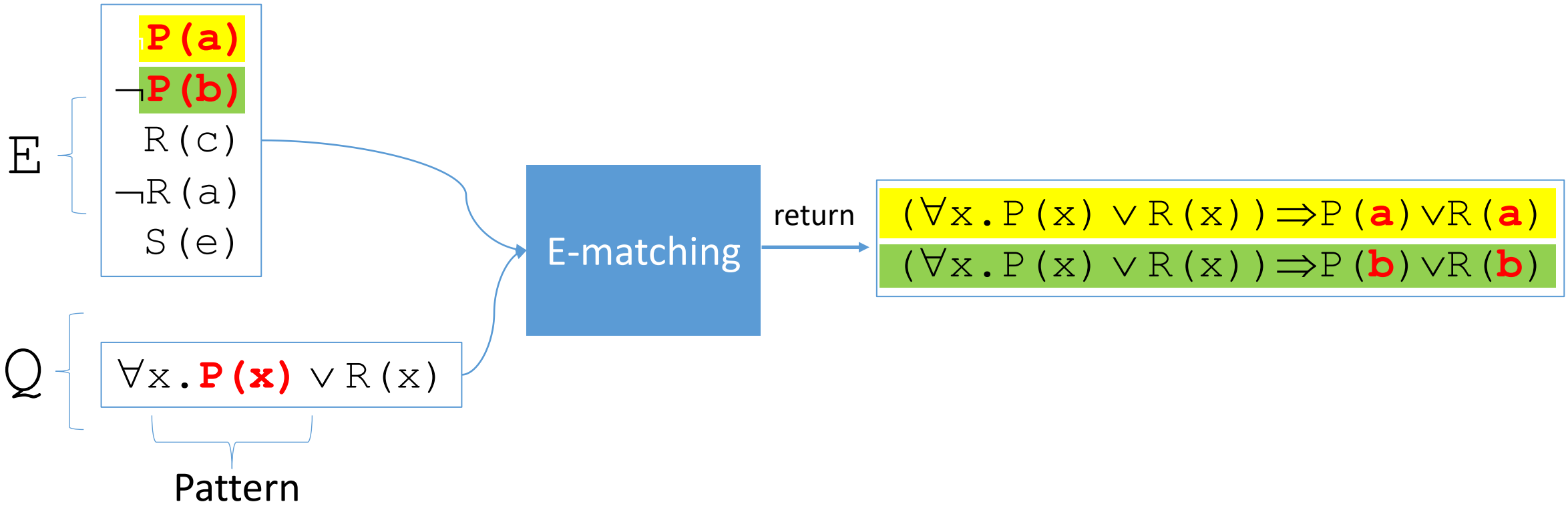
E-matching



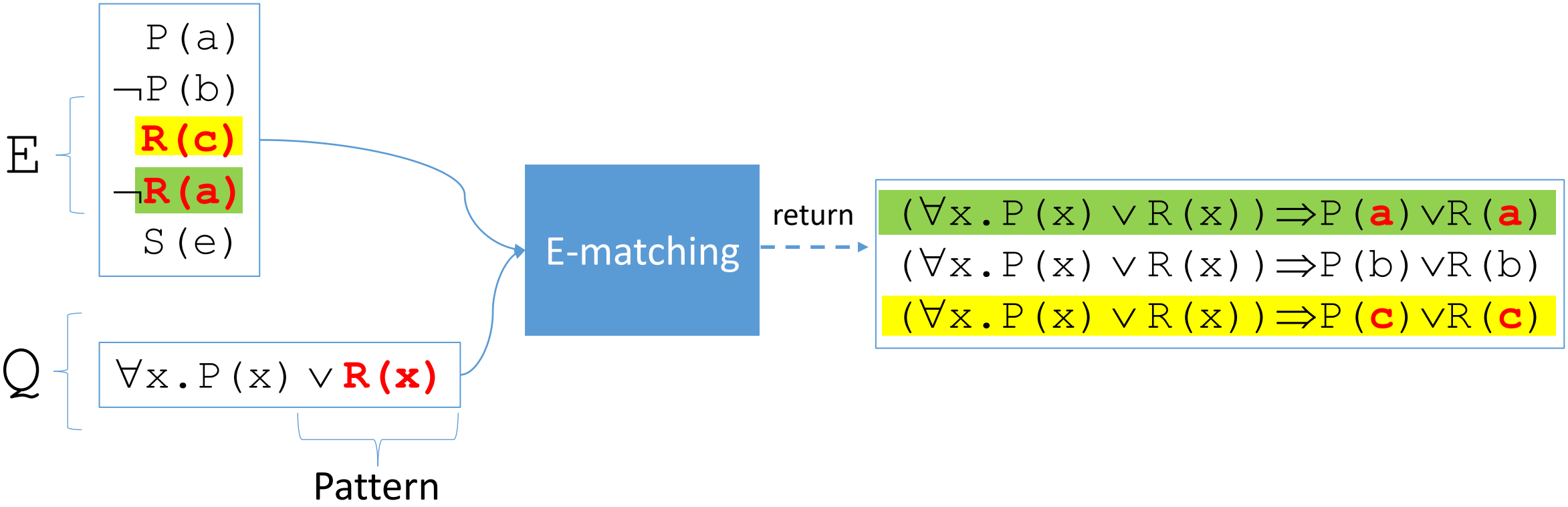
"Instantiation Lemma"

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

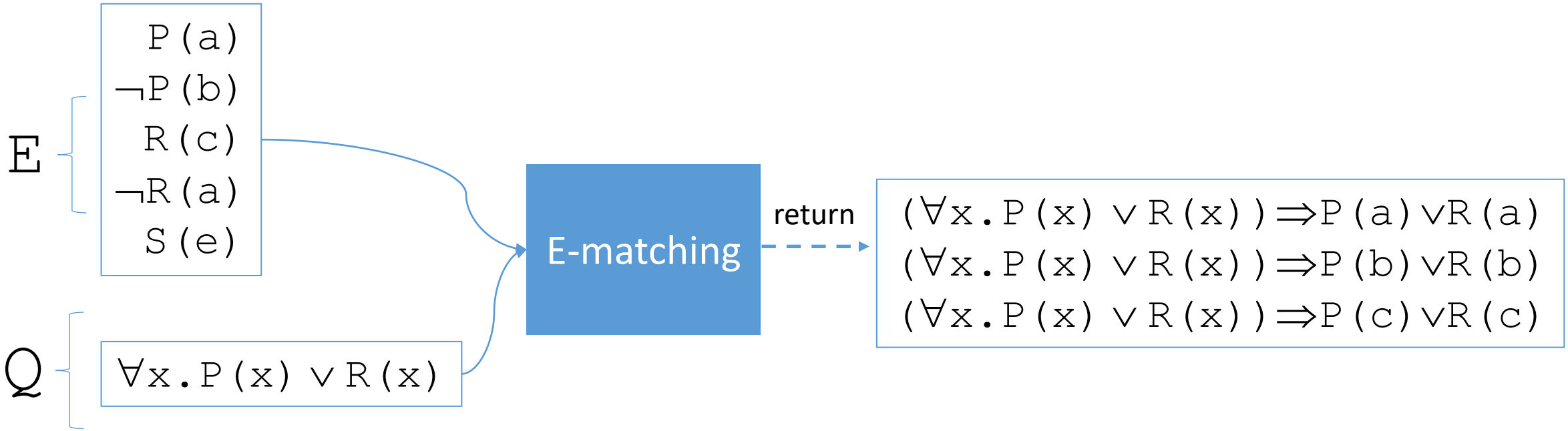
E-matching


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)

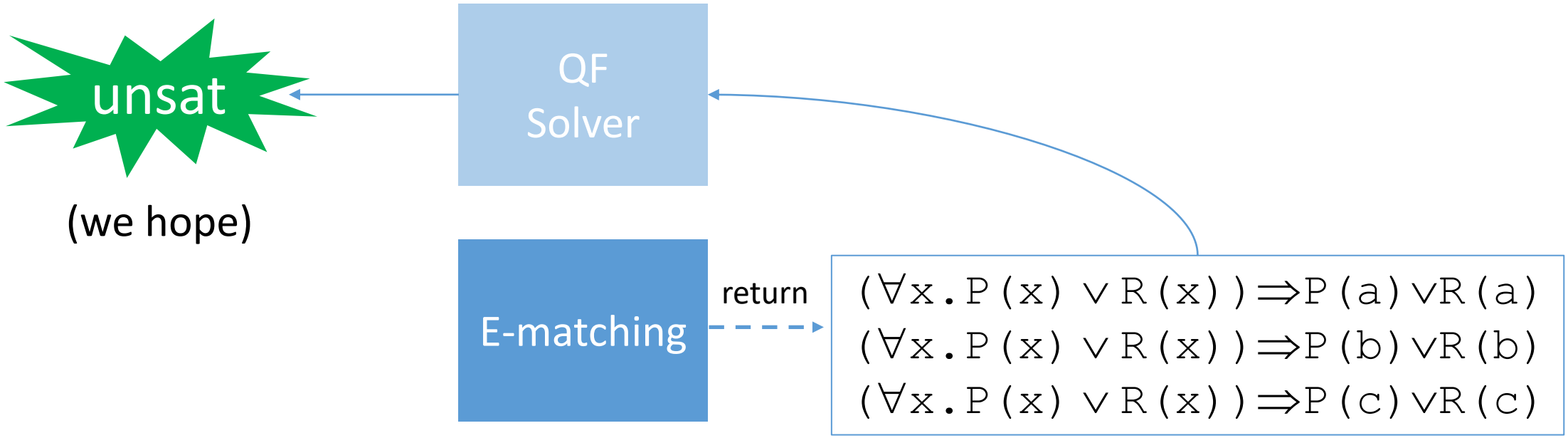
E-matching


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)

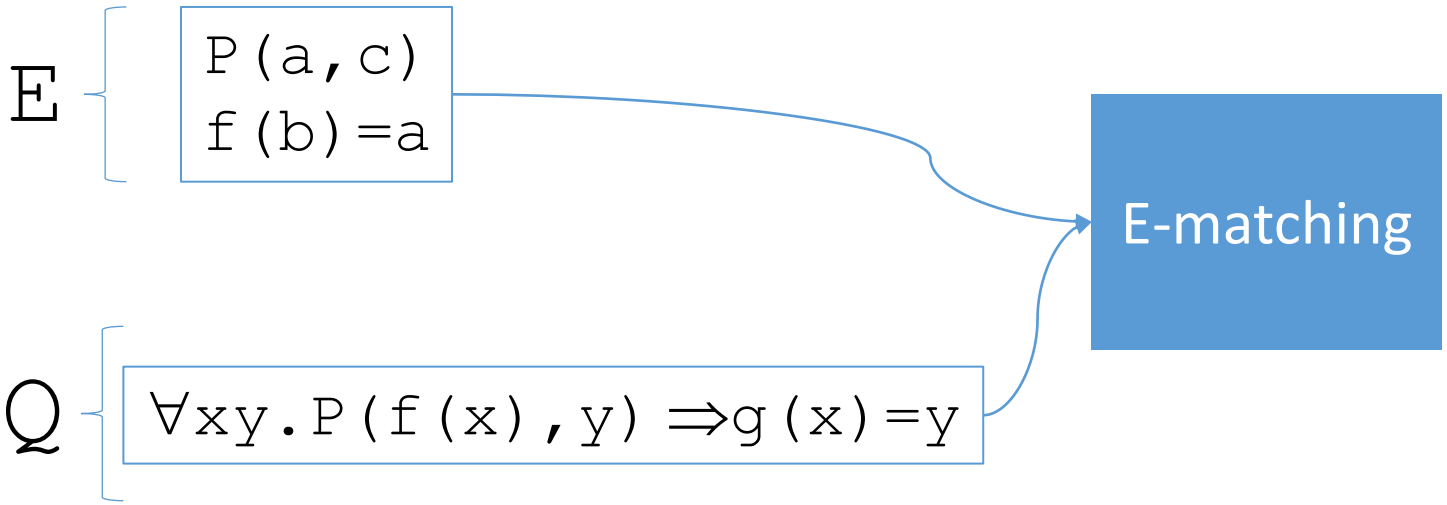
E-matching


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)

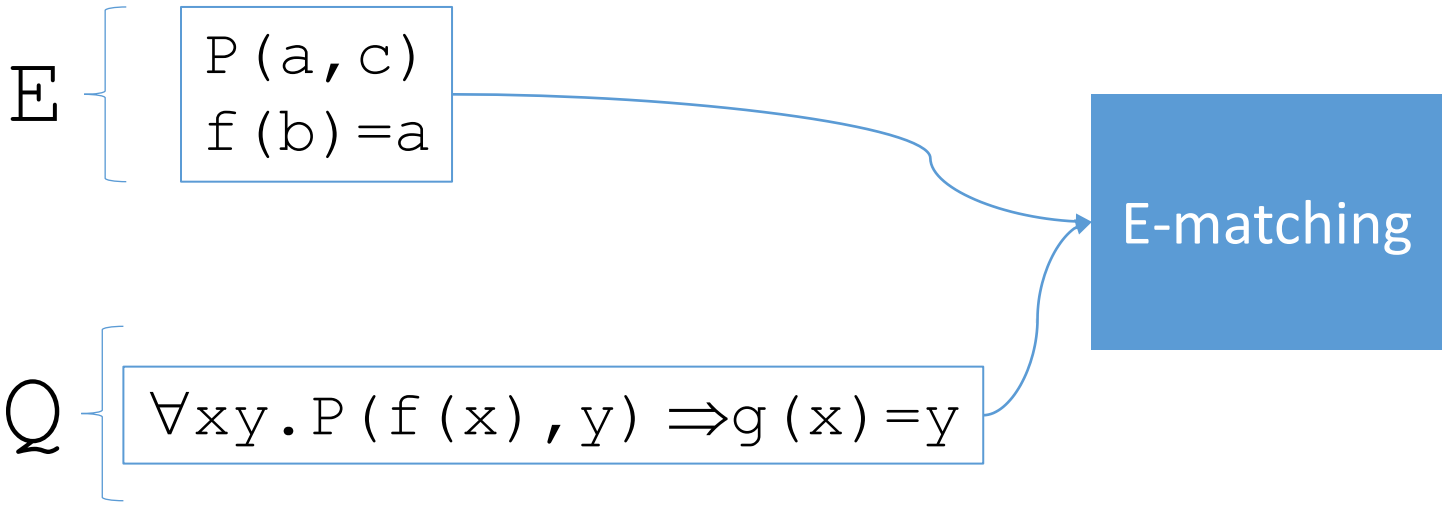
E-matching


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

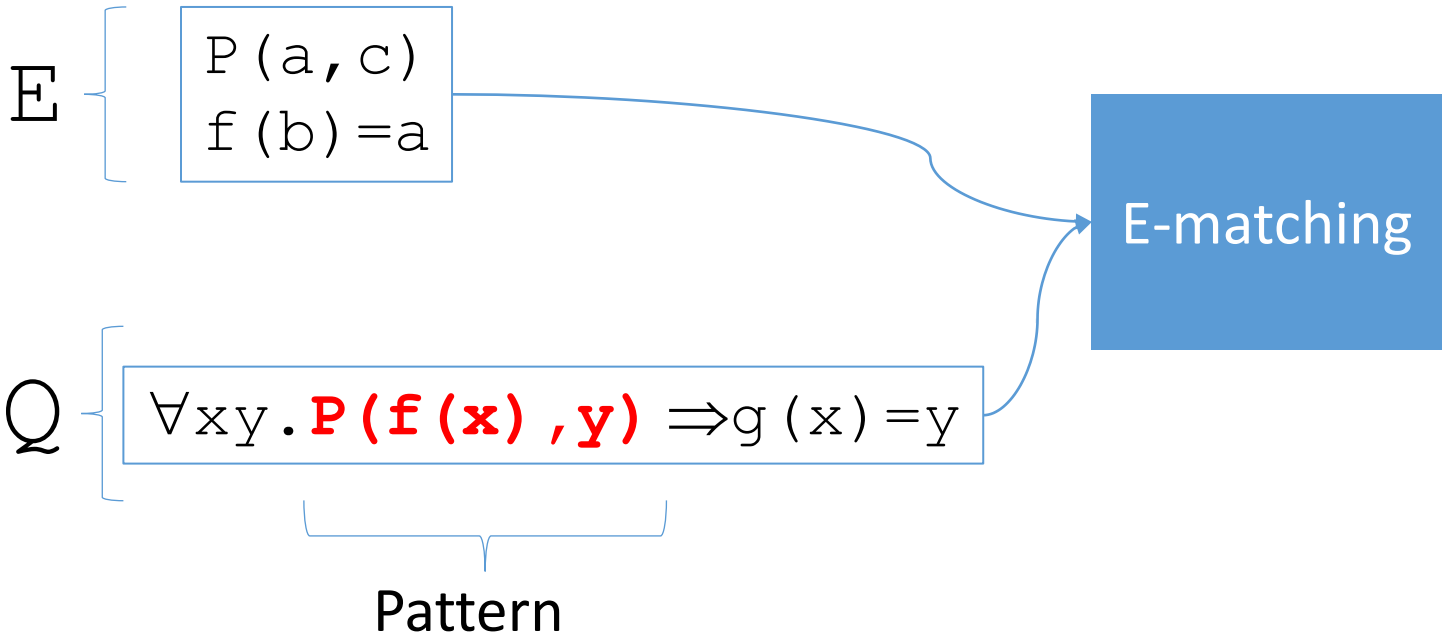


E-matching: Functions, Equality

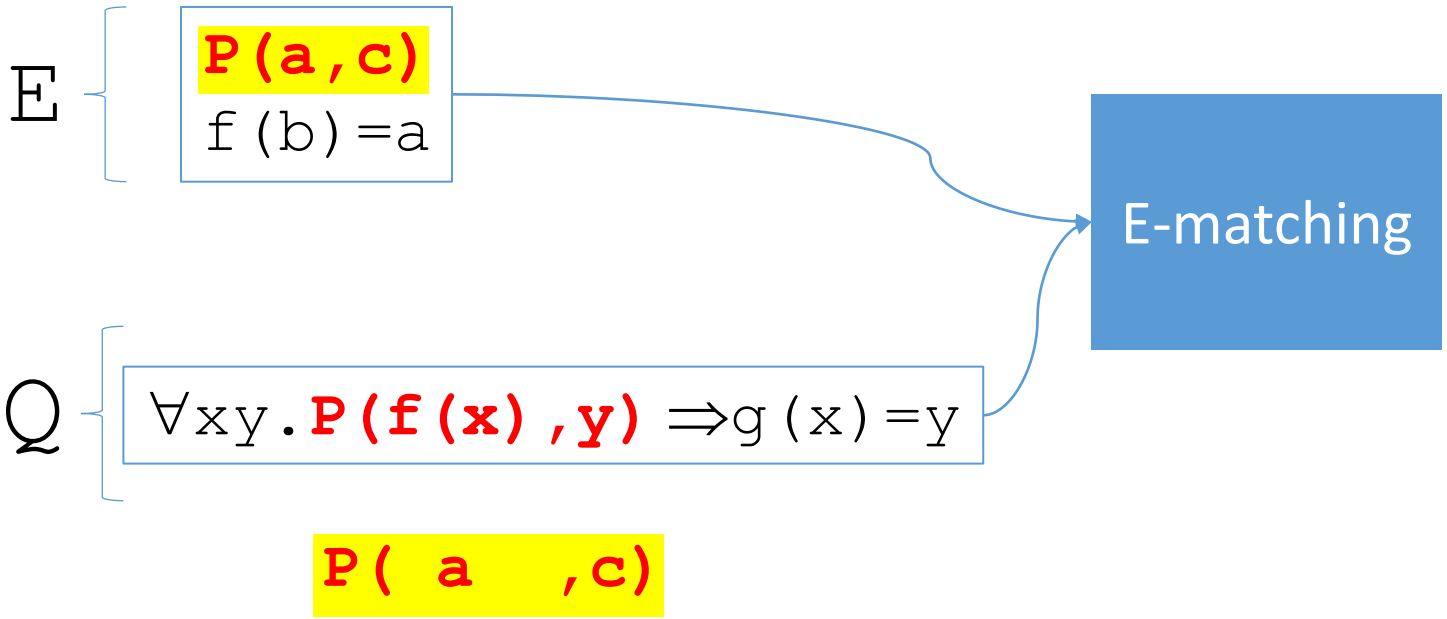


⇒ In E-matching, Pattern ***matching*** takes into account equalities in ***E***

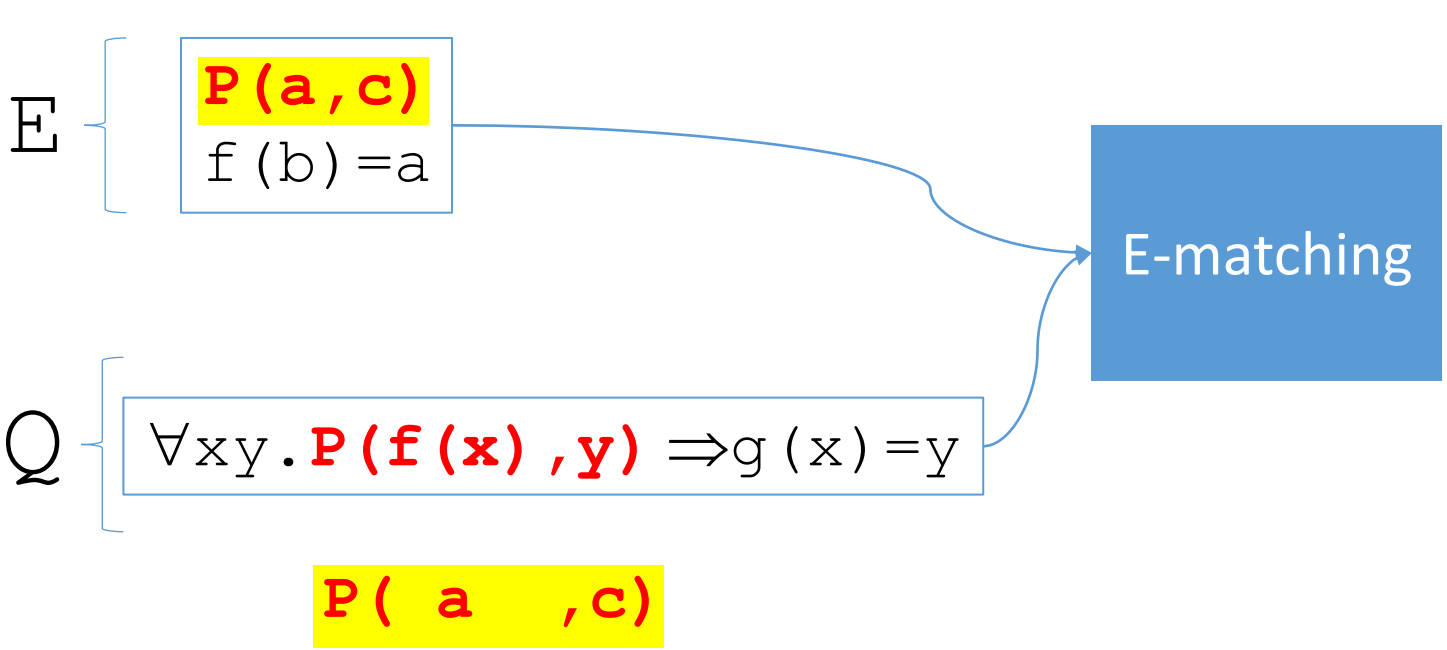
E-matching: Functions, Equality

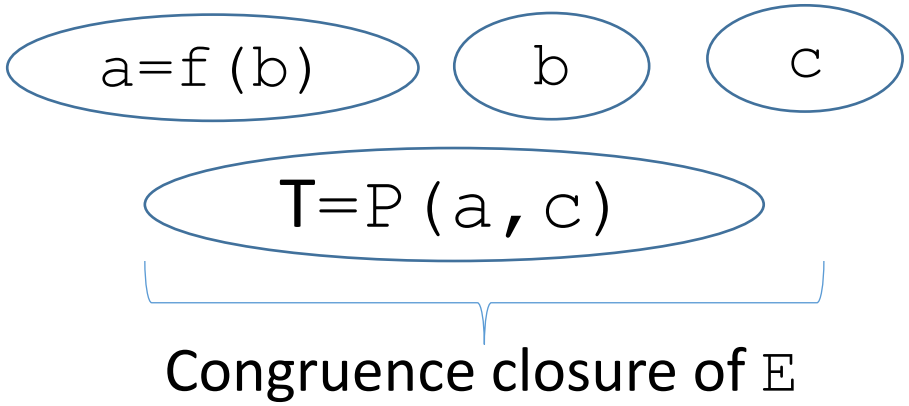


E-matching: Functions, Equality

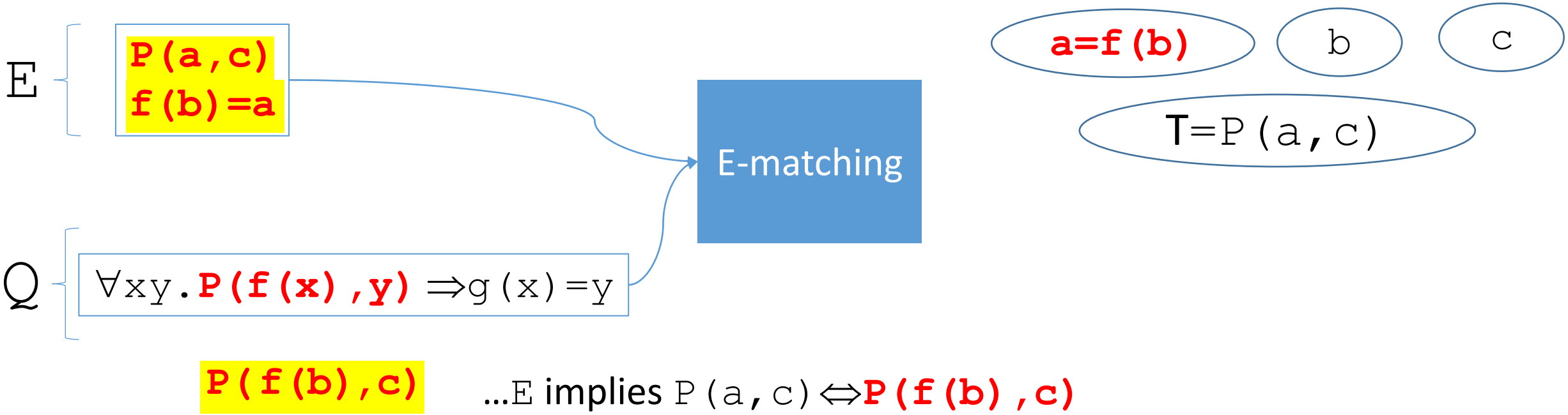


E-matching: Functions, Equality



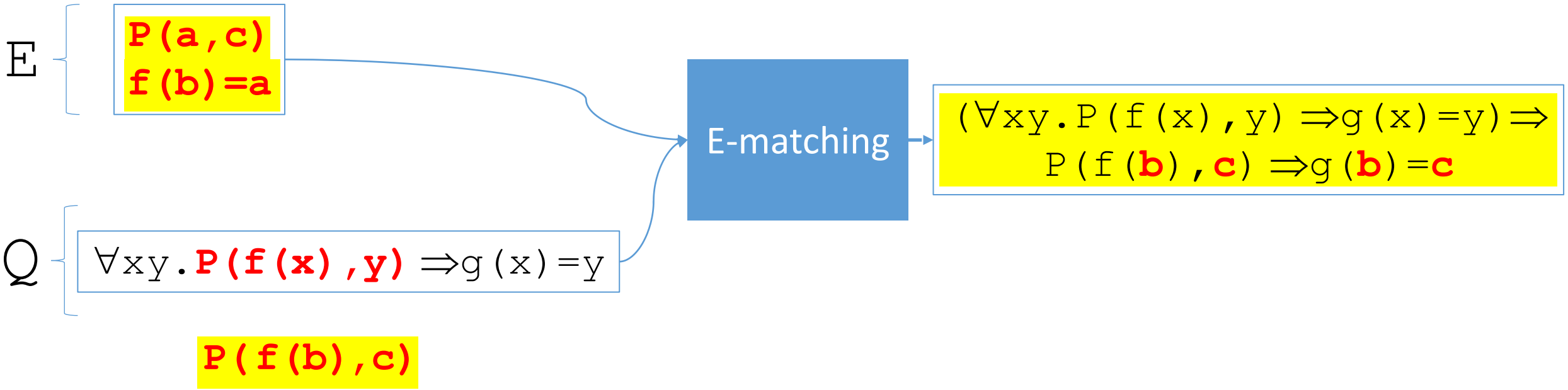


E-matching: Functions, Equality



P(f(b),c) …E implies P(a,c)⇔**P(f(b),c)**

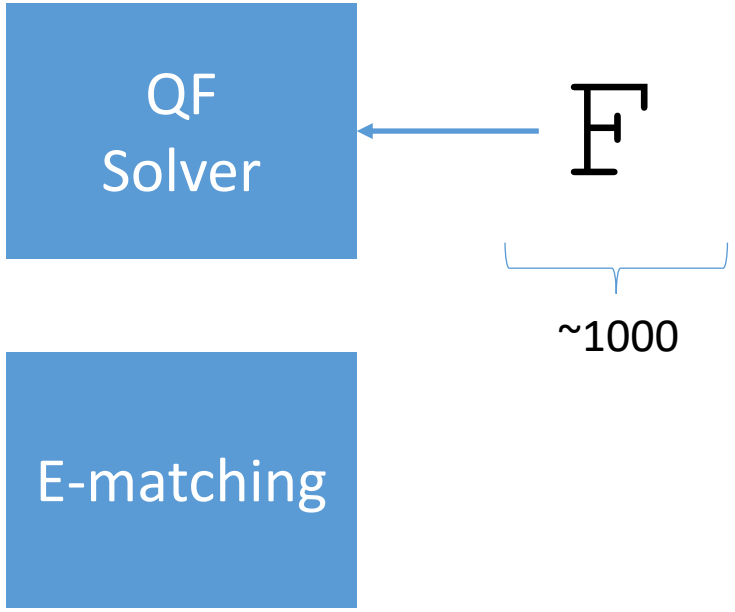
E-matching: Functions, Equality



E-matching

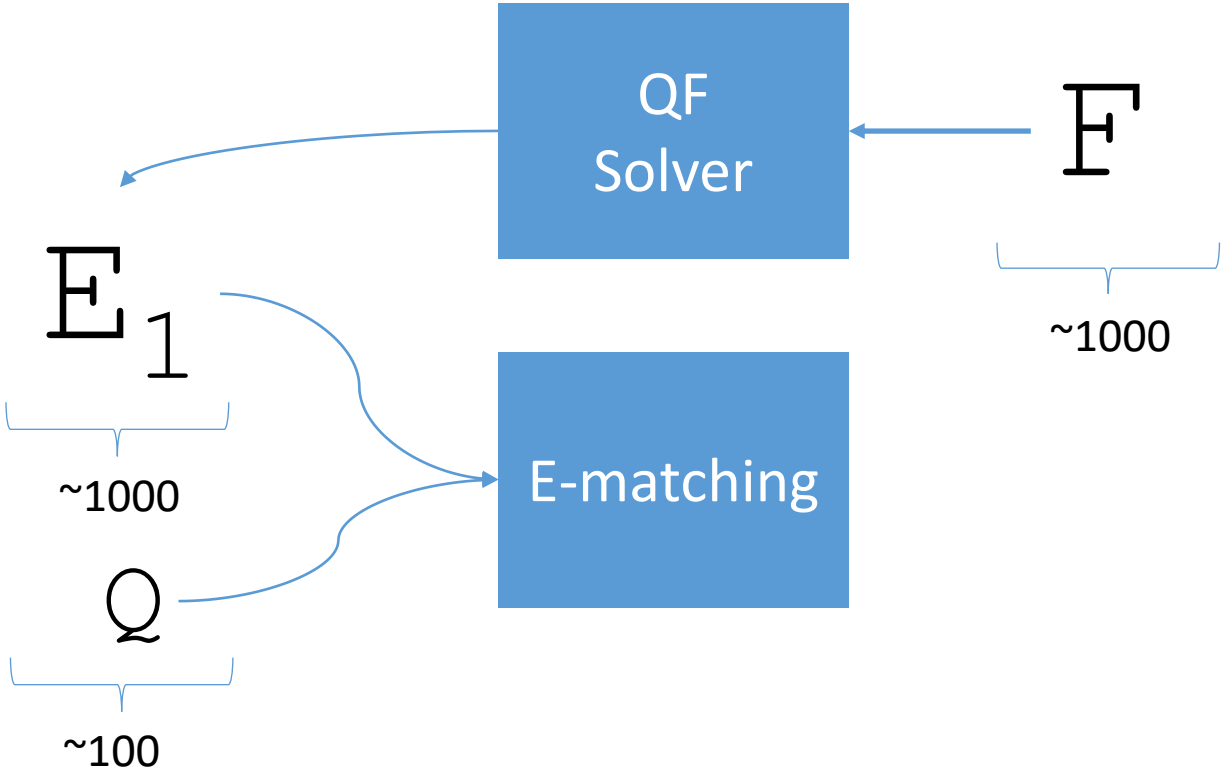
- Most widely used technique for unsatisfiable $\forall$ problems in SMT
  - Variants implemented in:
    - Z3 [deMoura et al 07], CVC3 [Ge et al 07], CVC4, Princess [Ruemmer 12], VeriT, Alt-Ergo
  - Used in:
    - Software verification
      - Boogie, Dafny, Leon, SPARK, Why3
    - Automated Theorem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

Challenge #1 : Too Many Instances



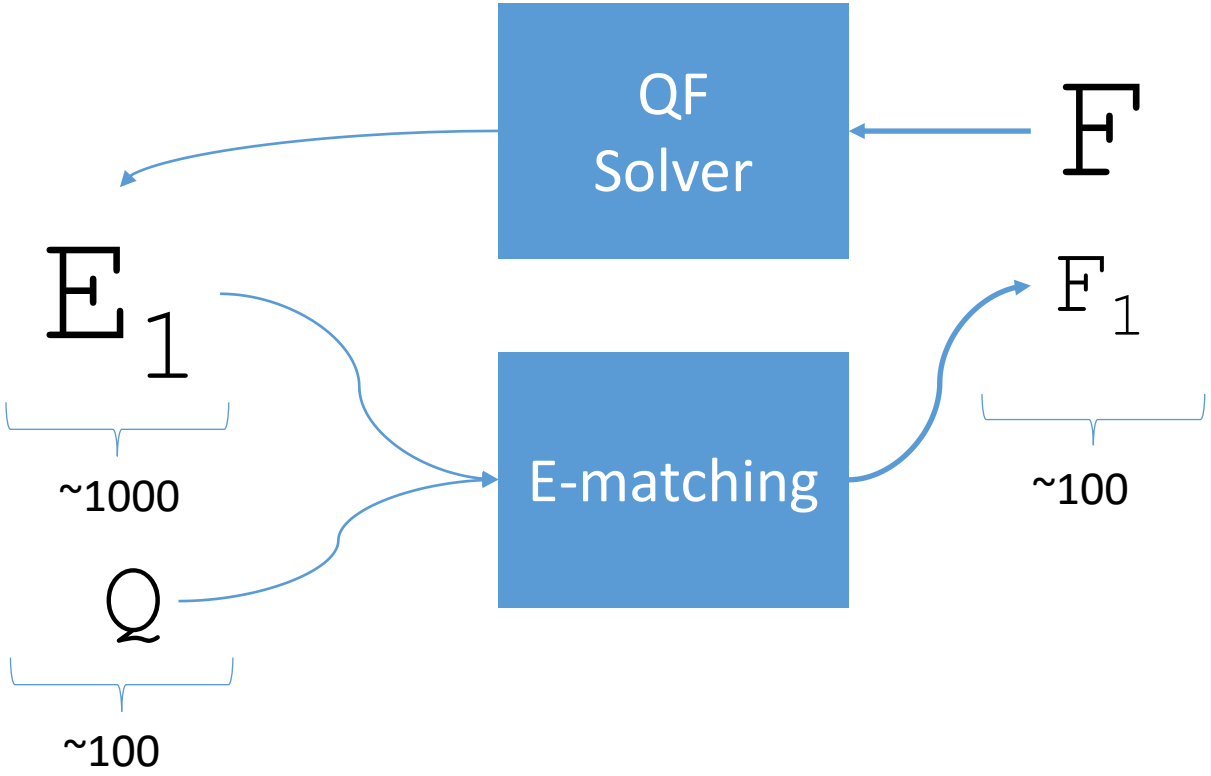
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses

Challenge #1 : Too Many Instances



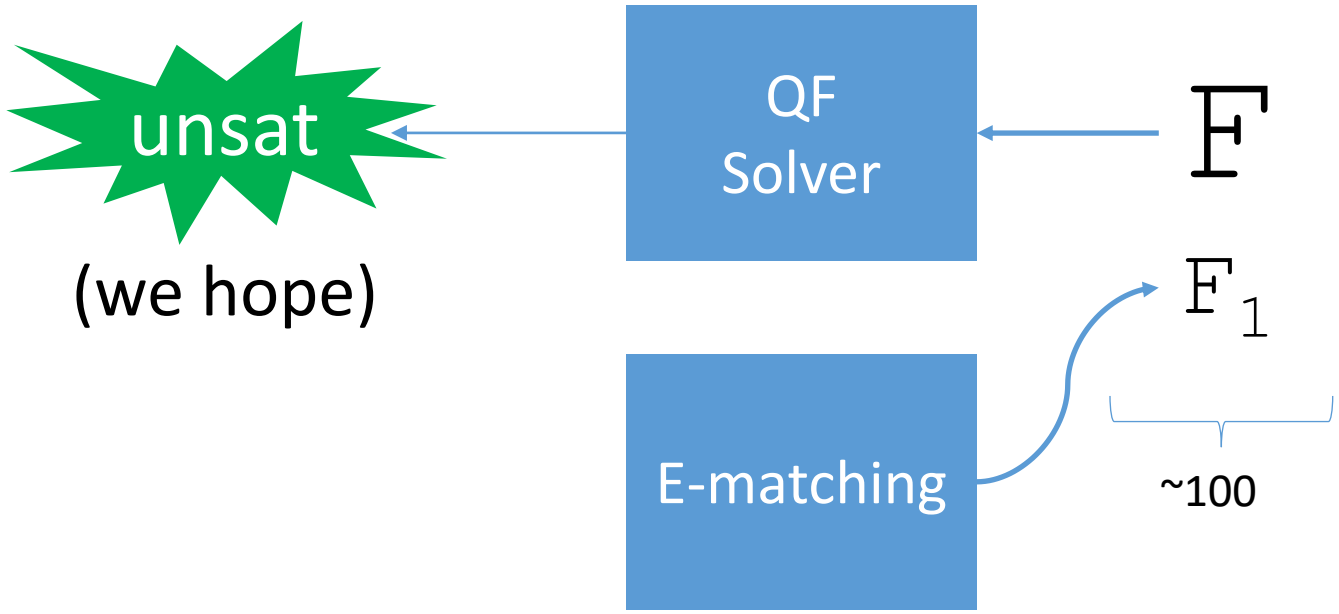
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Contexts contain 1000s of terms in $\mathbb{E}$, 100s of $\forall$ in $\mathbb{Q}$

Challenge #1 : Too Many Instances



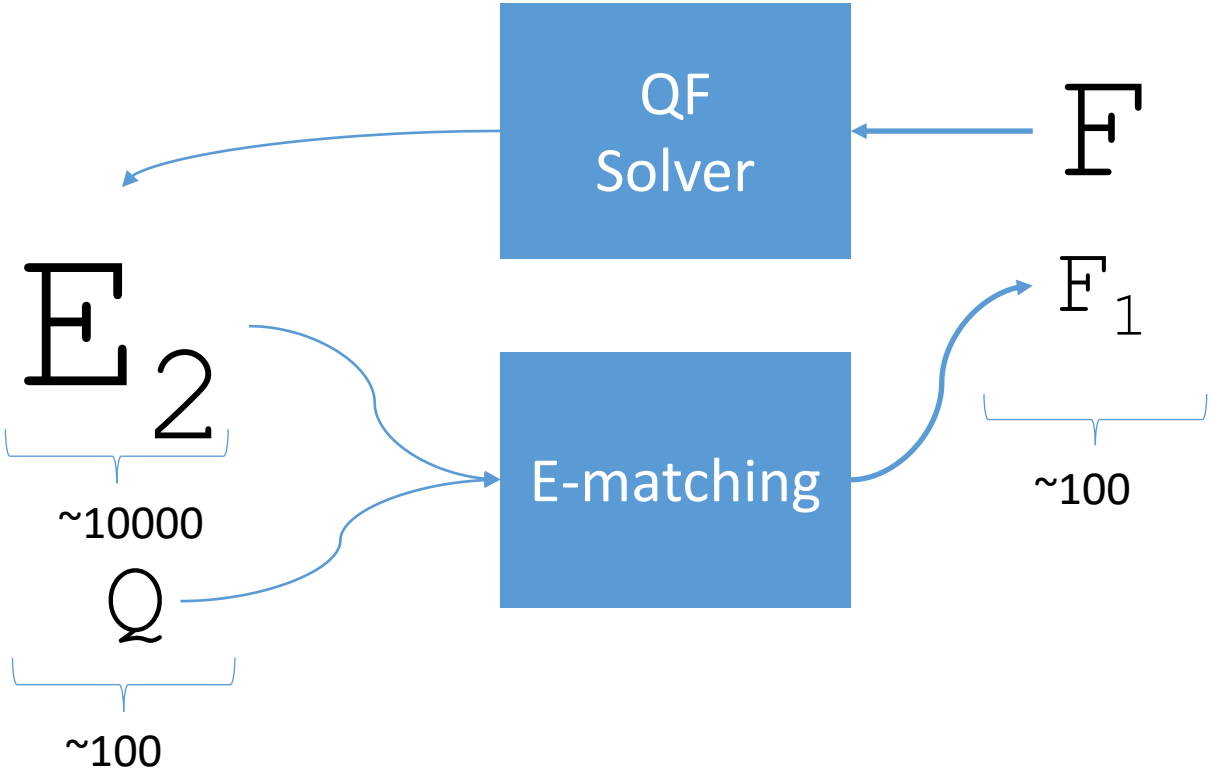
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Contexts contain 1000s of terms in $\mathbb{E}$, 100s of $\forall$ in $\mathbb{Q}$

Challenge #1 : Too Many Instances



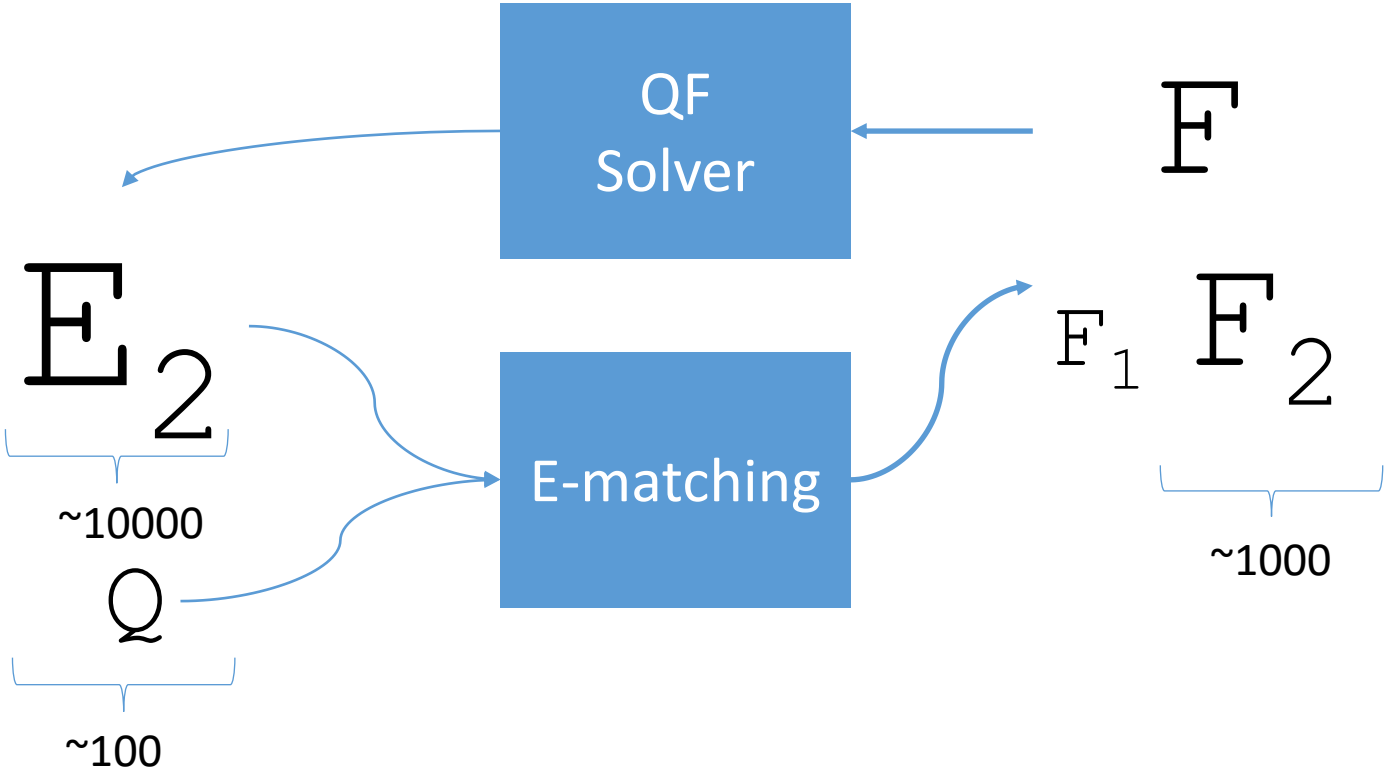
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Contexts contain 1000s of terms in $\mathbb{E}$, 100s of $\forall$ in $\mathbb{Q}$

Challenge #1 : Too Many Instances



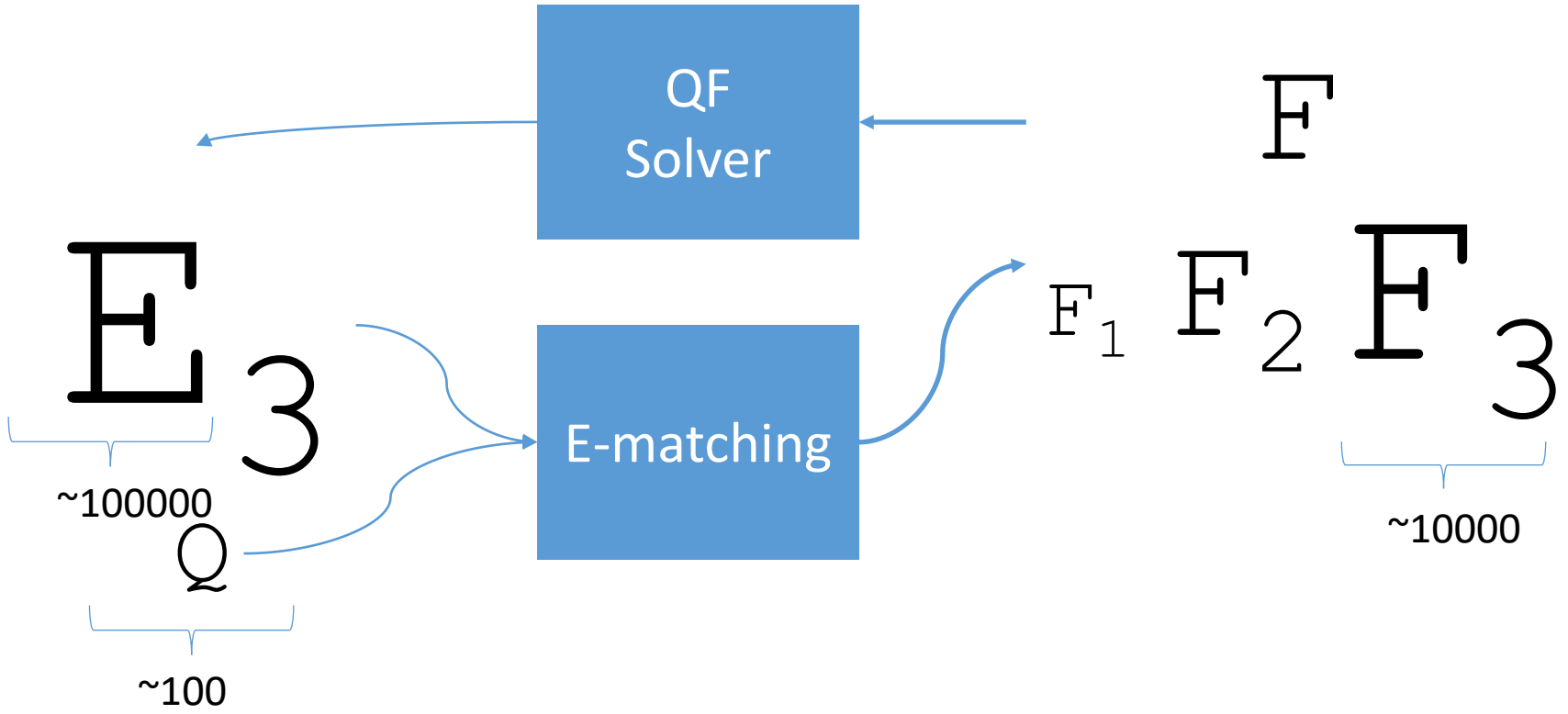
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Contexts contain 1000s of terms in $\mathbb{E}$, 100s of $\forall$ in $\mathbb{Q}$
  - Leads to 100s

Challenge #1 : Too Many Instances



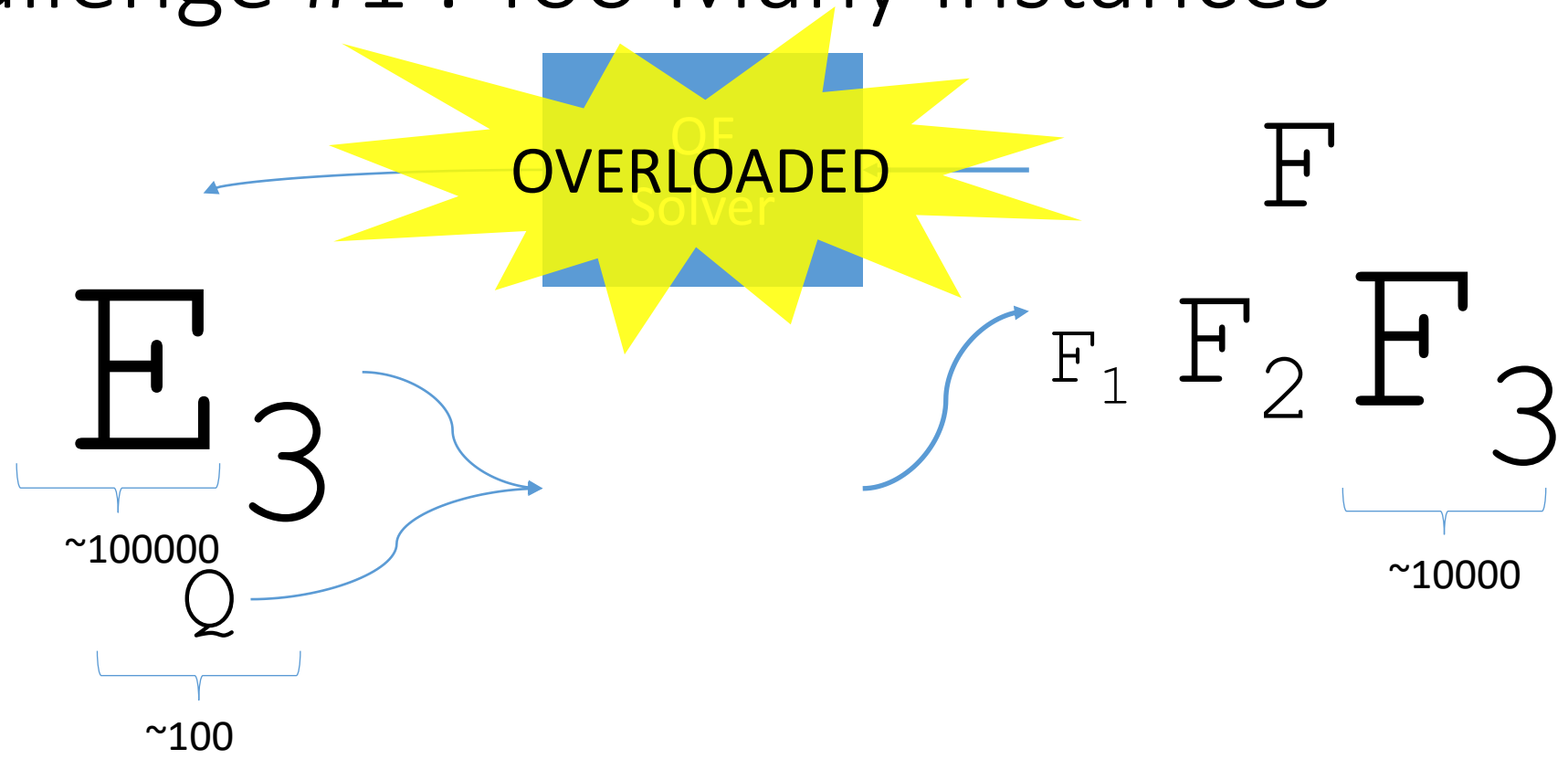
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Contexts contain 1000s of terms in $\mathbb{E}$, 100s of $\forall$ in $\mathbb{Q}$
  - Leads to 100s, 1000s

Challenge #1 : Too Many Instances



- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Contexts contain 1000s of terms in $\mathbb{E}$, 100s of $\forall$ in $\mathbb{Q}$
  - 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

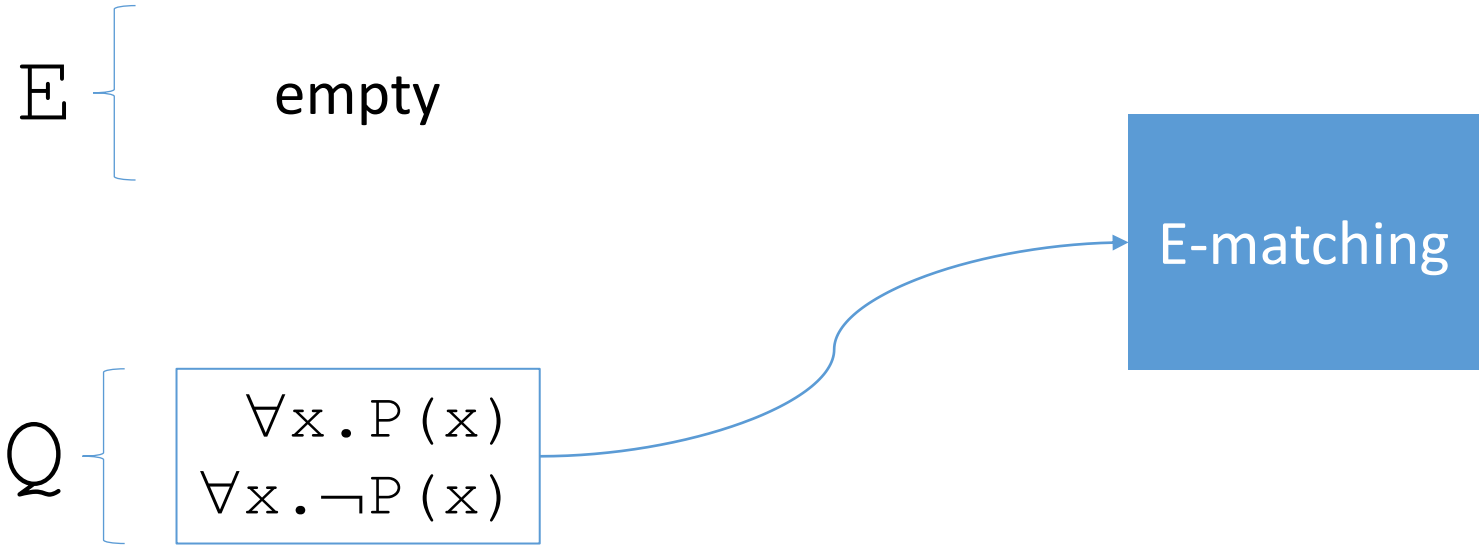
# Instances	cvc3		cvc4		z3	
	#	%	#	%	#	%
1-10	1464	13.49%	1007	8.87%	1321	11.43%
10-100	1755	16.17%	1853	16.31%	2554	22.11%
100-1000	**3816**	35.16%	**3680**	32.40%	**4553**	39.41%
1000-10k	1893	17.44%	2468	21.73%	1779	15.40%
10k-100k	1162	10.71%	1414	12.45%	823	7.12%
100k-1M	560	5.16%	607	5.34%	376	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**

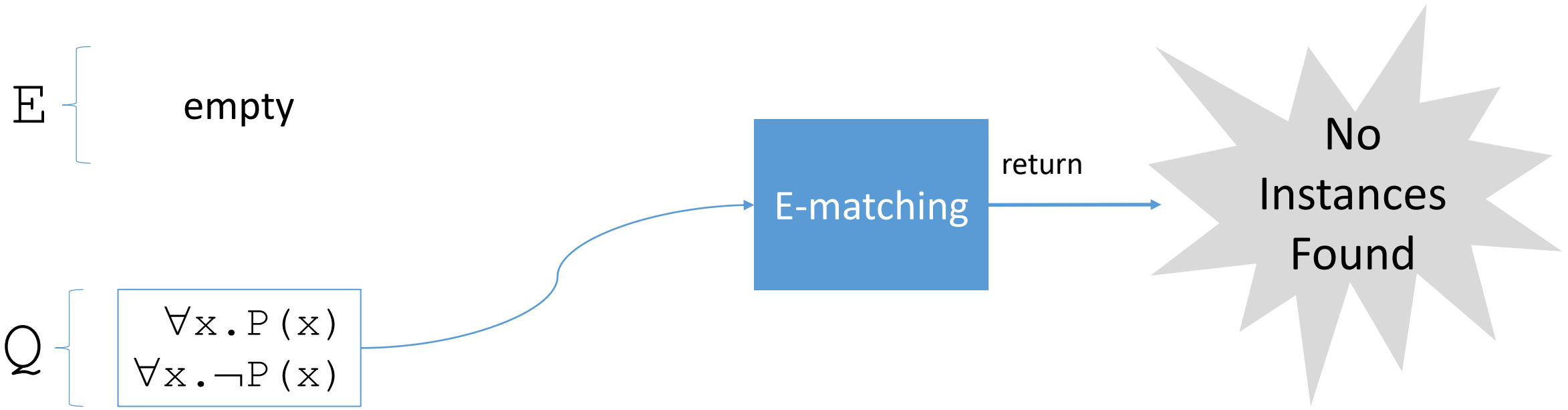
    (Above, 16.6% required >10k, max 19.5M by z3 on a software verification benchmark from TPTP)

Challenge #2 : Incompleteness



⇒ E-matching is an incomplete procedure

Challenge #2 : Incompleteness

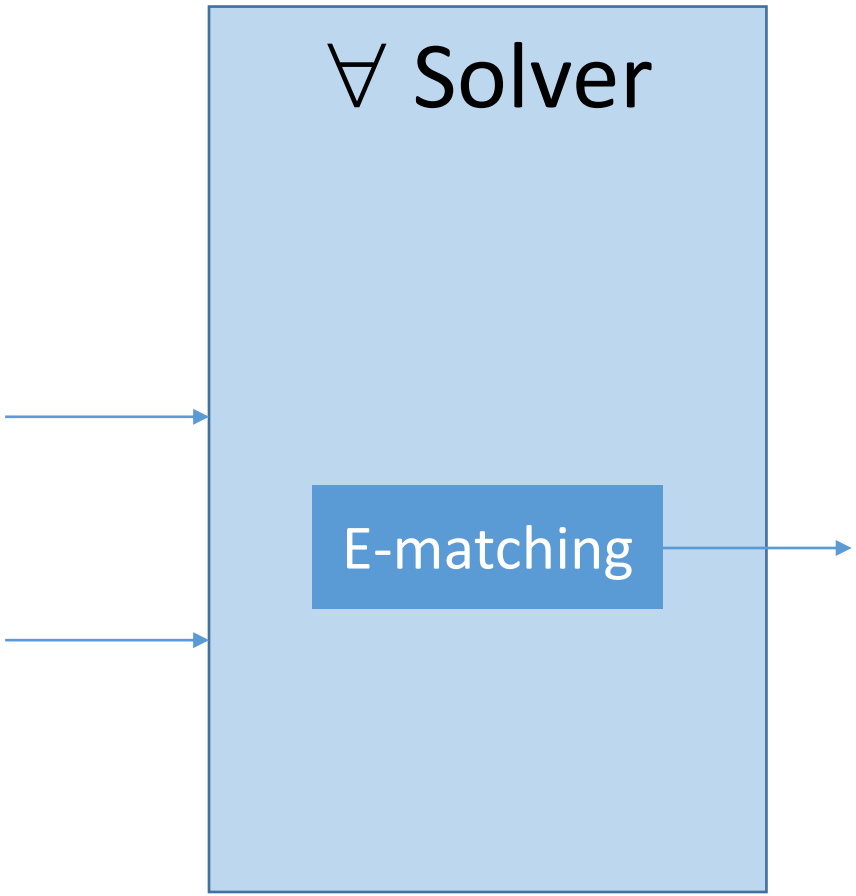


$\Rightarrow$ If E-matching produces no instances,
this *does not guarantee $E \cup Q$ is T-satisfiable*

E-matching : Challenges Addressed

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

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

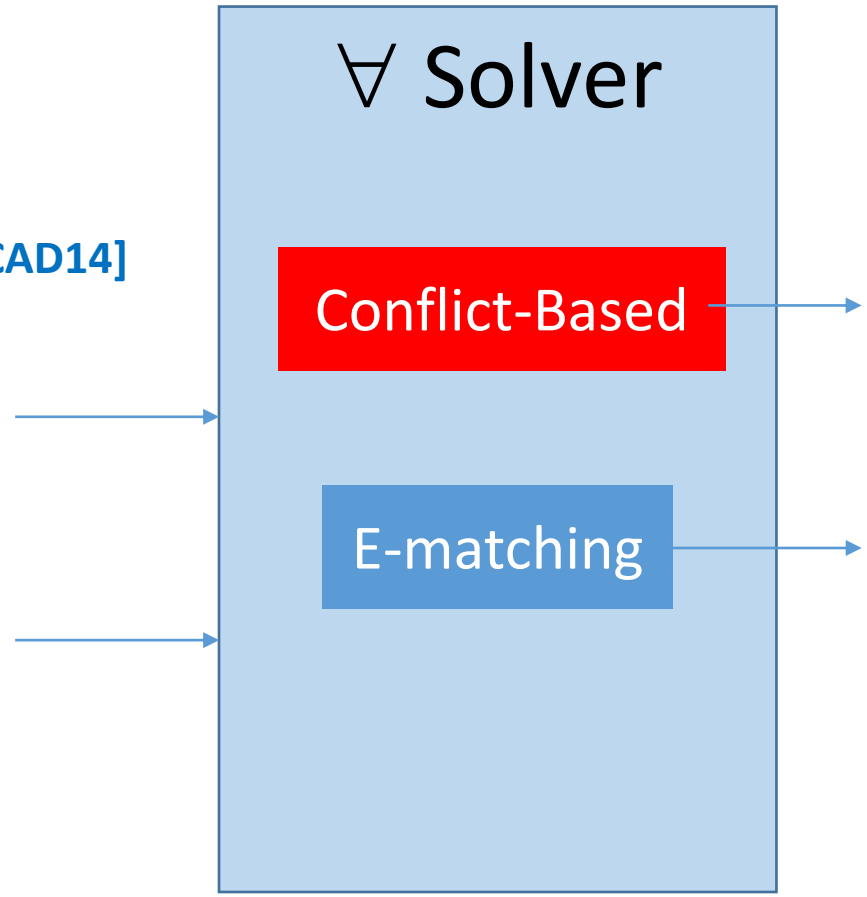


E-matching : Challenges Addressed

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

  $\Rightarrow$*Use conflict-based instantiation* [Reynolds et al FMCAD14]

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



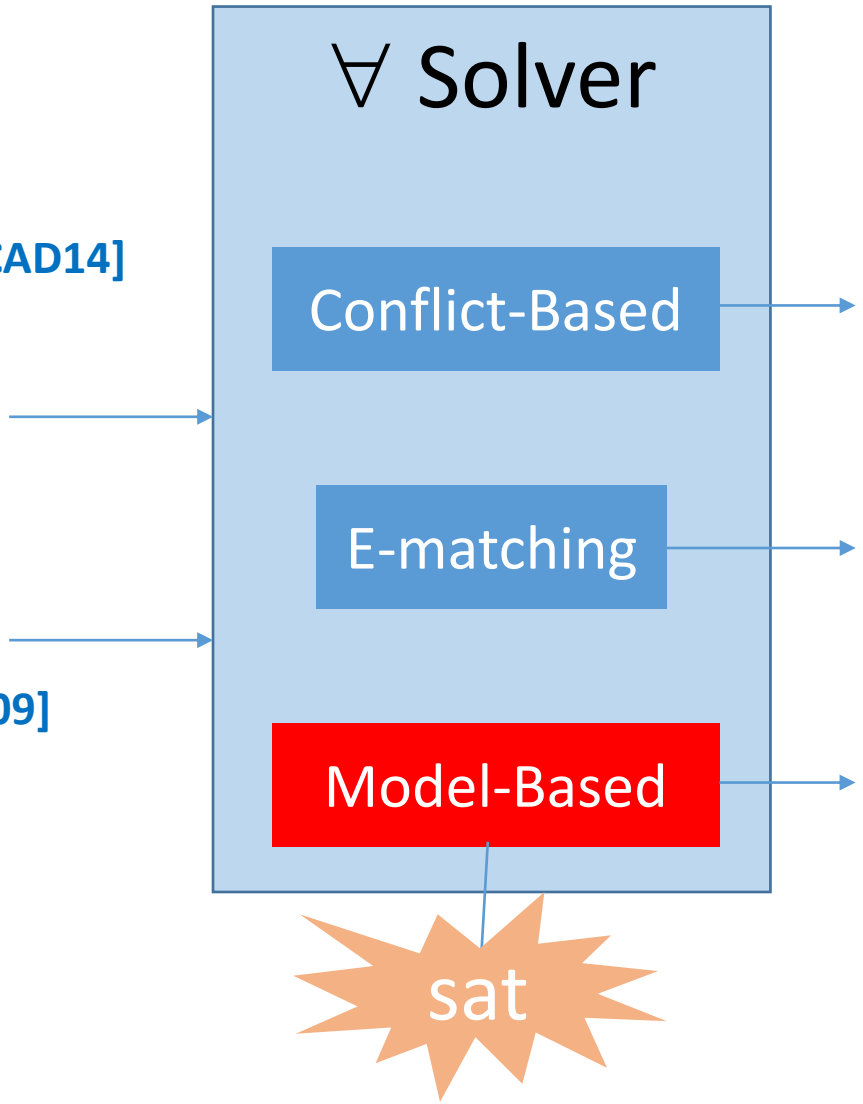
E-matching : Challenges Addressed

- What if there are too many instances?

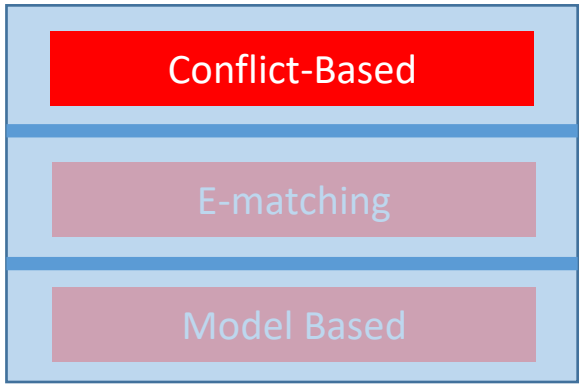
  $\Rightarrow$Use conflict-based instantiation [Reynolds et al FMCAD14]

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

  $\Rightarrow$*Use model-based instantiation* [Ge/deMoura CAV2009]



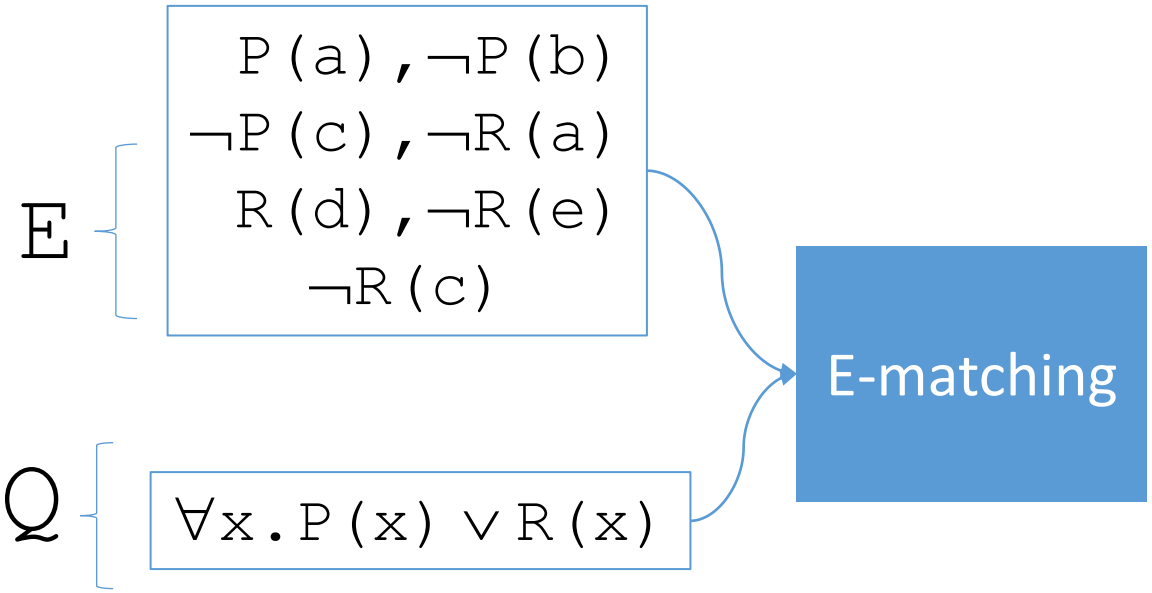
Conflict-Based Instantiation

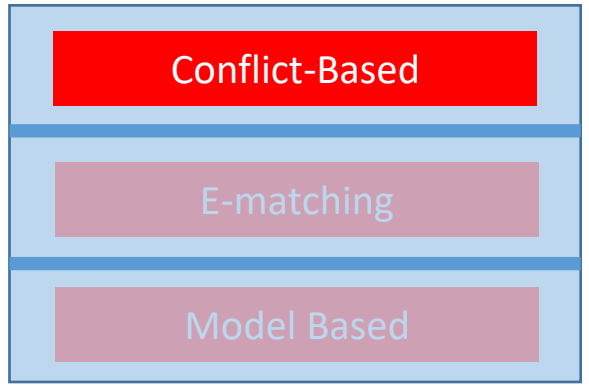


- Basic idea:
  - Since we are interested in whether e.g. $\mathtt{E}, \forall \mathtt{x}.\mathtt{P}(\mathtt{x})$ is satisfiable,
    - Try to find one "conflict instance" such that $\mathtt{E}, \mathtt{P}(\mathtt{a}) \models \bot$
    - If this is possible, don't run E-matching

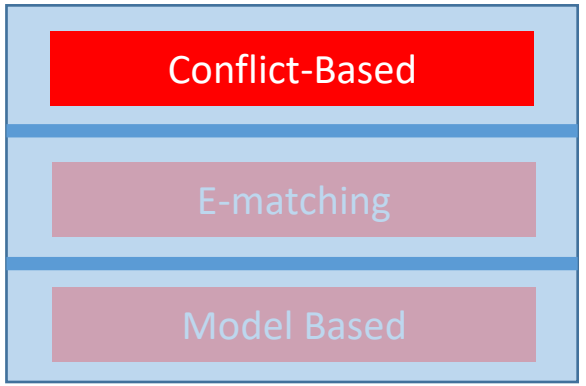
$\Rightarrow$ *Leads to fewer instances, improved ability to answer* unsat

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

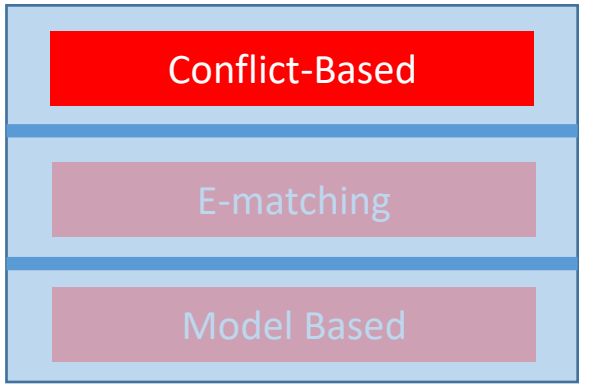
$\forall x.P(x) \vee R(x)$

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

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

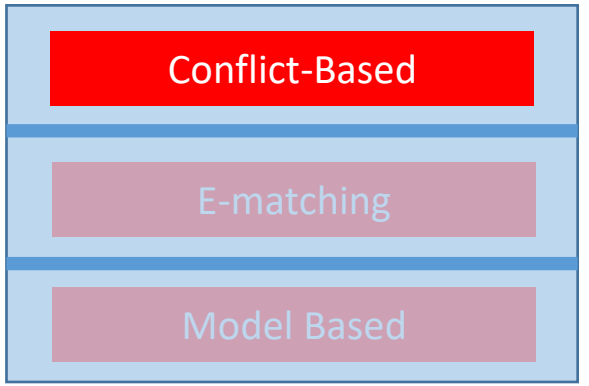
E-matching

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

$\Rightarrow$ 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)**, ¬P(b), ¬P(c), ¬R(a), R(d), ¬R(e), ¬R(c) }

Q { ∀x.P(x) ∨ R(x) }

E-matching

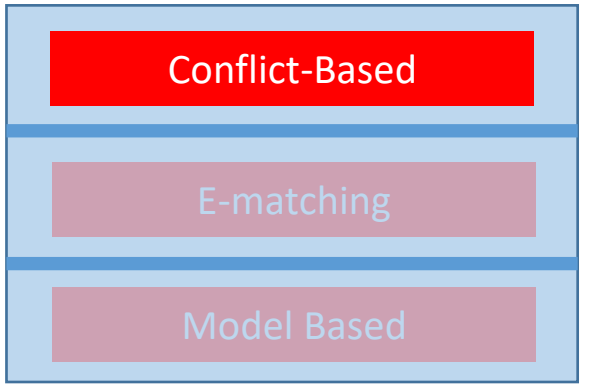
(∀x.P(x) ∨ R(x)) ⇒ P(a) ∨ R(a)
(∀x.P(x) ∨ R(x)) ⇒ P(b) ∨ R(b)
(∀x.P(x) ∨ R(x)) ⇒ P(c) ∨ R(c)
(∀x.P(x) ∨ R(x)) ⇒ P(d) ∨ R(d)
(∀x.P(x) ∨ R(x)) ⇒ P(e) ∨ R(e)

⇒ Consider what we learn from these instances:

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

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

Conflict-Based Instantiation



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

Q { $\forall x.P(x) \vee R(x)$ }

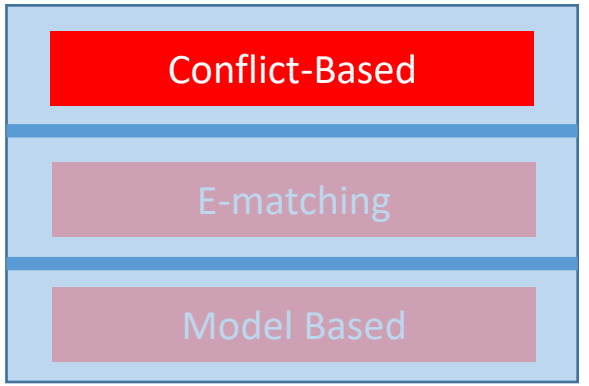
E-matching

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

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

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

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

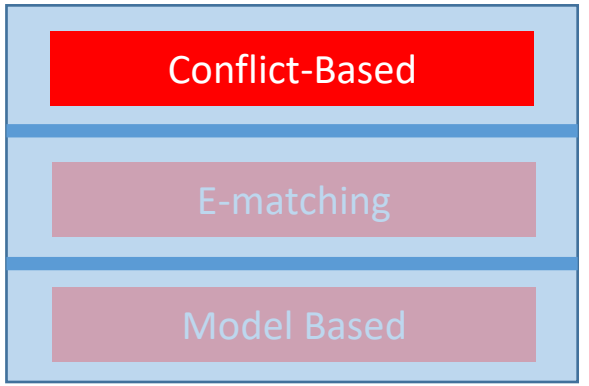
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

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

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

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

Conflict-Based Instantiation



E

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

Q

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

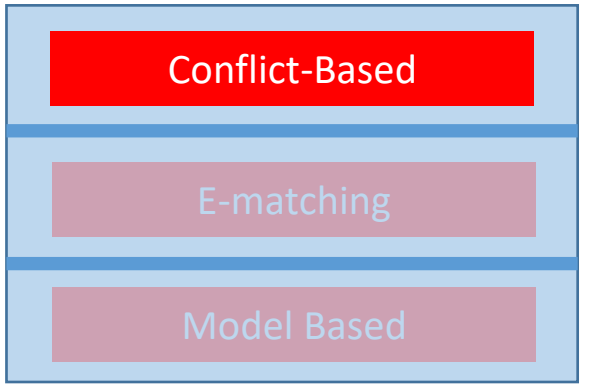
E-matching

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

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

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

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

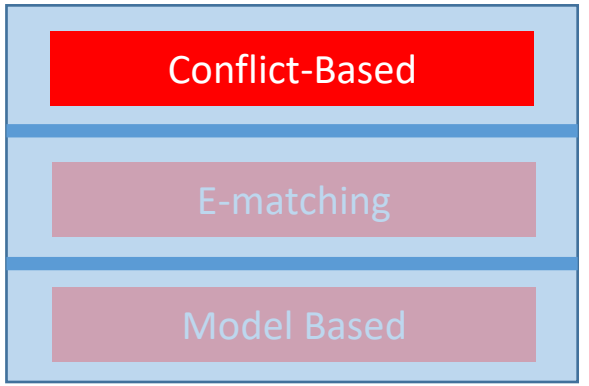
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

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

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

We know $P(c) \Leftrightarrow \bot$

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

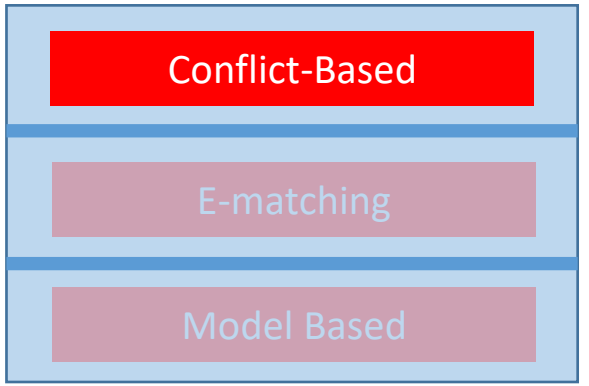
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

$\Rightarrow$ 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

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

Q

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

E-matching

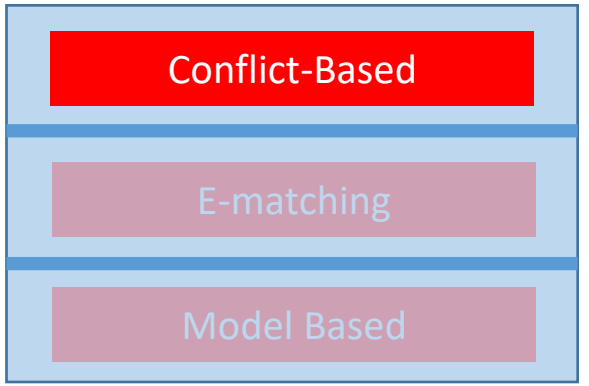
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

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

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

We know $\mathbf{R(e)} \Leftrightarrow \bot$

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

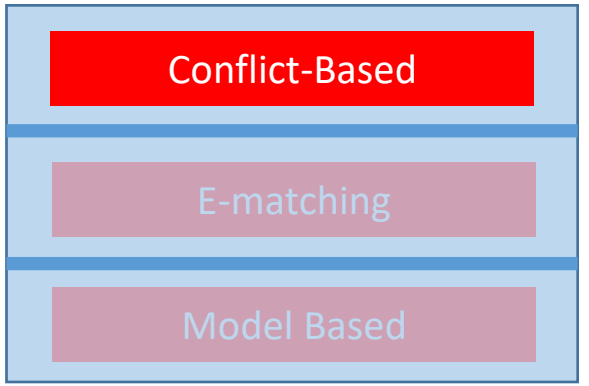
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

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

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

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

Conflict-Based Instantiation



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

Q { $\forall x.P(x) \vee R(x)$ }

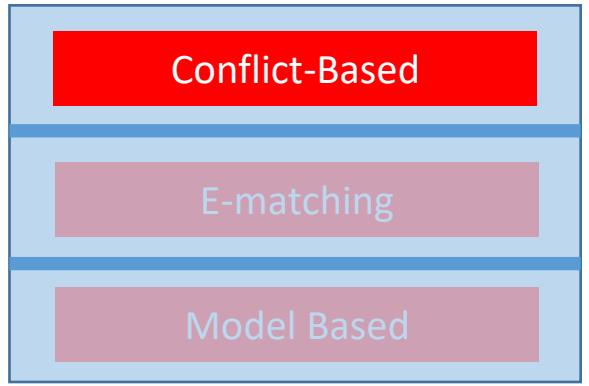
E-matching

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

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

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

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

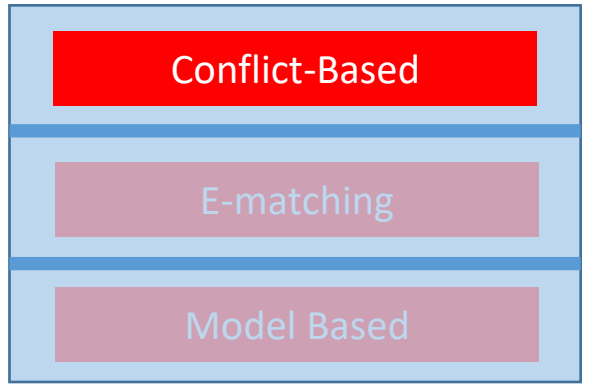
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$$
$$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$$

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

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

*$P(c) \vee R(c)$ 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

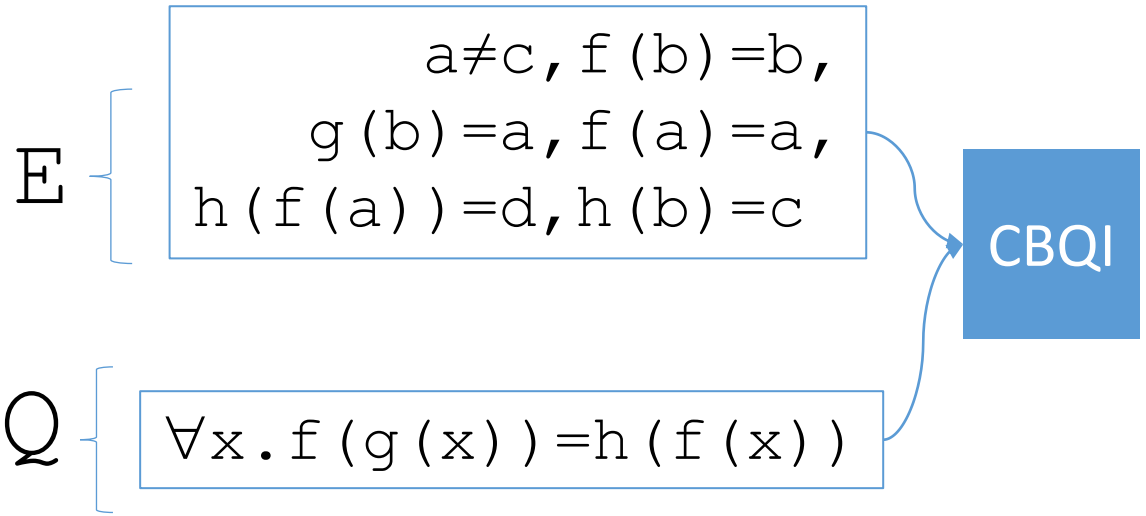
$(\forall x.P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

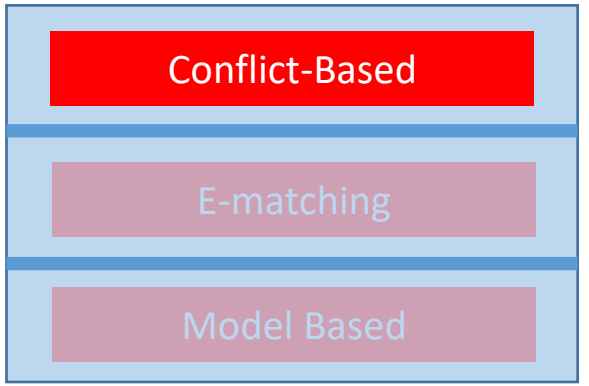
$\Rightarrow$ Consider what we learn from these instances:

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

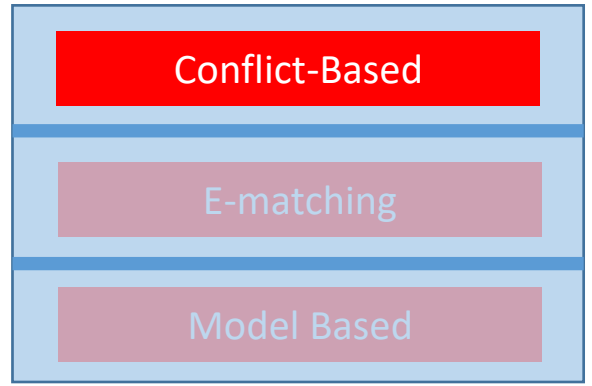
Since $P(c) \vee R(c)$ suffices to derive $\bot$, return ***only*** this instance

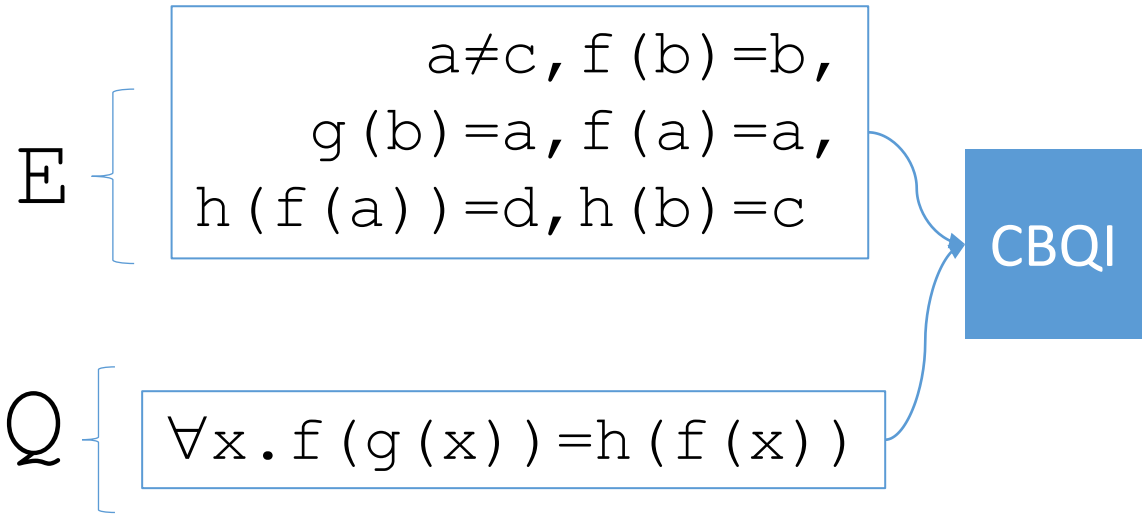
Conflict-Based Instantiation: EUF





Conflict-Based Instantiation: EUF

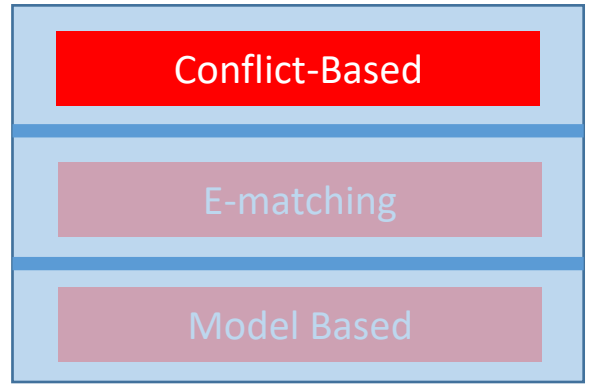


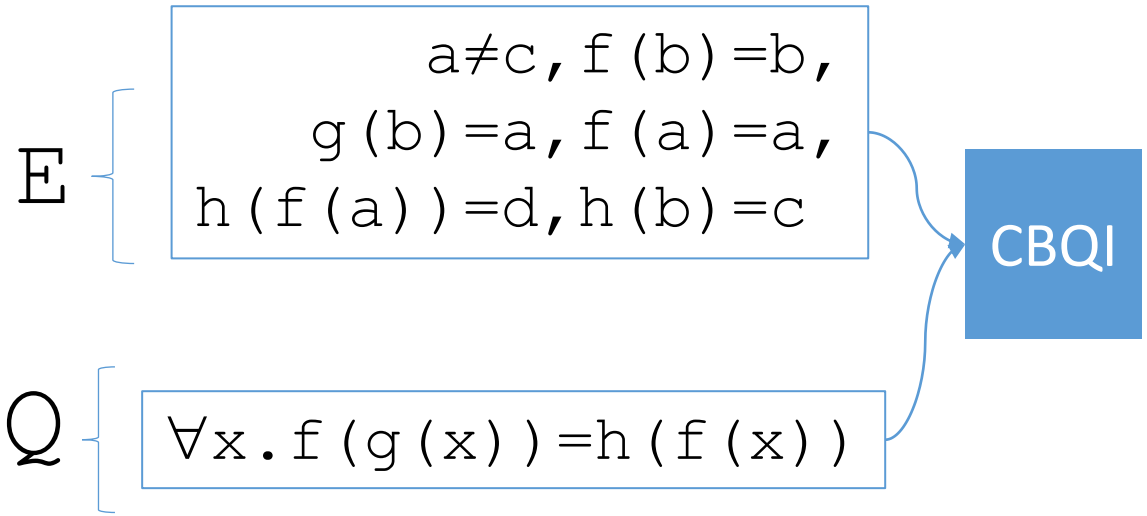


⇒ Consider the instance $\forall x.\, f(g(x)) = h(f(x)) \Rightarrow f(g(\mathbf{b})) = h(f(\mathbf{b}))$

- Is this conflicting for (E, Q) ?

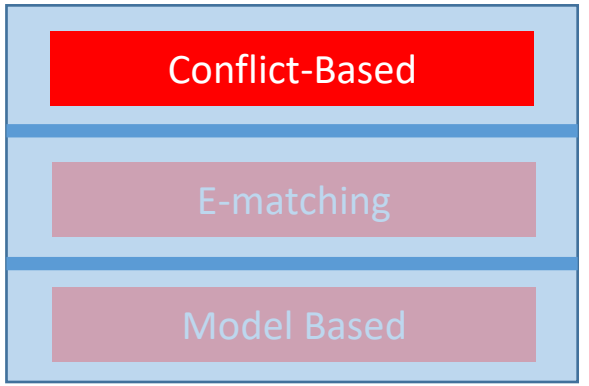
Conflict-Based Instantiation: EUF





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

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

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

$b=f(b)$

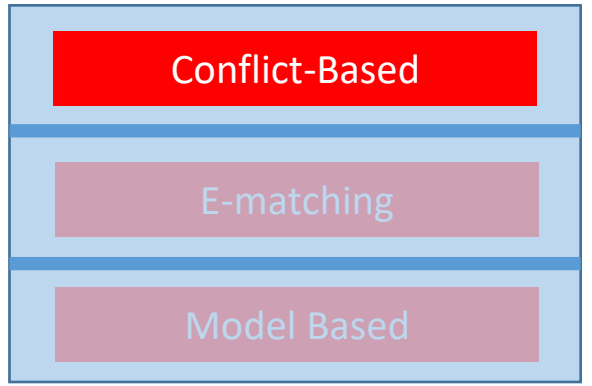
$c=h(b)$

$d=h(f(a))$

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

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

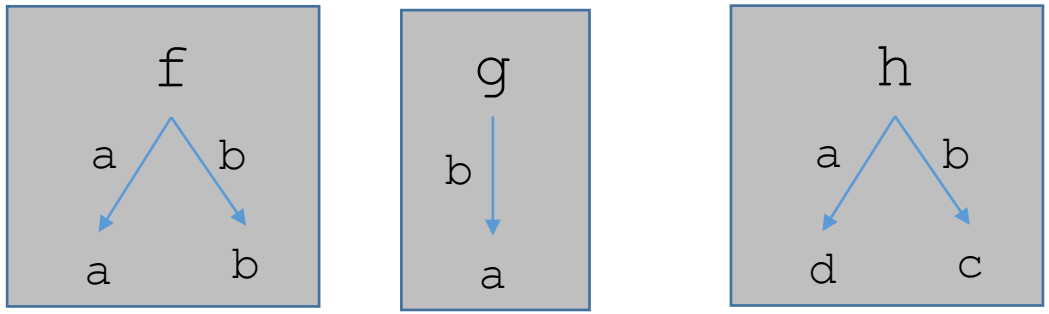
CBQI

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

b$=f(b)$

c$=h(b)$

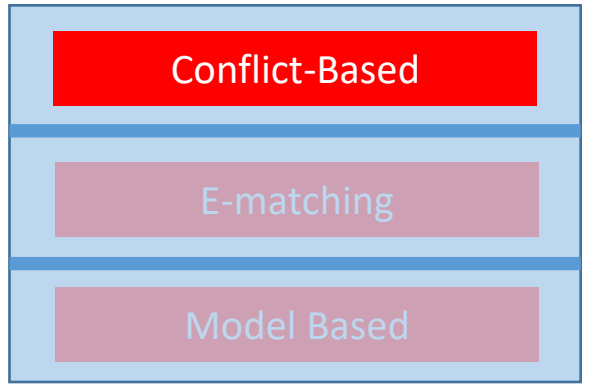
d$=h(f(a))$



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

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

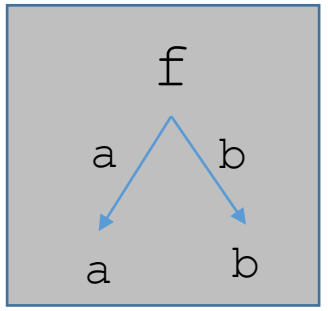
CBQI

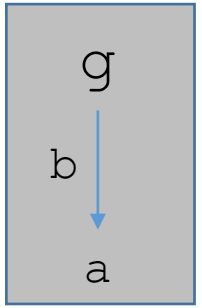
$a=g(b)=f(a)$

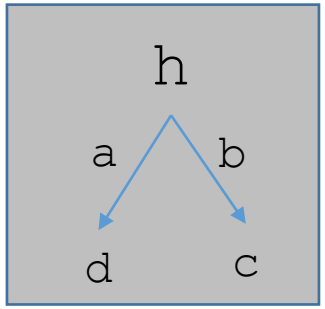
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

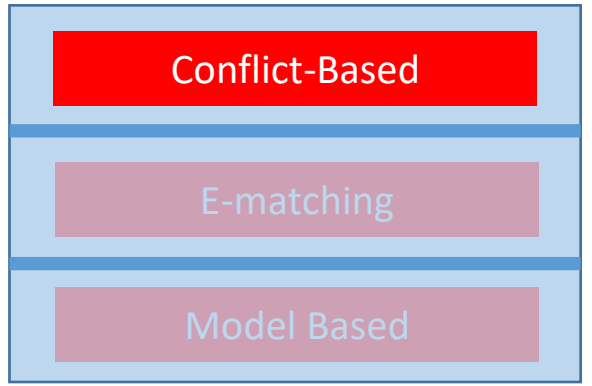






$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$

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

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

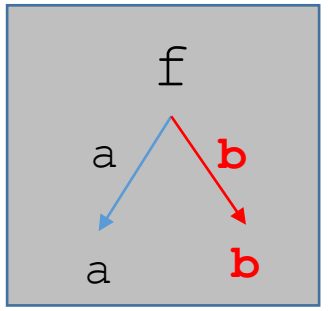
CBQI

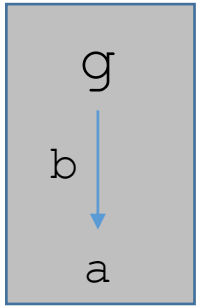
$a=g(b)=f(a)$

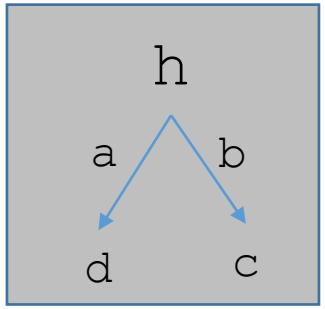
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

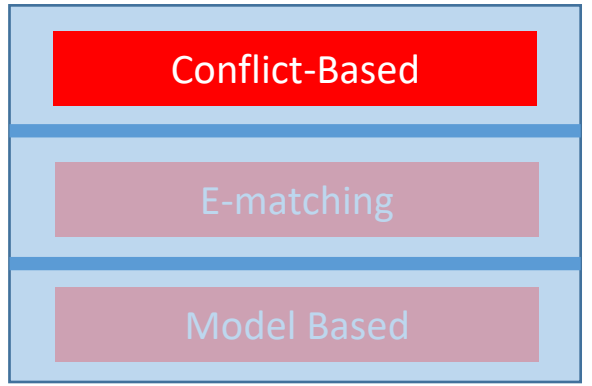






$E, f(g(b))=h(f(b)) \models_E f(g(b))=h(\ \mathbf{b}\ )$

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

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

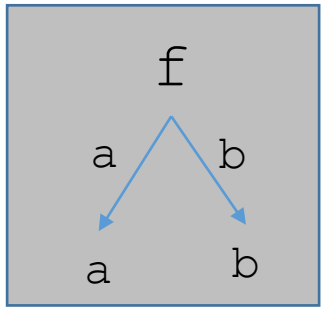
CBQI

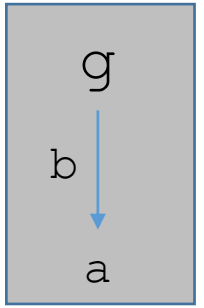
$a=g(b)=f(a)$

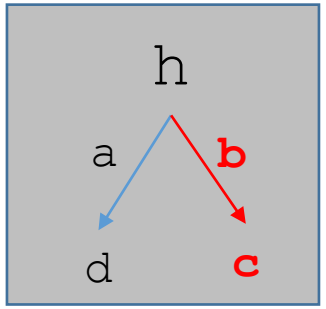
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

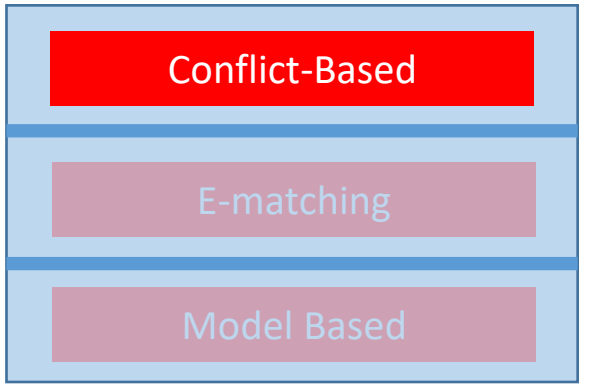






$E, f(g(b))=h(f(b)) \models_E f(g(b))=$ **c**

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

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

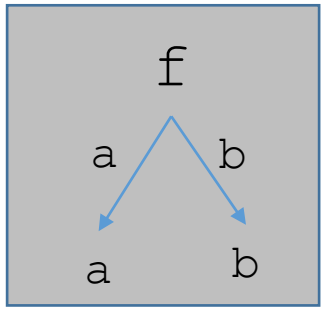
CBQI

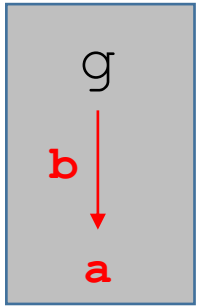
$a=g(b)=f(a)$

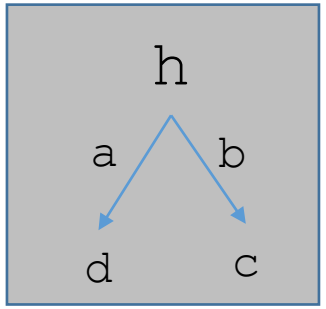
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

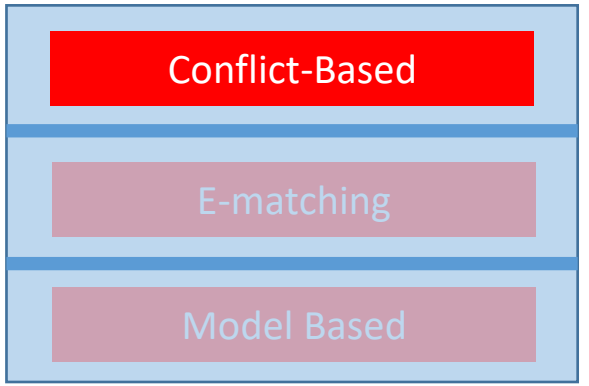






$E, f(g(b))=h(f(b)) \models_E f(\ \mathbf{a}\ )=\ c$

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

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

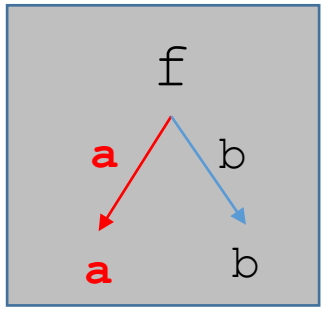
CBQI

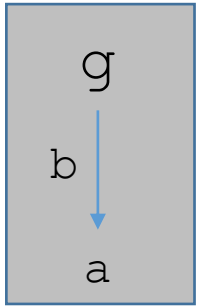
$a=g(b)=f(a)$

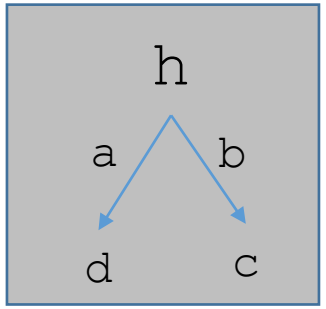
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

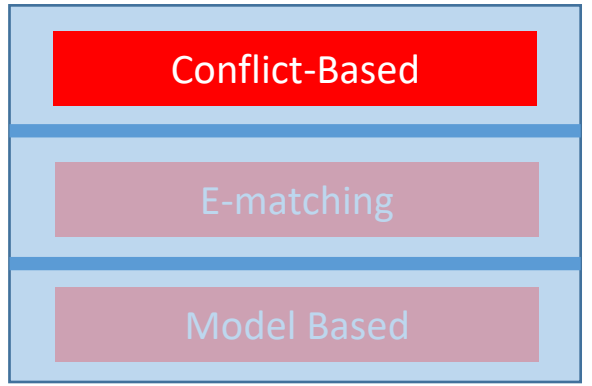






$E, f(g(b))=h(f(b)) \models_E a = c$

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

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

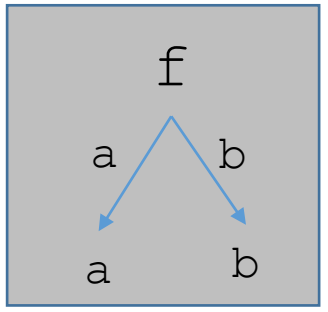
CBQI

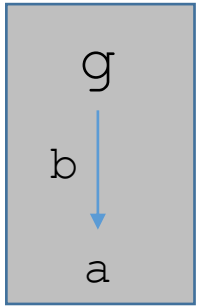
$a=g(b)=f(a)$

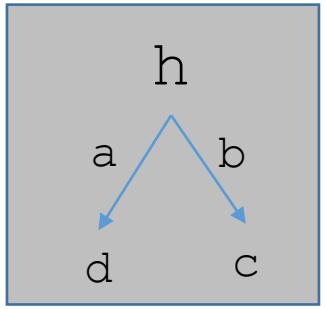
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

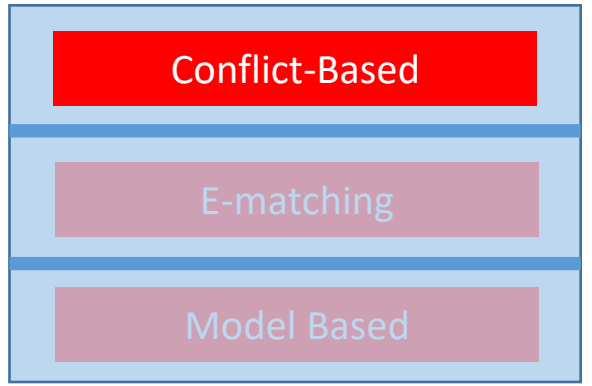






$E, f(g(b))=h(f(b)) \models_E \quad a=c$

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

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

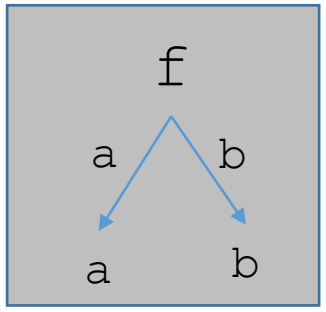
CBQI

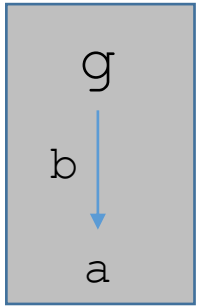
$a=g(b)=f(a)$

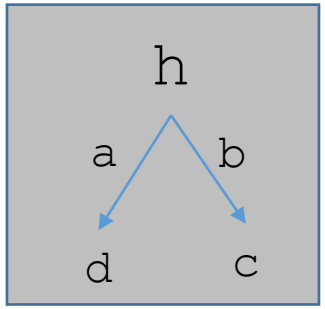
$b=f(b)$

$c=h(b)$

$d=h(f(a))$



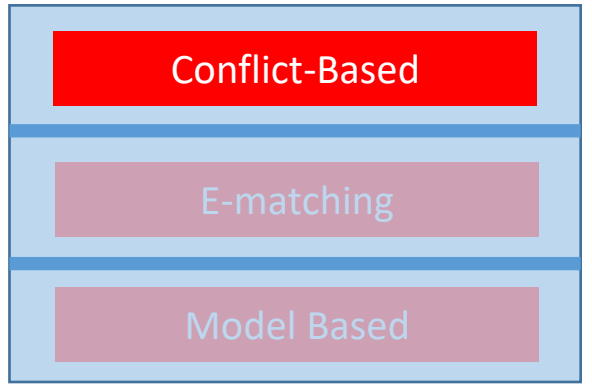




$E, f(g(b))=h(f(b)) \models_E \bot$

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

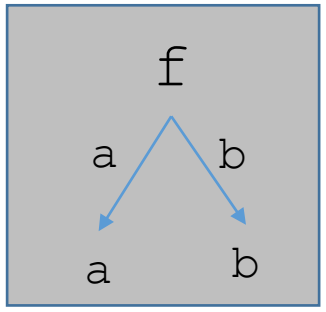
CBQI

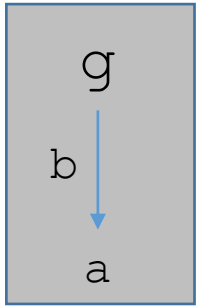
$a=g(b)=f(a)$

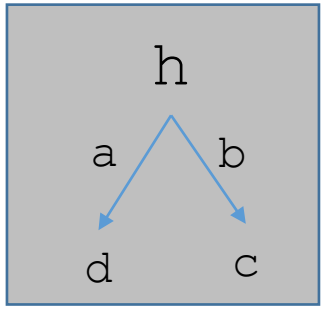
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

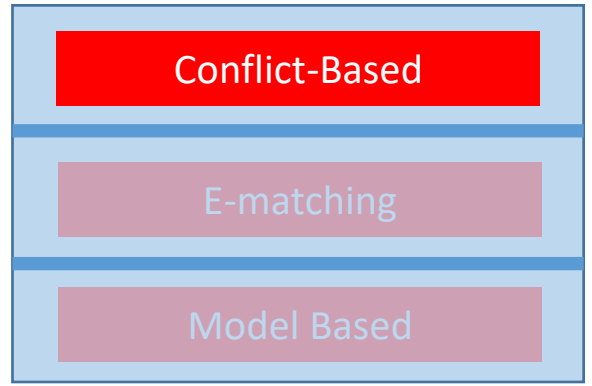


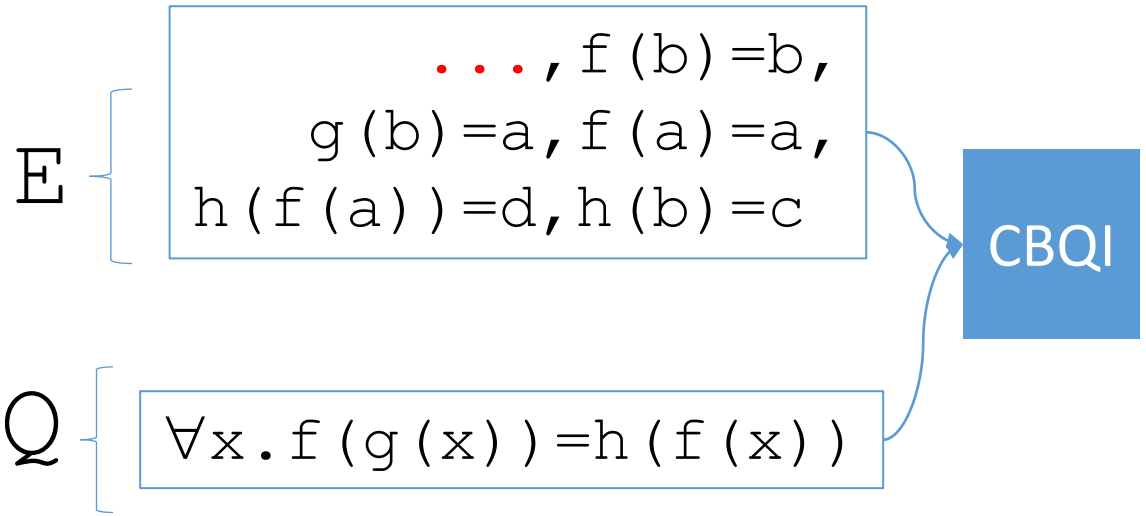




$E, f(g(b))=h(f(b)) \models_E \bot$ { $f(g(b))=h(f(b))$ is a conflicting instance for (E, Q) !

Conflict-Based Instantiation: EUF

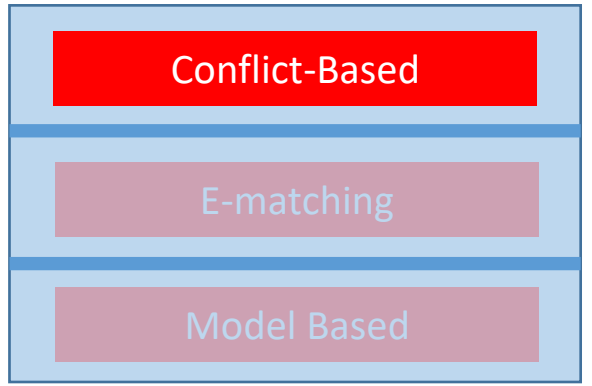




⇒ Consider the same example, but where we don't know a≠c

- 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

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

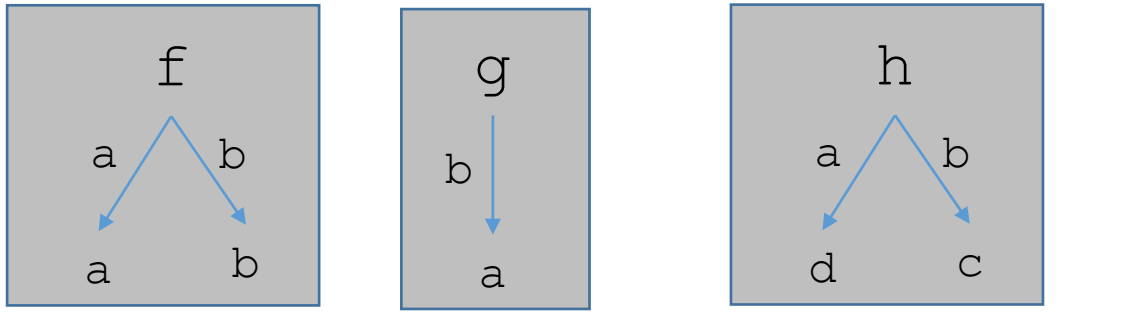
CBQI

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

$b=f(b)$

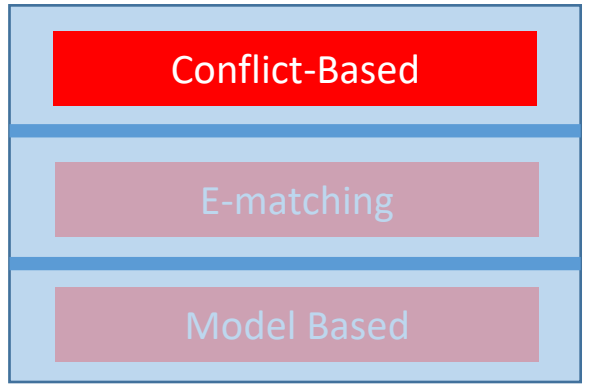
$c=h(b)$

$d=h(f(a))$



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

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

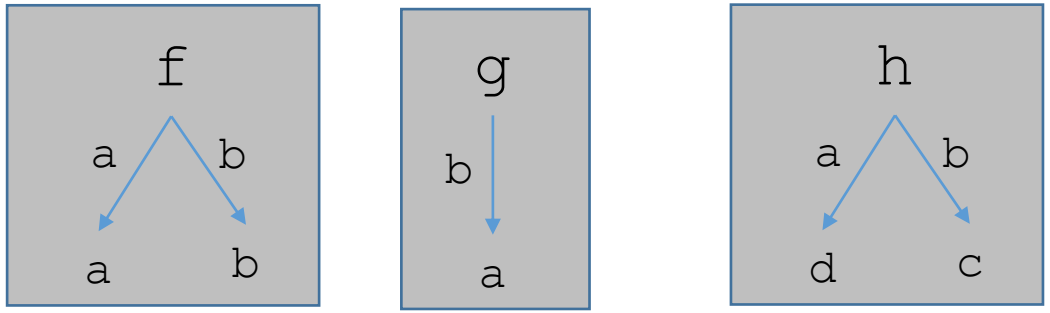
CBQI

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

$b=f(b)$

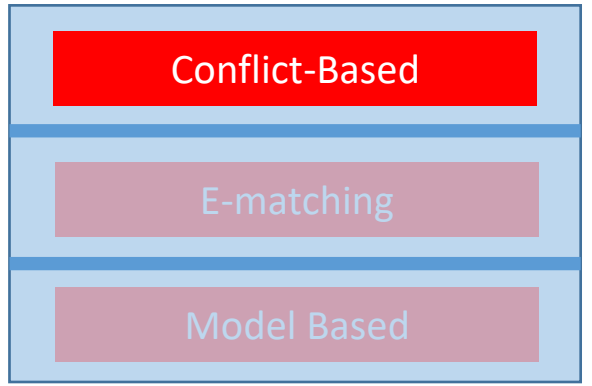
$c=h(b)$

$d=h(f(a))$



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

Conflict-Based Instantiation: EUF



E

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

Q

∀x.f(g(x))=h(f(x))

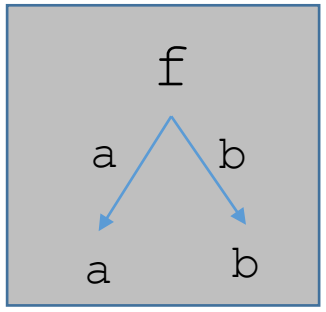
CBQI

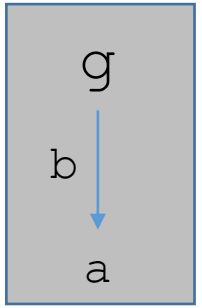
a=g(b)=f(a)

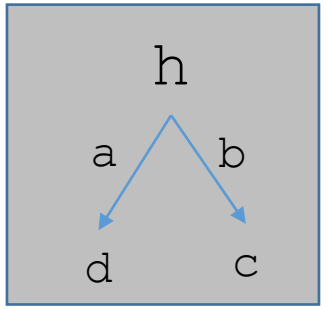
b=f(b)

c=h(b)

d=h(f(a))

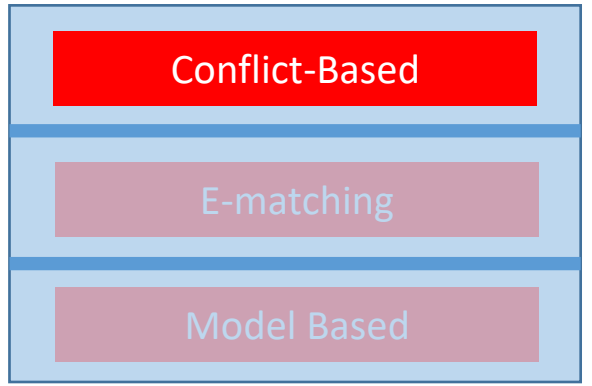






$E, f(g(b))=h(f(b)) \models_E a=c$

Conflict-Based Instantiation: EUF



E { ..., f(b)=b, g(b)=a, f(a)=a, h(f(a))=d, h(b)=c }

Q { ∀x. f(g(x))=h(f(x)) }

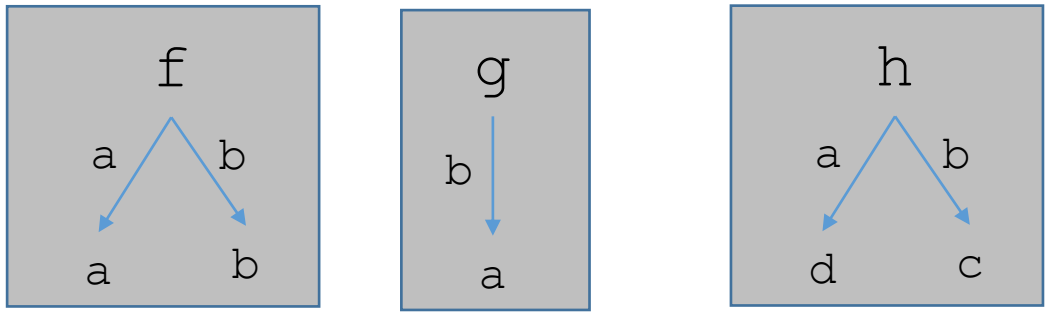
CBQI

a=g(b)=f(a)

b=f(b)

c=h(b)

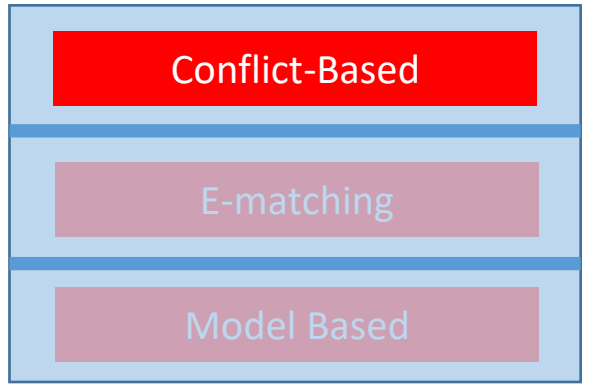
d=h(f(a))



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

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

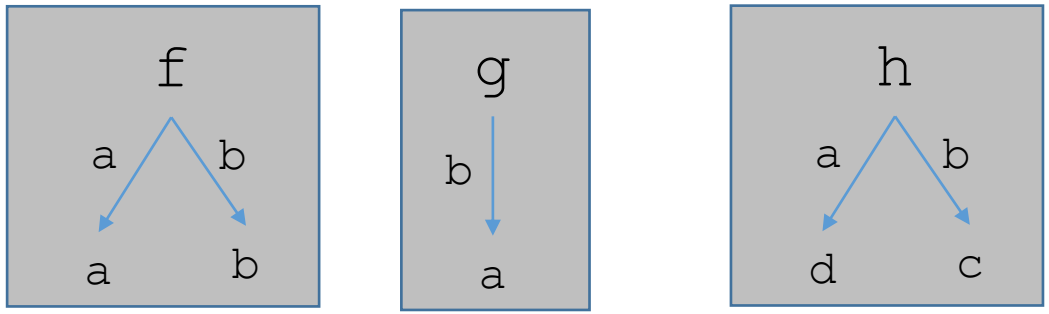
CBQI

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

$b=f(b)$

$c=h(b)$

$d=h(f(a))$

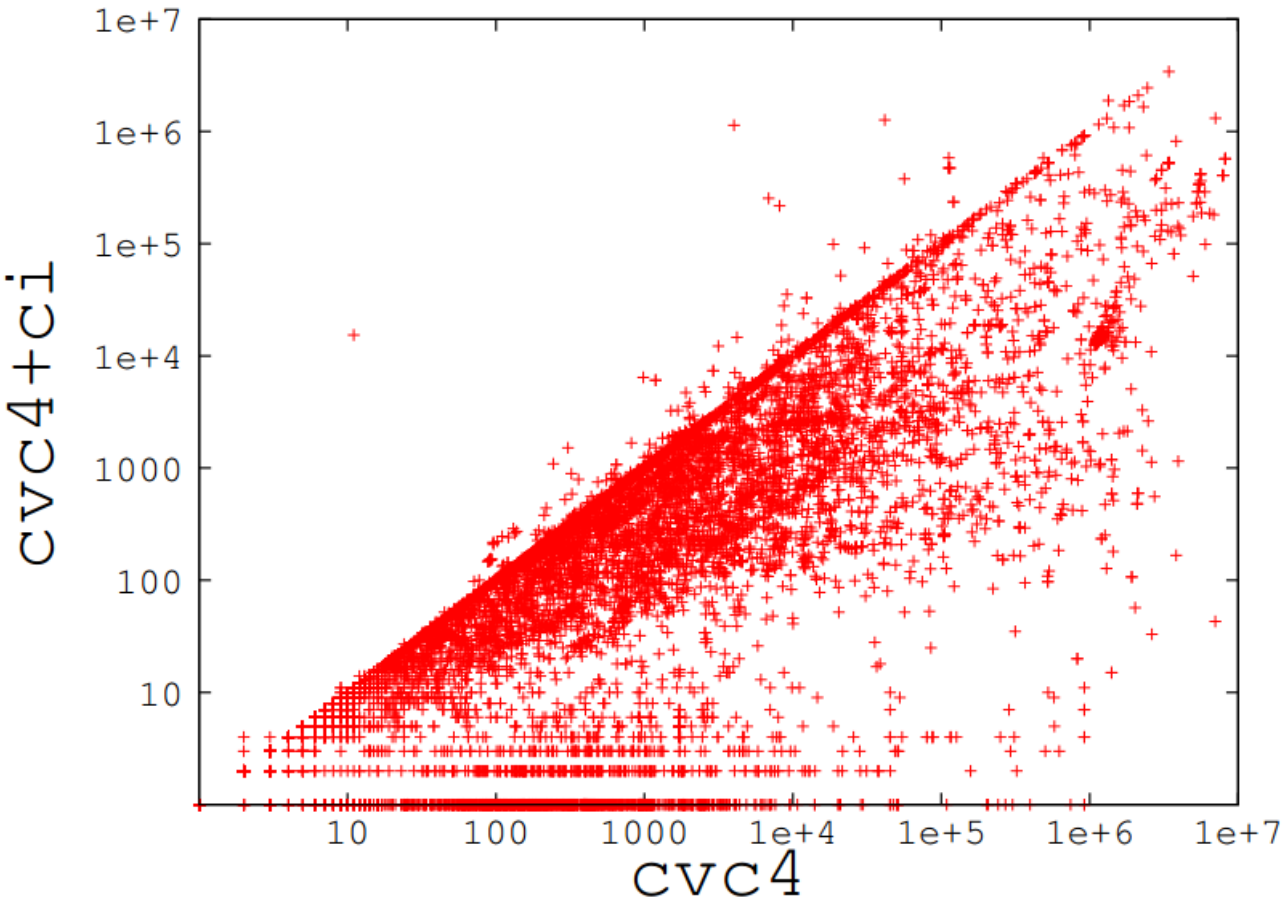


$f(g(b))=h(f(b))$ is a propagating instance for (E, Q)

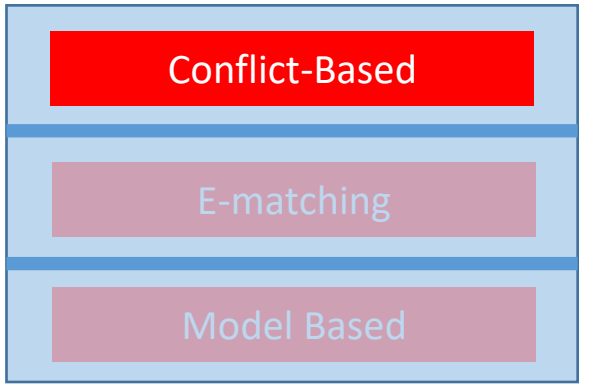
$\Rightarrow$ *These are also useful*

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

Conflict-Based Instantiation: Impact



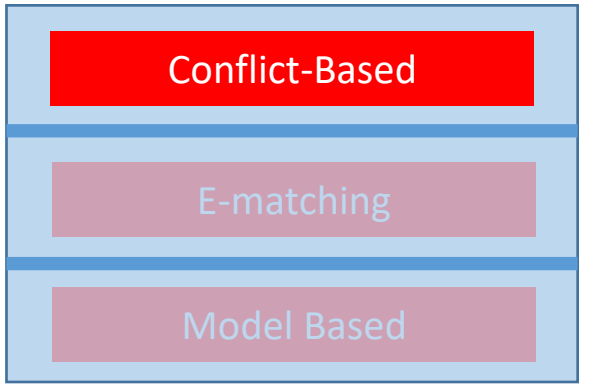
Reported number of instances.



- 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)

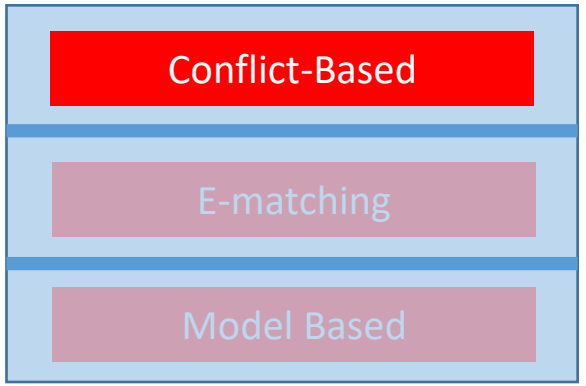
Conflict-Based Instantiation: Impact



- 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

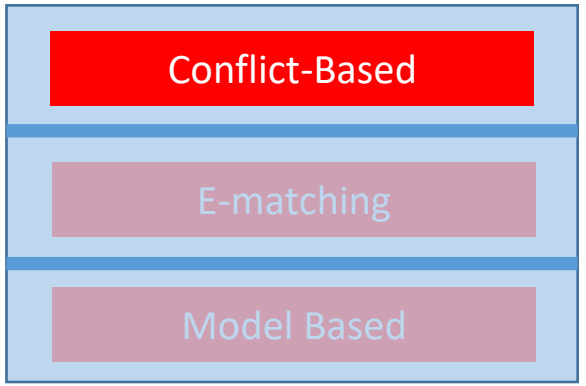


- CVC4 with conflicting instances **`cvc4+ci`**
  - Solves the most benchmarks for TPTP and Isabelle
  - Requires almost an order of magnitude fewer instantiations

	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

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

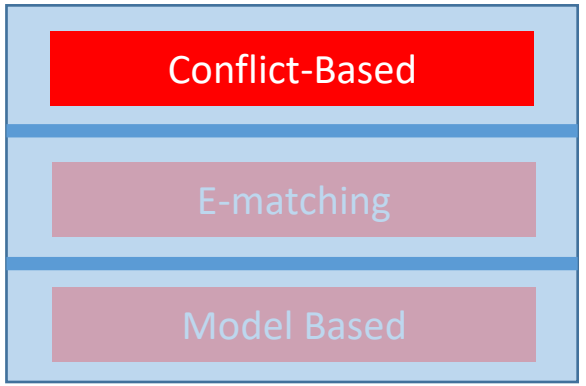
Challenge : Finding Conflicting Instances



- 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

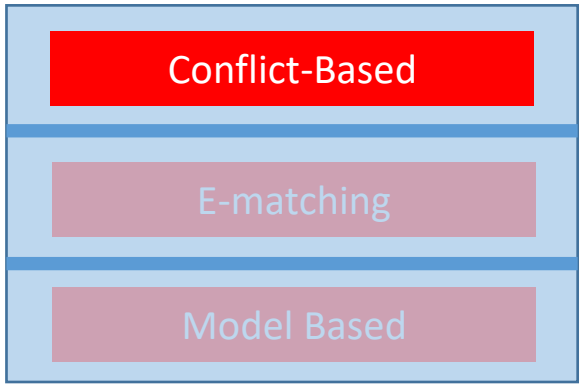


- Difficult for quantified formulas that contain *theory symbols:*

E — `f(1)=5`

Q — `∀xy.f(x+y)>x+2*y`

Challenge : Theory Symbols



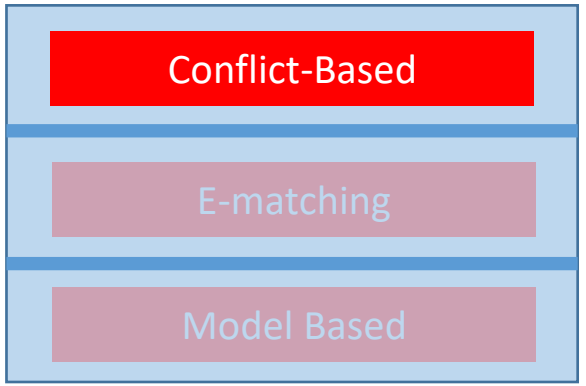
- Difficult for quantified formulas that contain *theory symbols:*

$\mathbb{E}$ `f(1)=5`

$\mathbb{Q}$ `∀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:*

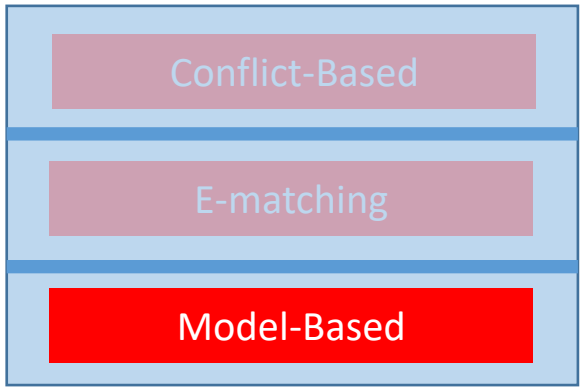
$\mathbb{E}$ `f(1)=5`

$\mathbb{Q}$ `∀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

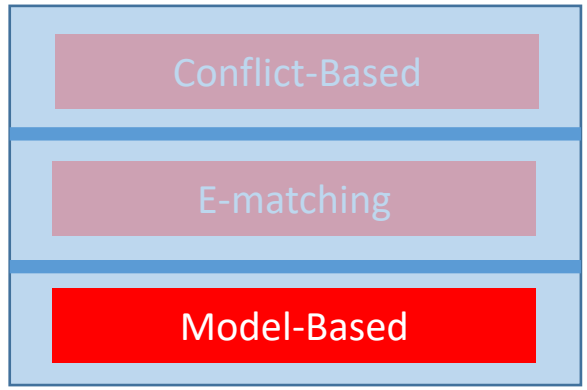


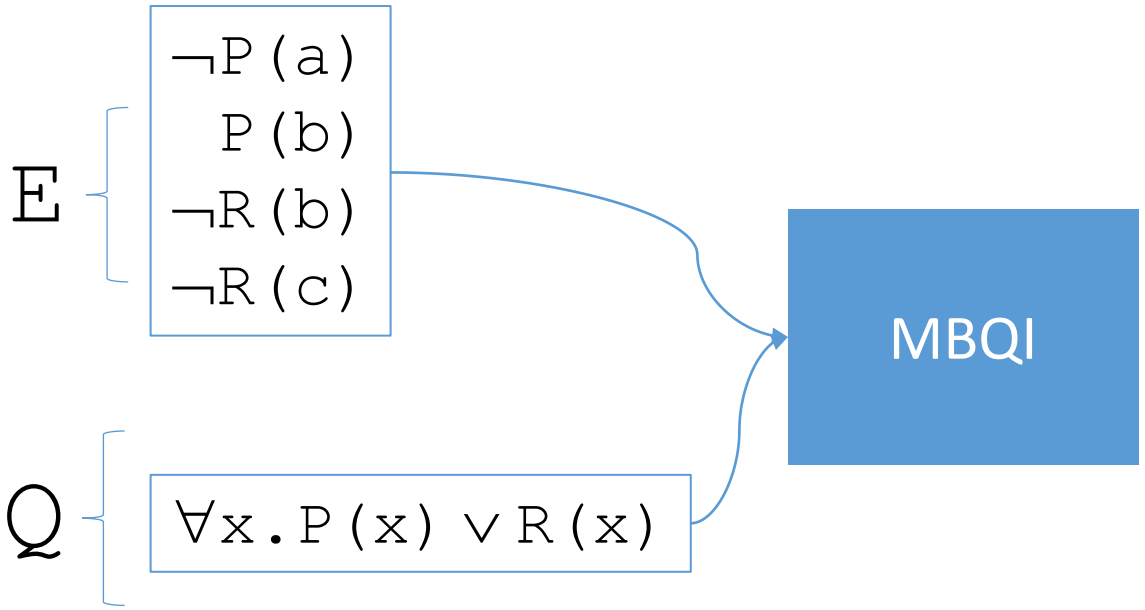
- Basic idea:
  - If E-matching saturates, build "candidate model" M satisfying $\mathbb{E}$
    - Check if M also satisfies Q

      (using a quantifier-free satisfiability query)

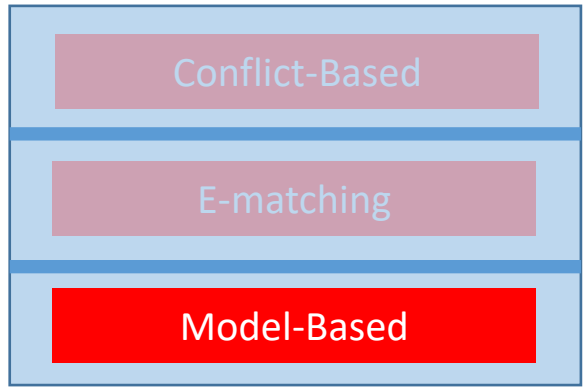
$\Rightarrow$ *Ability to answer* sat

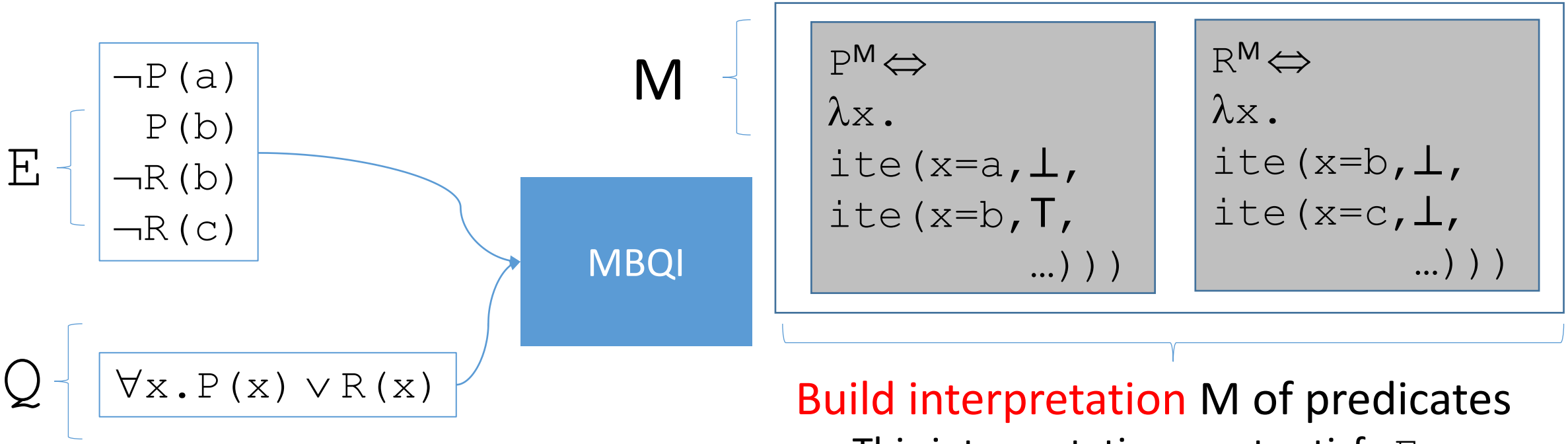
Model-based Instantiation





Model-based Instantiation

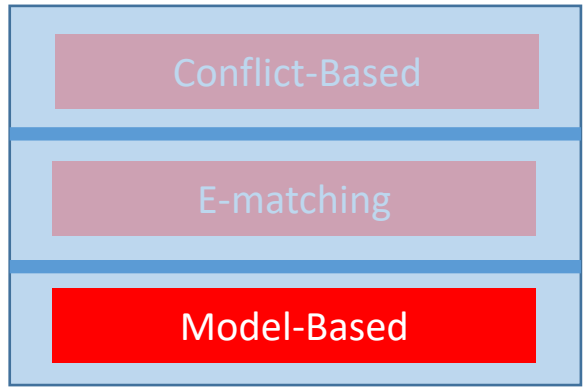


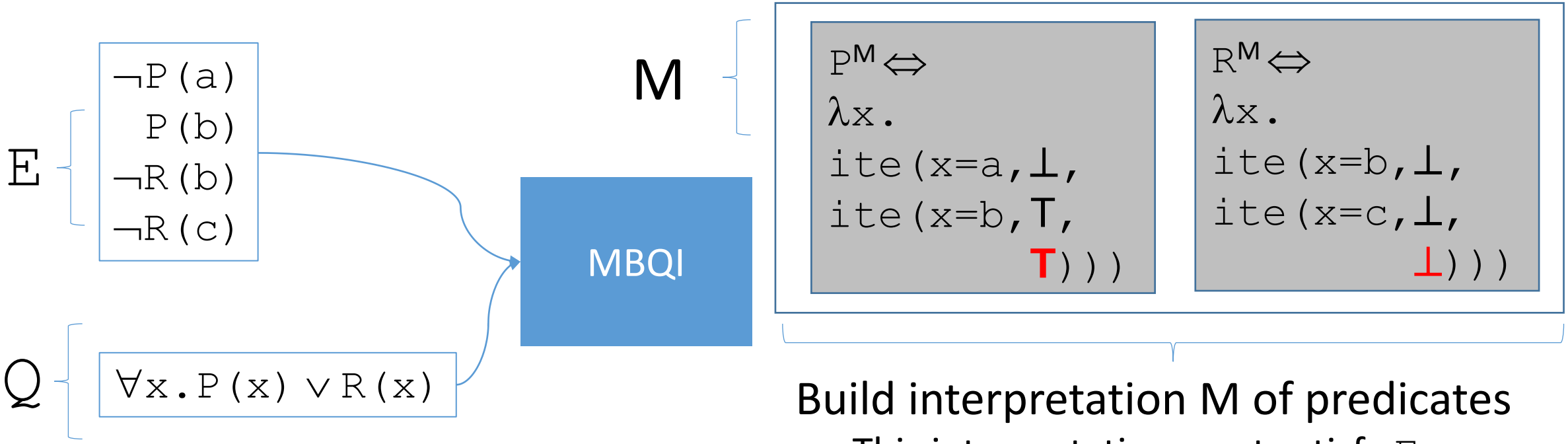


Build interpretation M of predicates

- This interpretation must satisfy E

Model-based Instantiation

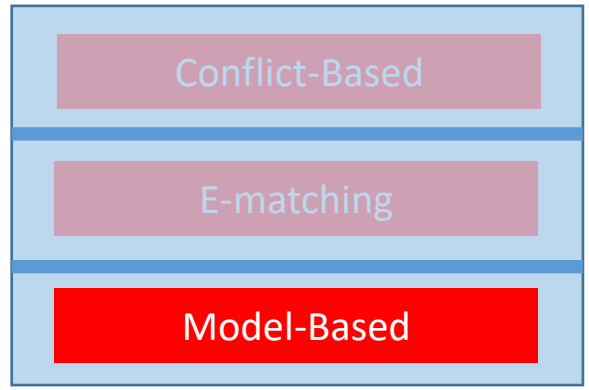


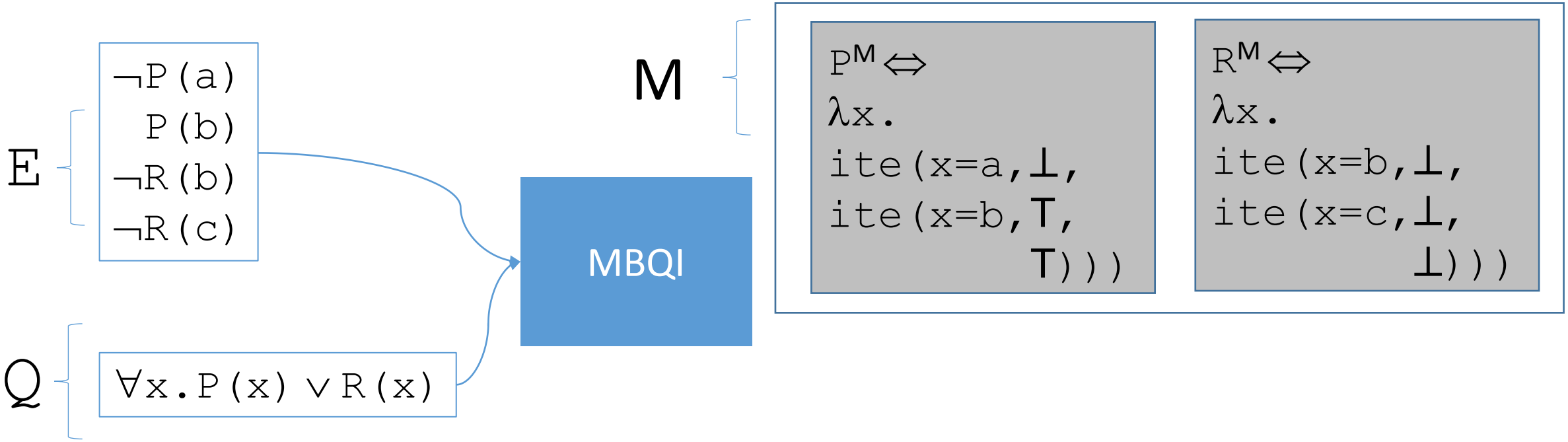


Build interpretation M of predicates

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

Model-based Instantiation

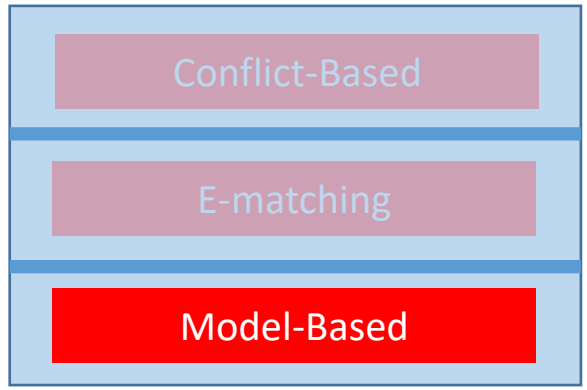


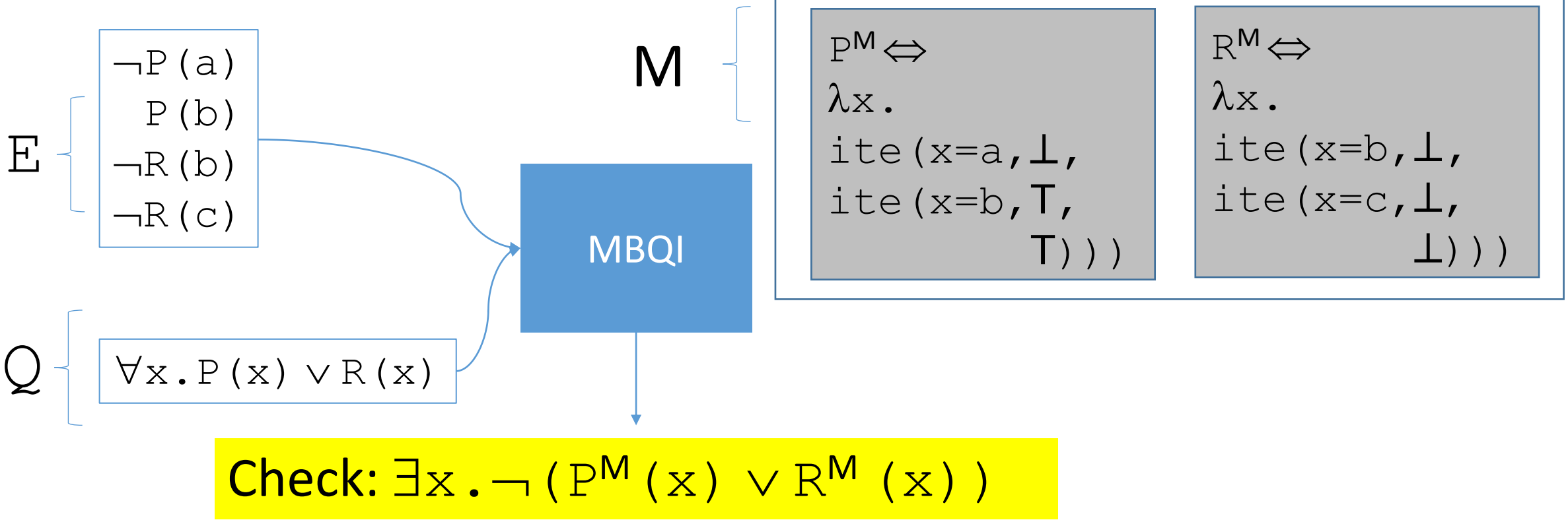


⇒ Does M satisfy Q?

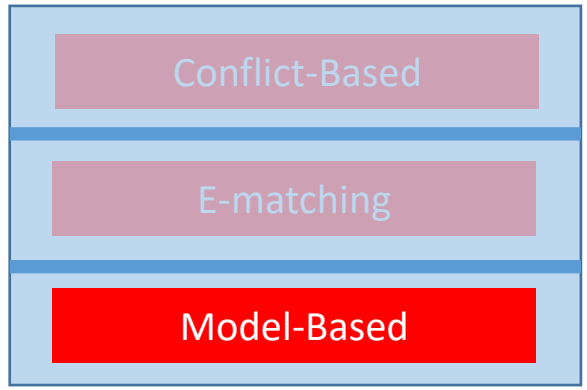
- Check (un)satisfiability of: $\exists x . \neg (P^M(x) \vee R^M(x))$

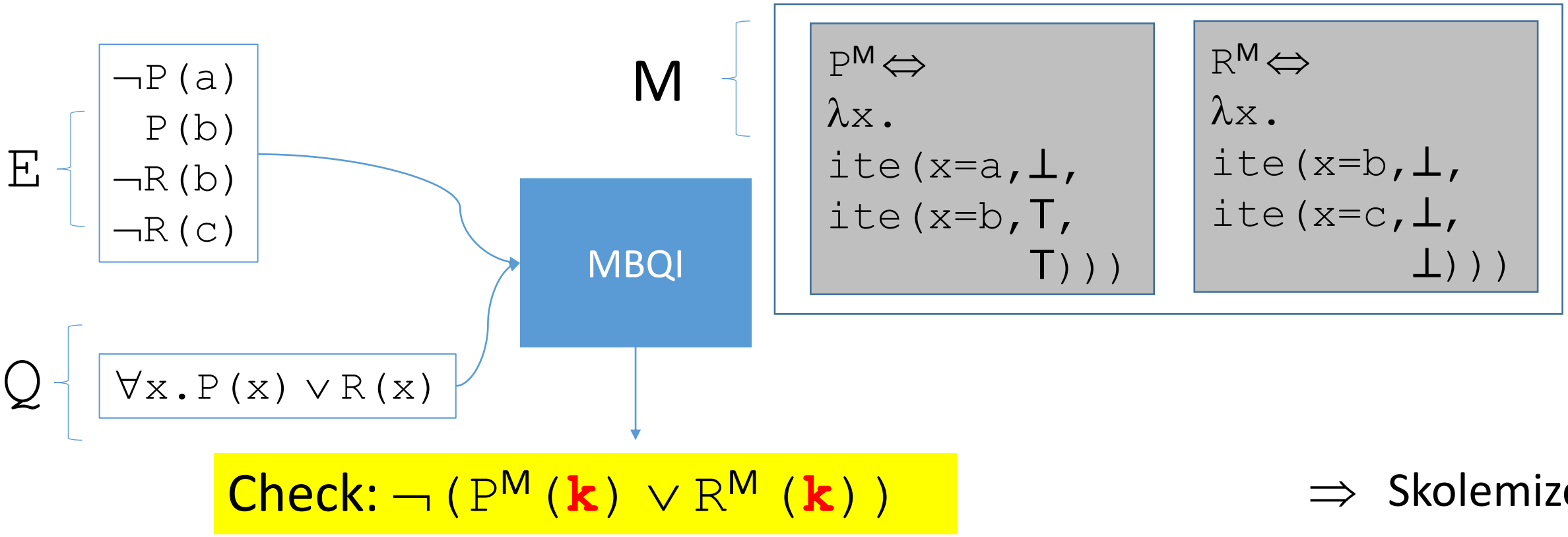
Model-based Instantiation



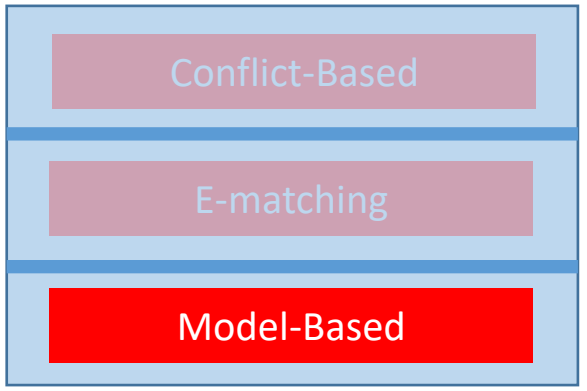


Model-based Instantiation





Model-based Instantiation



E

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

Q

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

MBQI

M

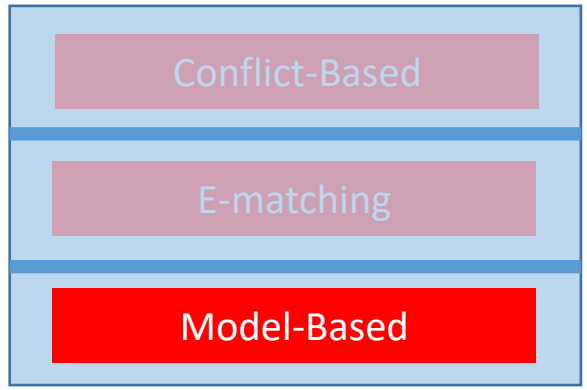
$P^M \Leftrightarrow$
$\lambda x.$
$ite(x=a,\bot,$
$ite(x=b,T,$
$,T)))$

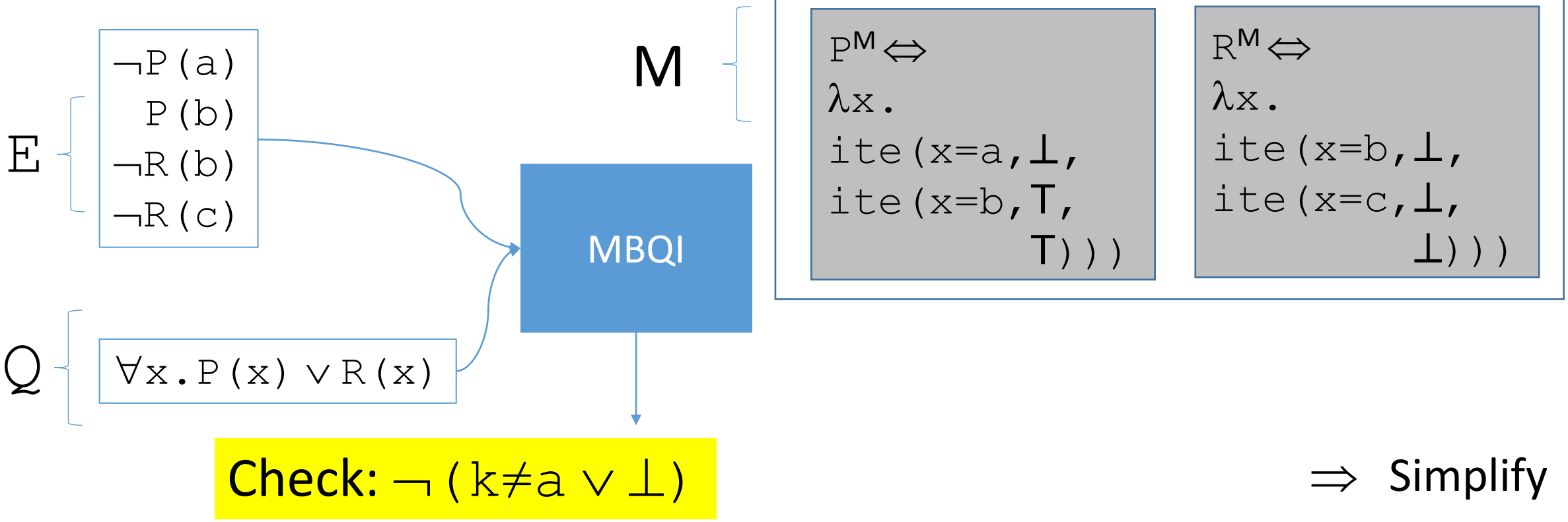
$R^M \Leftrightarrow$
$\lambda x.$
$ite(x=b,\bot,$
$ite(x=c,\bot,$
$,\bot)))$

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

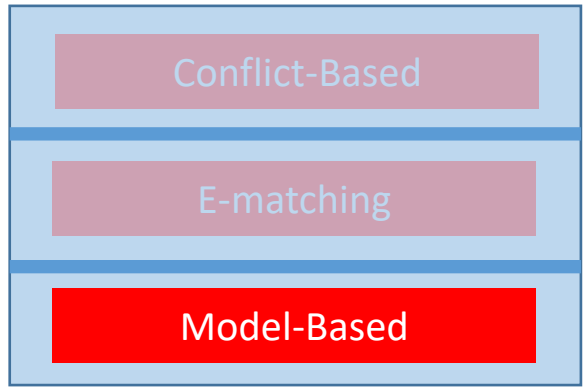
$\Rightarrow$ Substitute

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

M

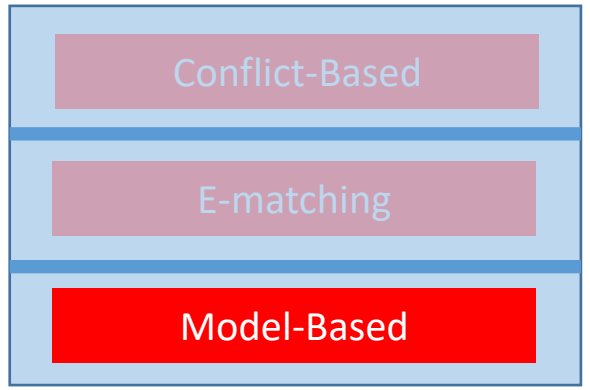
$P^M \Leftrightarrow \lambda x.\ ite(x=a, \bot, ite(x=b, \top, \top)))$

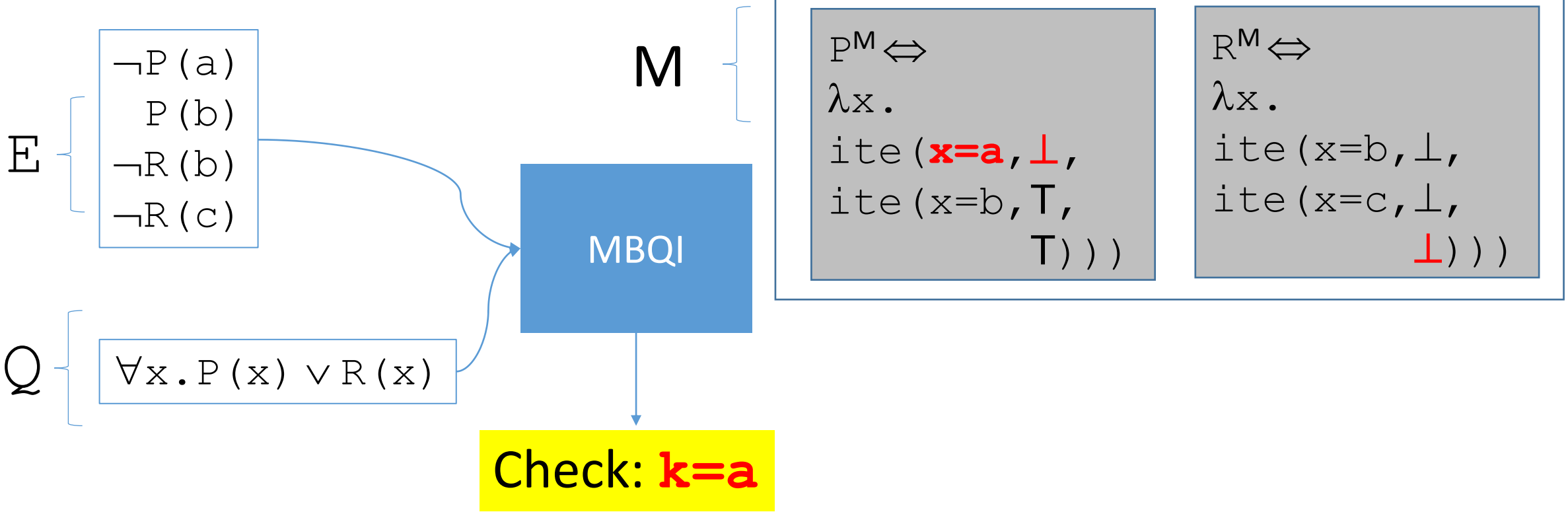
$R^M \Leftrightarrow \lambda x.\ ite(x=b, \bot, ite(x=c, \bot, \bot)))$

Check: $k=a$

$\Rightarrow$ Simplify

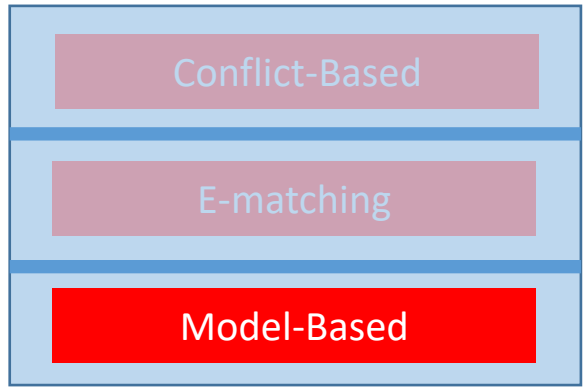
Model-based Instantiation

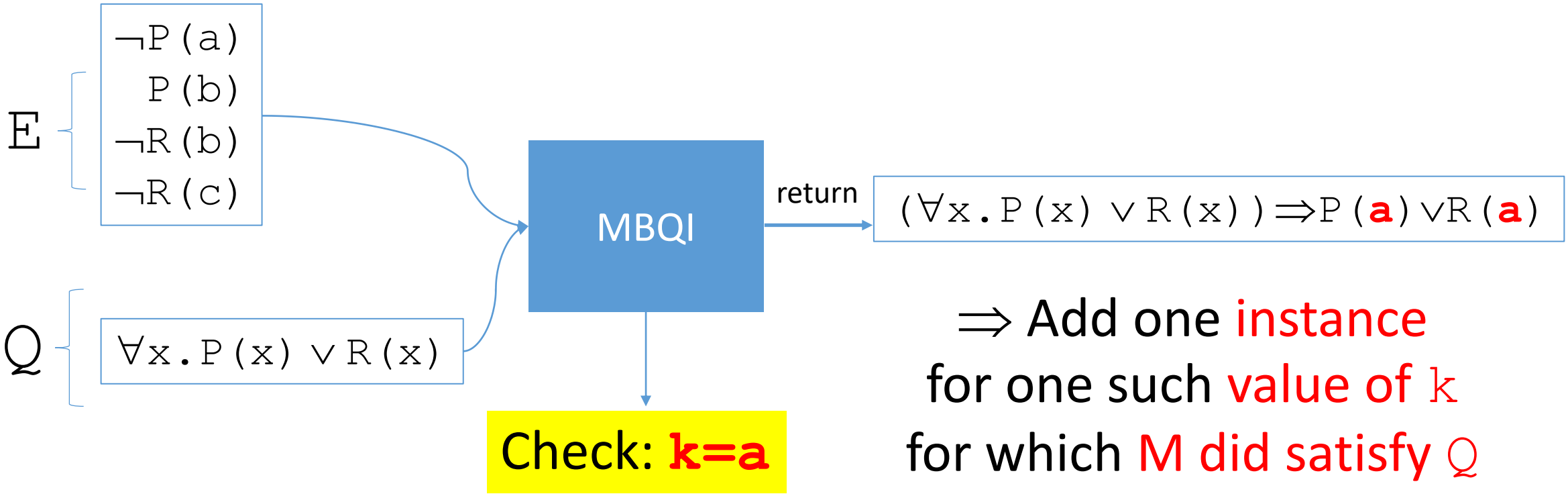




⇒ *Satisfiable! There are values k for which M does not satisfy Q*

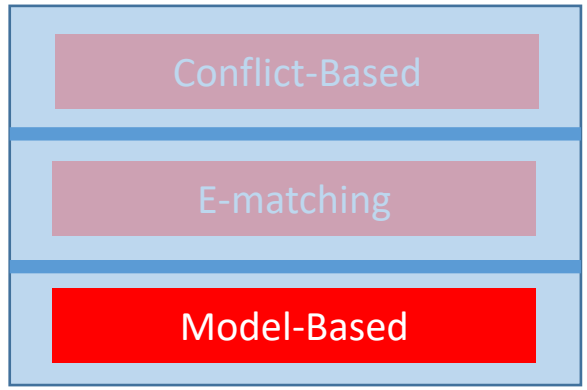
Model-based Instantiation

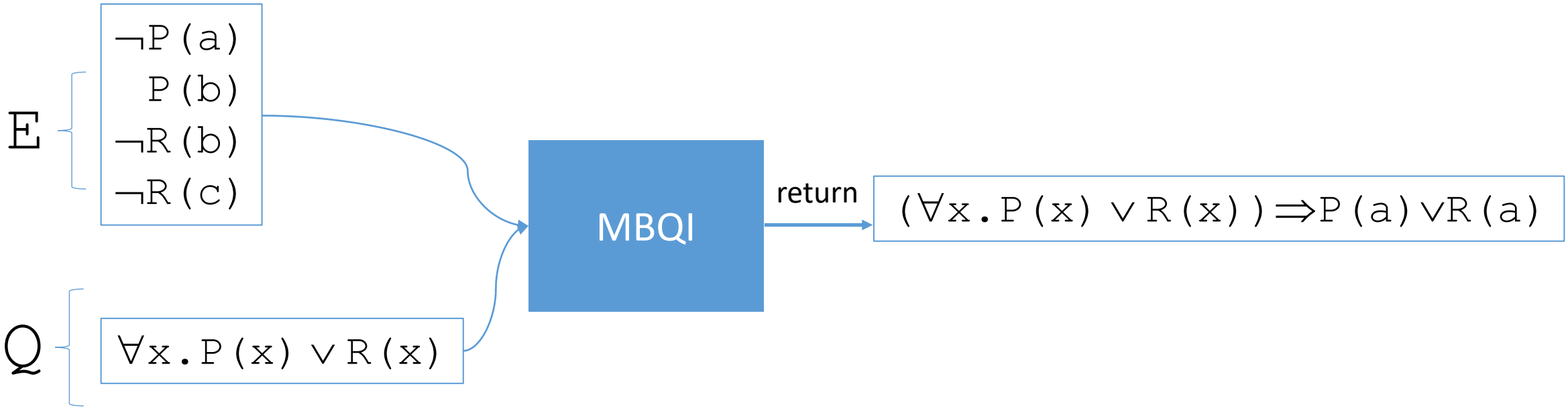




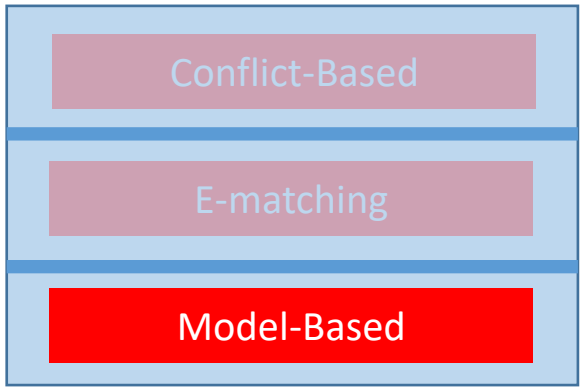
⇒ Add one instance
for one such value of k
for which M did satisfy Q

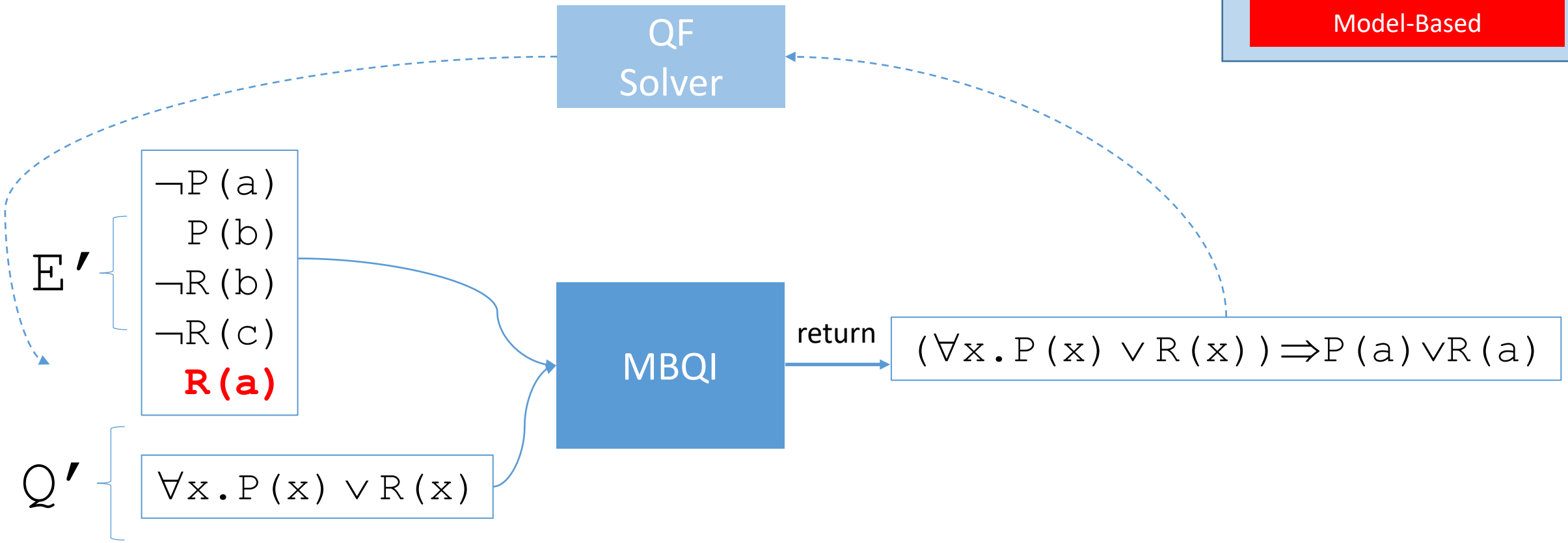
Model-based Instantiation





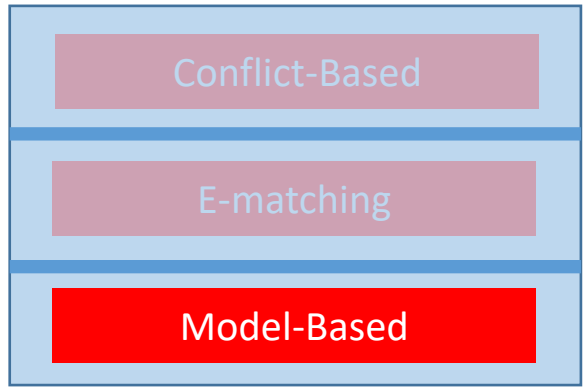
Model-based Instantiation

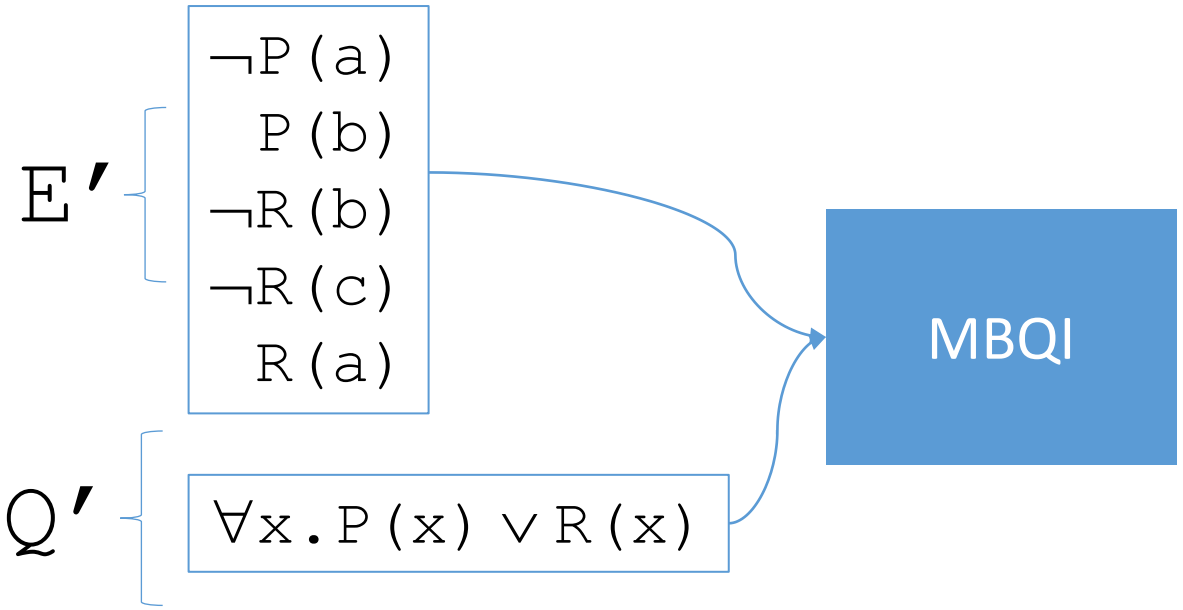




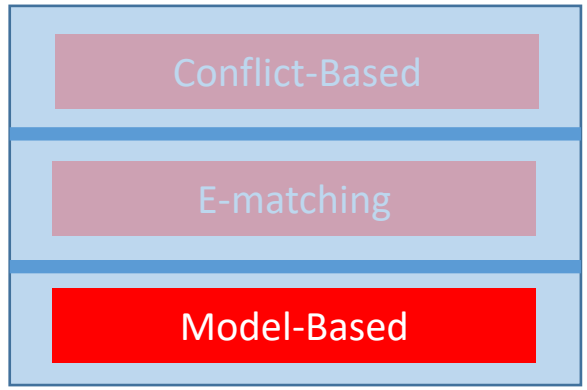
⇒ Subsequent models must satisfy $P(a) \vee \mathbf{R(a)}$

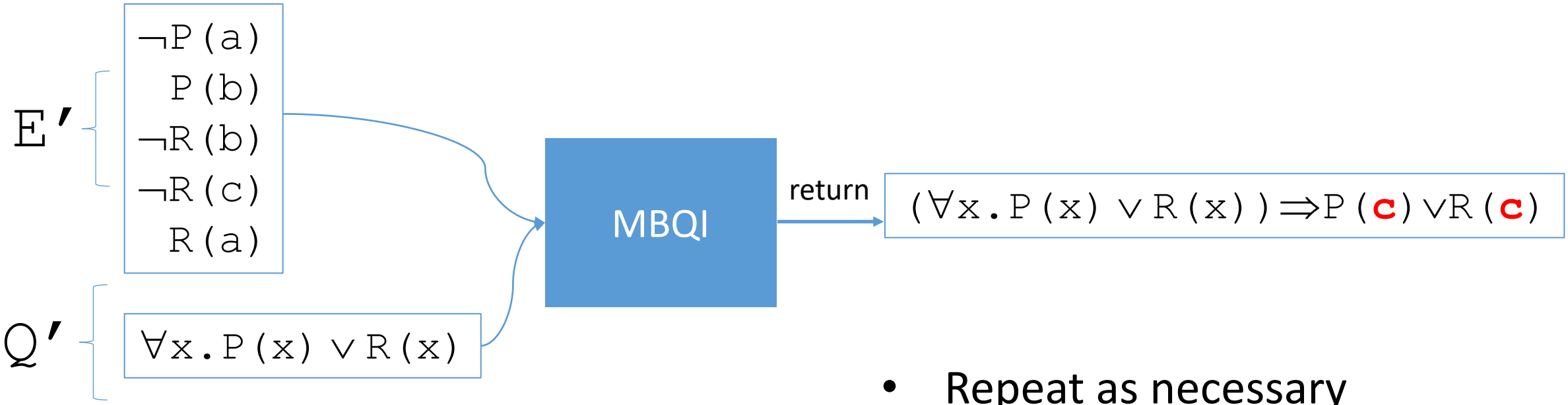
Model-based Instantiation





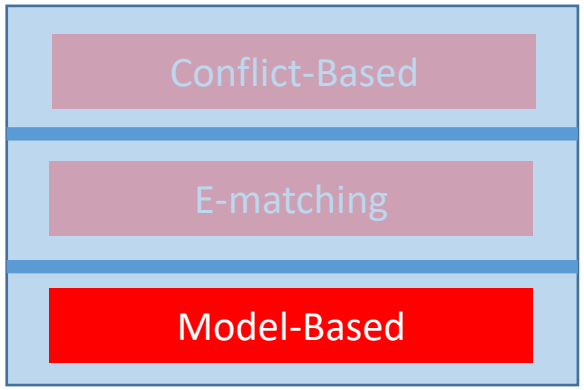
Model-based Instantiation





- Repeat as necessary
  ⇒ “Model refinement loop”

Model-based Instantiation



QF Solver

E''

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

Q''

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

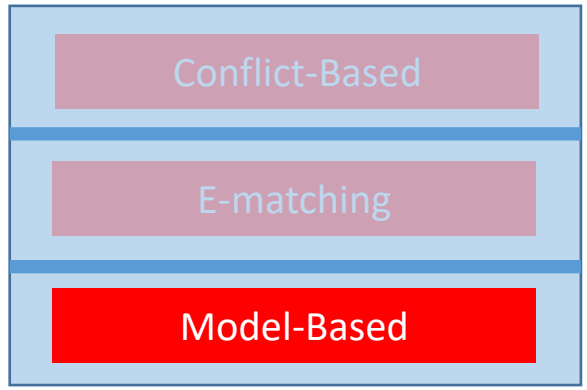
MBQI

return

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

- Repeat as necessary
  $\Rightarrow$ “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

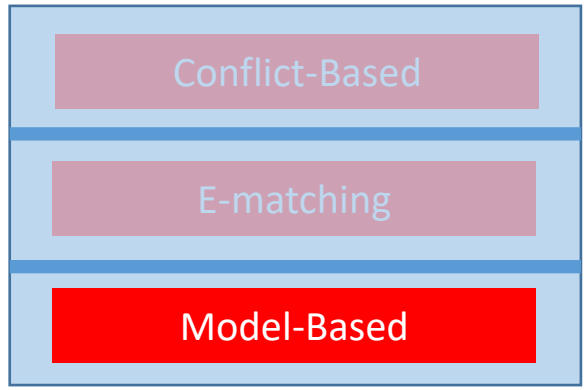
M''

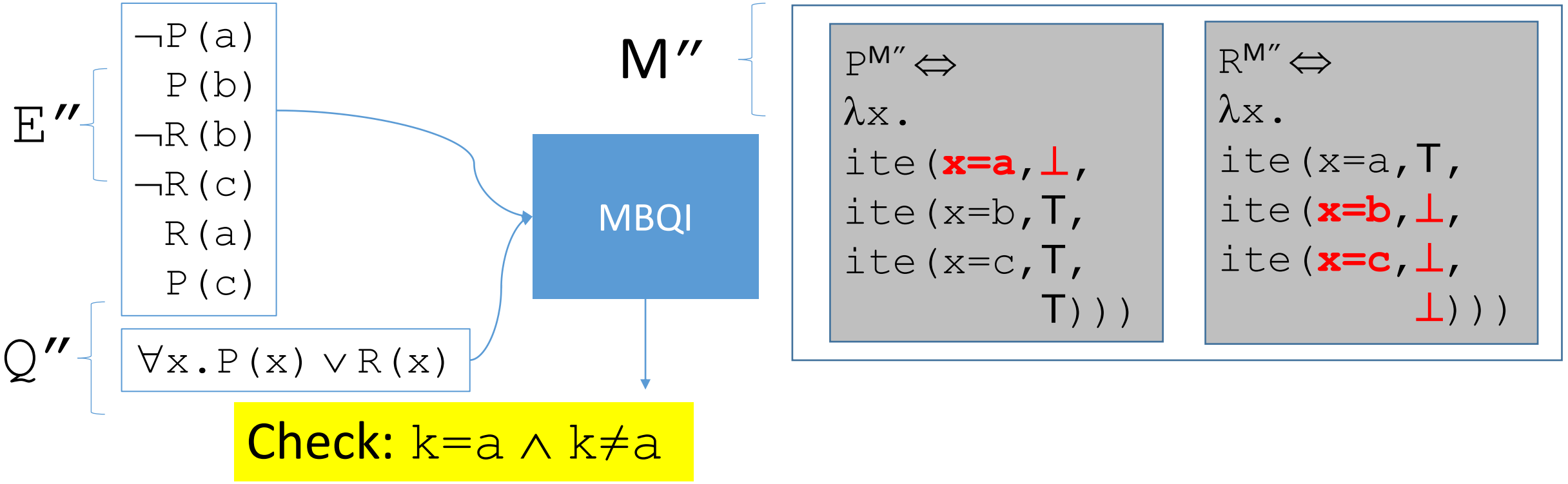
$P^{M''} \Leftrightarrow \lambda x.\ ite(x=a, \bot, ite(x=b, \top, ite(x=c, \top, \top)))$

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

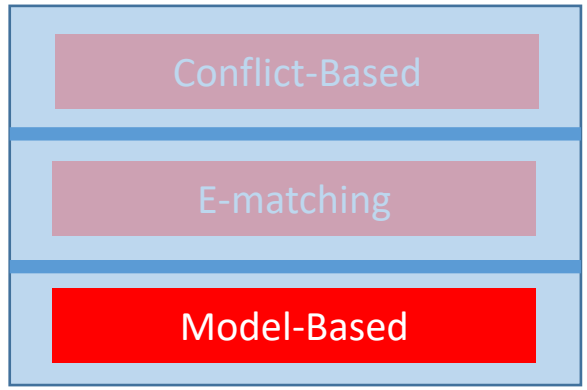
Check: $\exists x.\neg(P^{M''}(x) \vee R^{M''}(x))$

Model-based Instantiation





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

M''

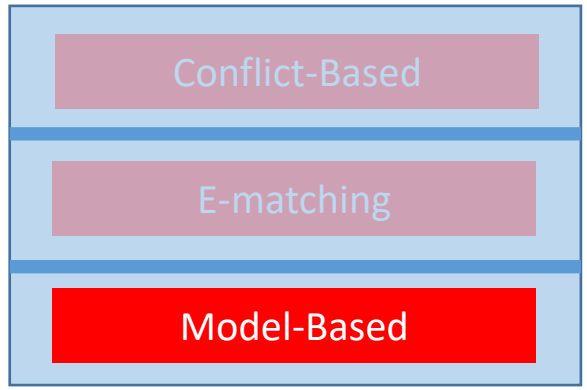
$P^{M''} \Leftrightarrow \lambda x.\ ite(x=a, \bot, ite(x=b, \top, ite(x=c, \top, \top)))$

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

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

$\Rightarrow$ Unsatisfiable, there are no values k for which 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

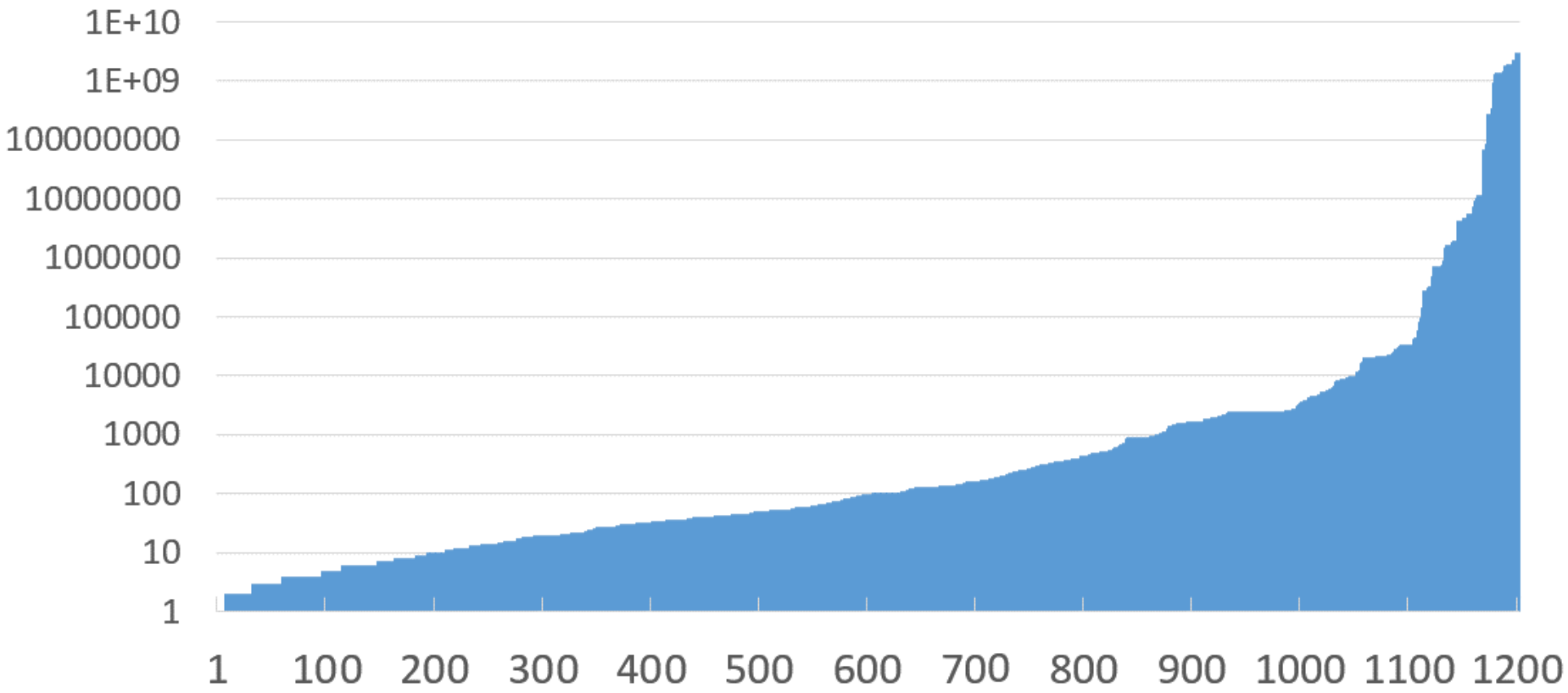
M''

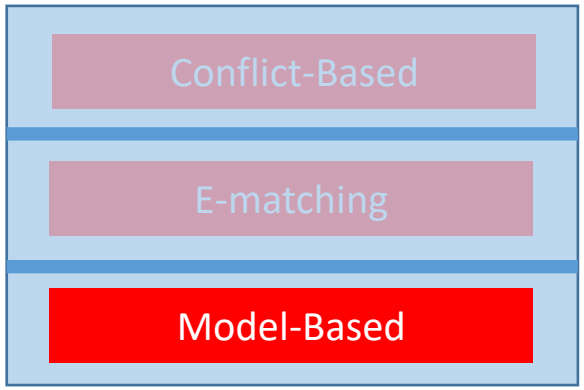
$P^{M''} \Leftrightarrow \lambda x.\ ite(x=a, \bot, ite(x=b, \top, ite(x=c, \top, \top)))$

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

sat, model M''

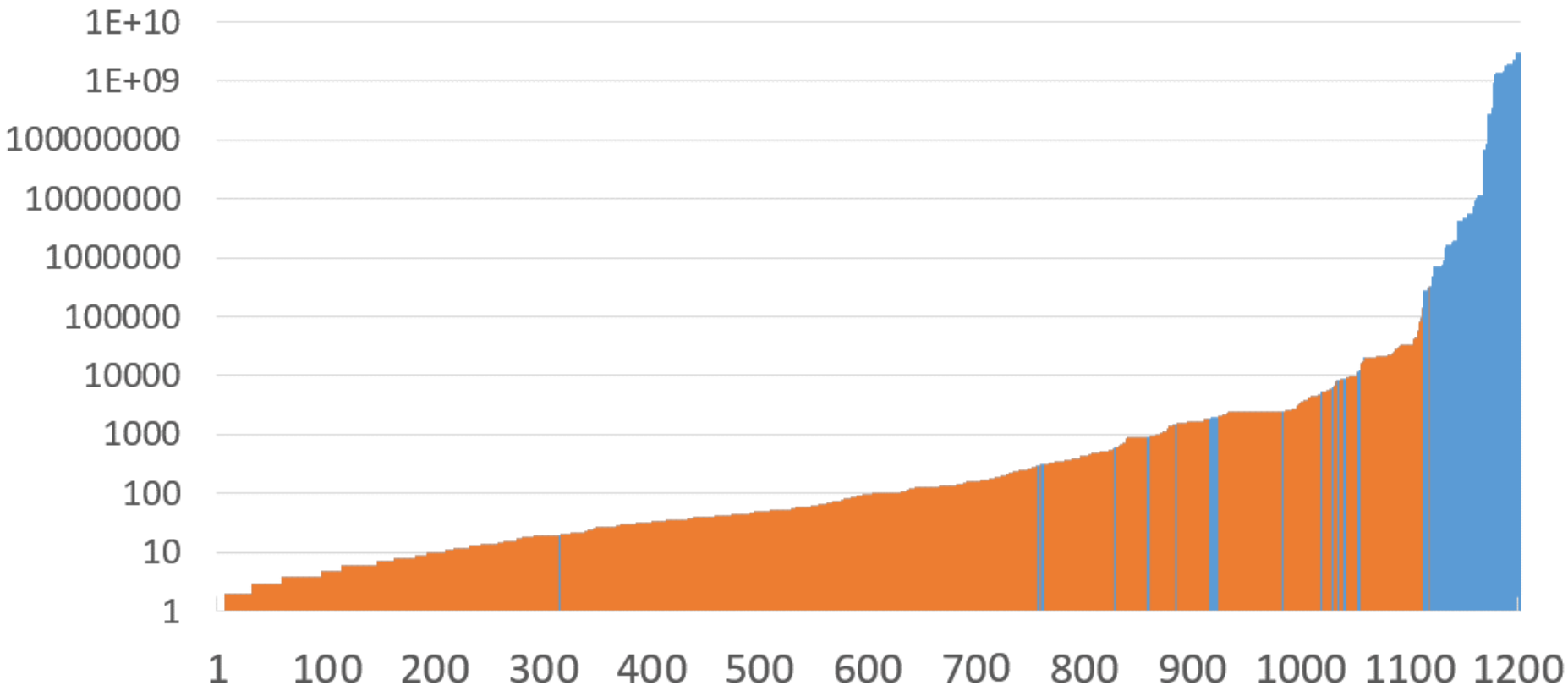
Model-based Instantiation: Impact

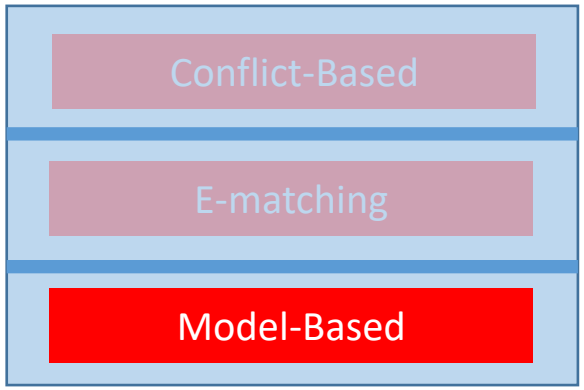




- 1203 satisfiable benchmarks from the TPTP library
  - Graph shows # instances required by exhaustive instantiation
    - E.g. $\forall xyz:U.P(x,y,z)$, if $|U|=4$, requires $4^3=64$ instances

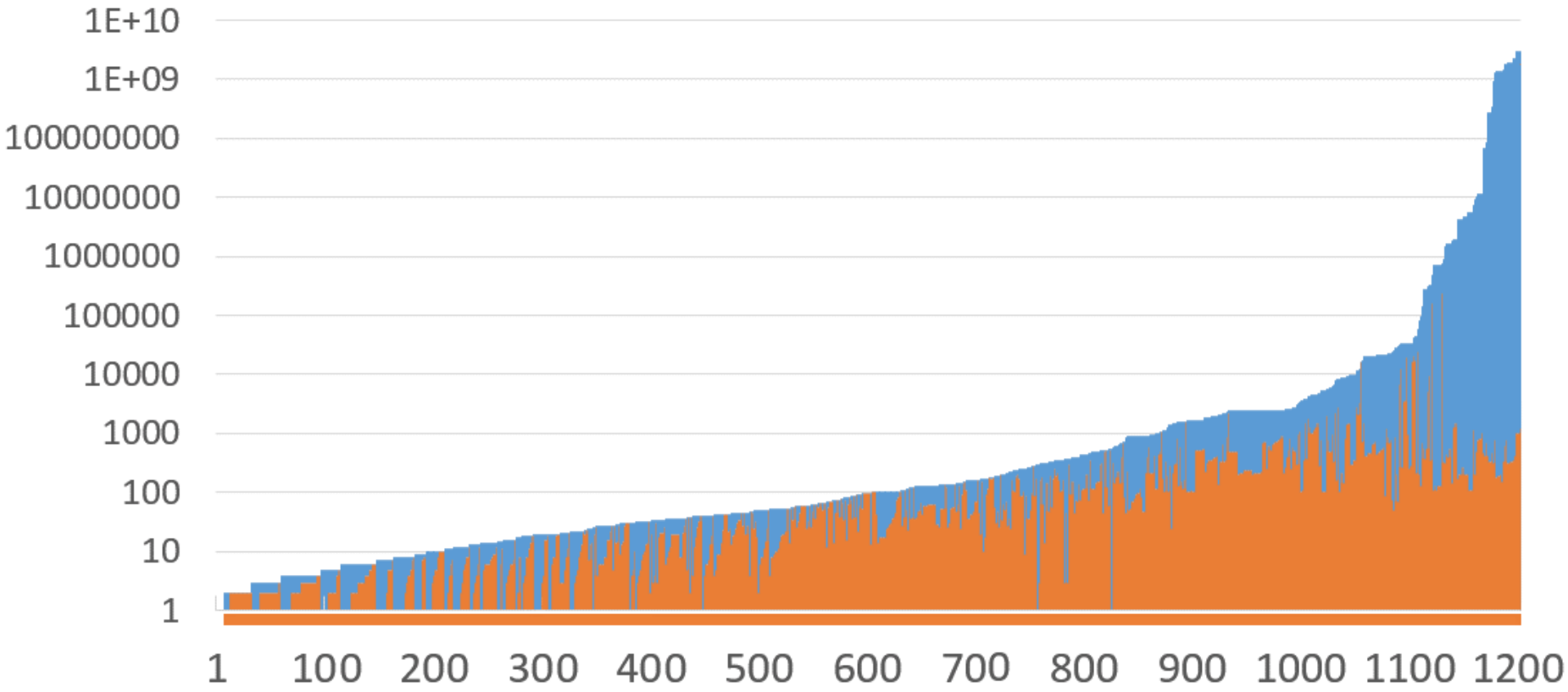
Model-based Instantiation: Impact

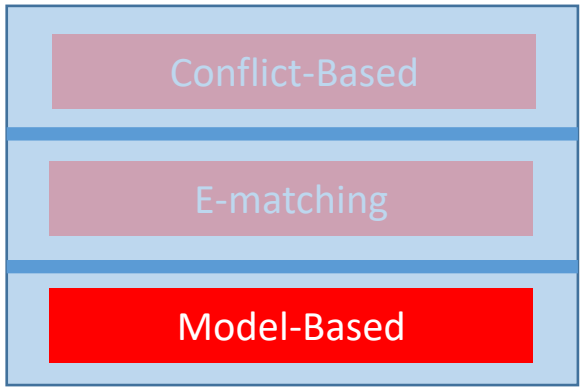




- CVC4 Finite Model Finding + Exhaustive instantiation
  - Scales only up to ~150k instances with a 30 sec timeout

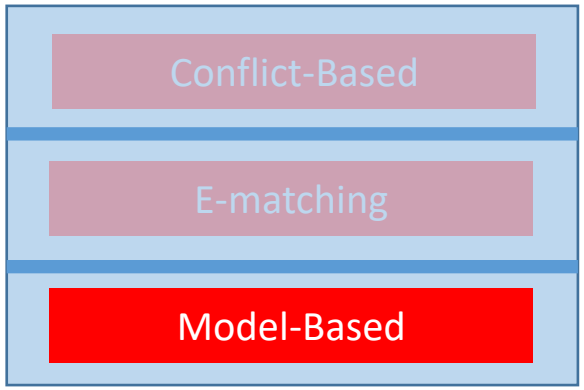
Model-based Instantiation: Impact





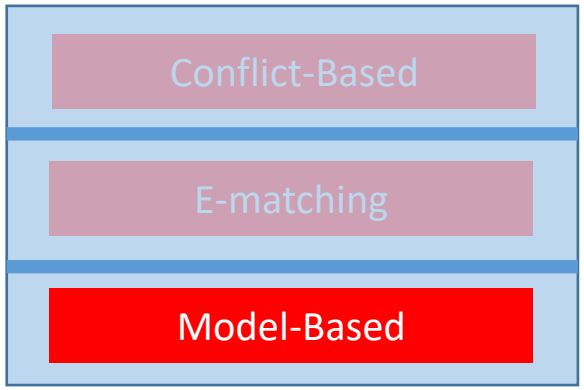
- CVC4 Finite Model Finding + Model-Based instantiation **[Reynolds et al CADE13]**
  - Scales to >2 billion instances with a 30 sec timeout, only adds fraction of possible instances

Model-based Instantiation: Challenges



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

Model-based Instantiation: Challenges



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

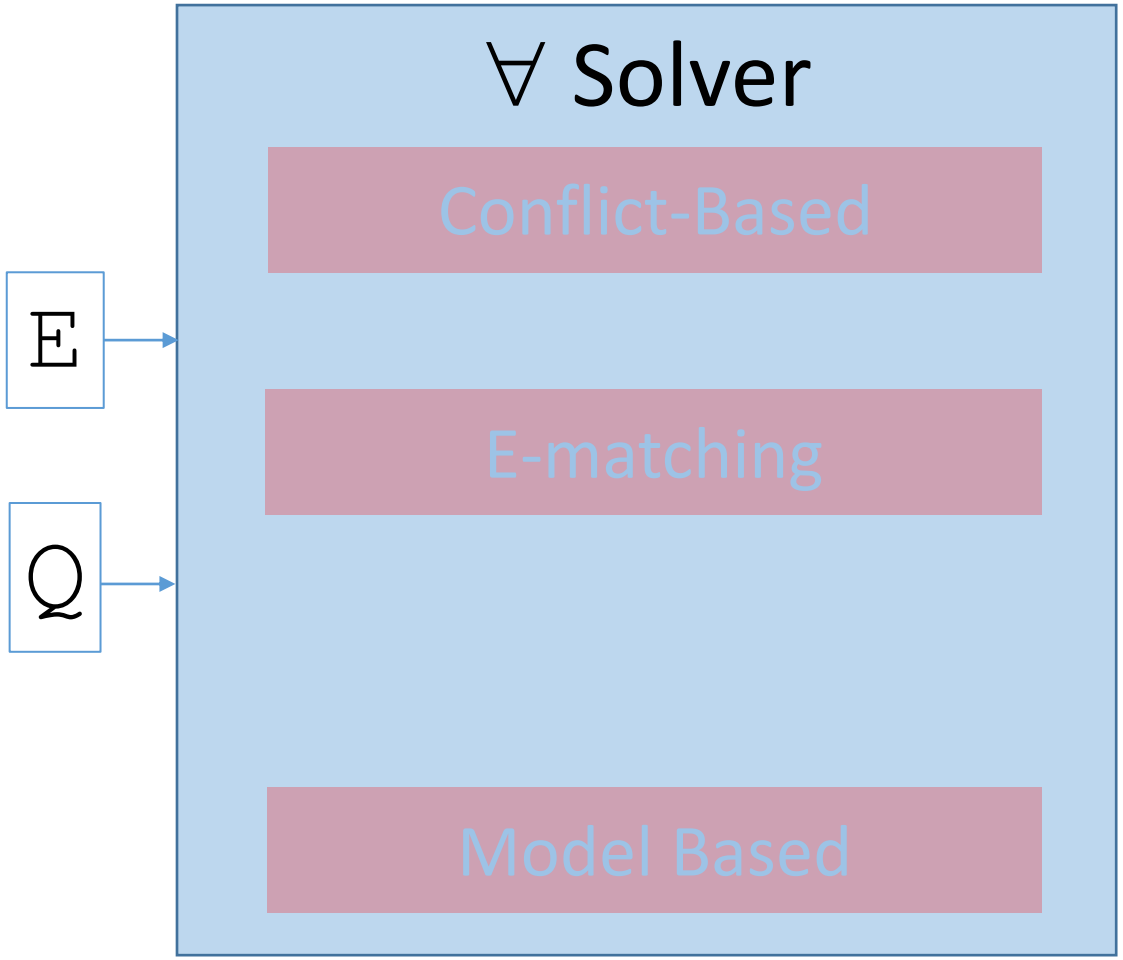
…but models may need to be more complex when *theories are present*:

$\forall xy:Int.(f(x,y) \geq x \wedge f(x,y) \geq y)$ ⟶ $f^M := \lambda xy.ite(x \geq y, x, y)$

$\forall x:Int.3*g(x)+5*h(x)=x$ ⟶ $g^M := \lambda x.-3*x$, $h^M := \lambda x.2*x$

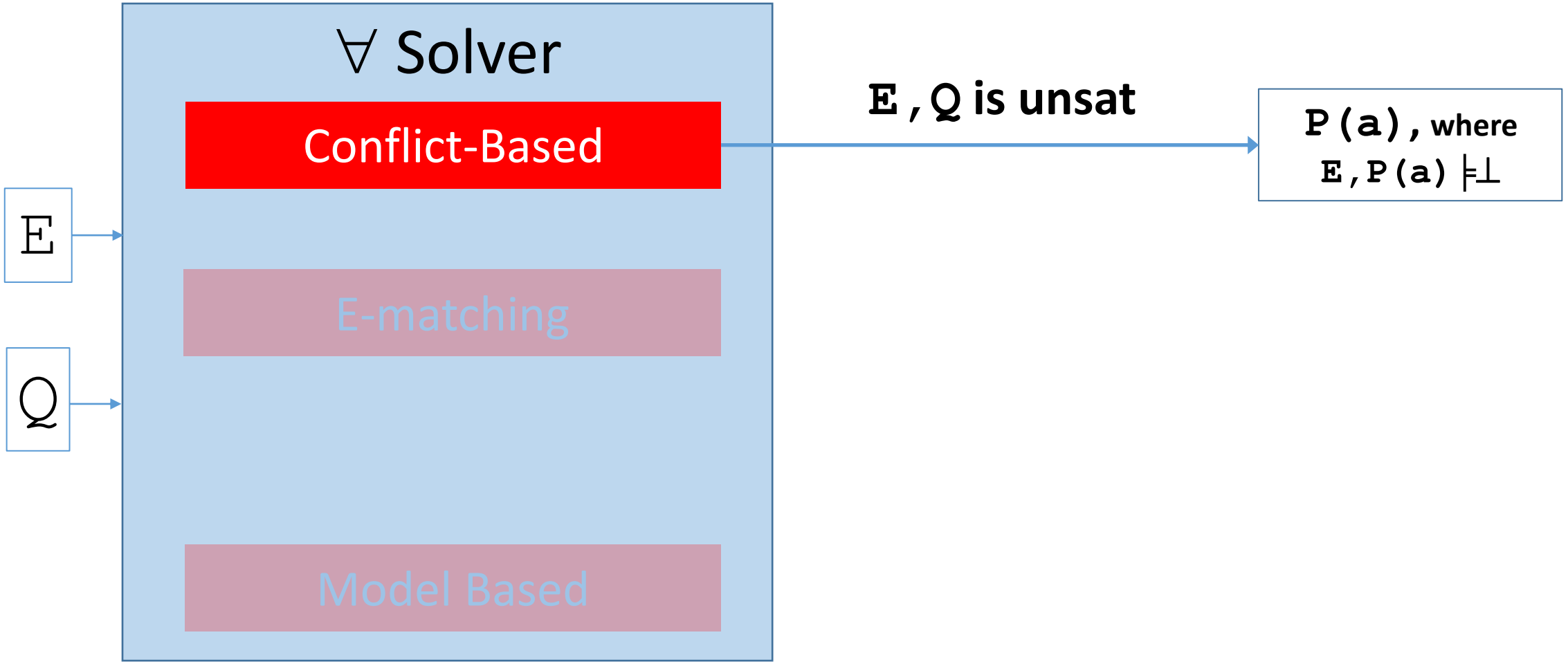
$\forall xy:Int.u(x+y)+11*v(w(x))=x+y$ ⟶ ???

Putting it Together

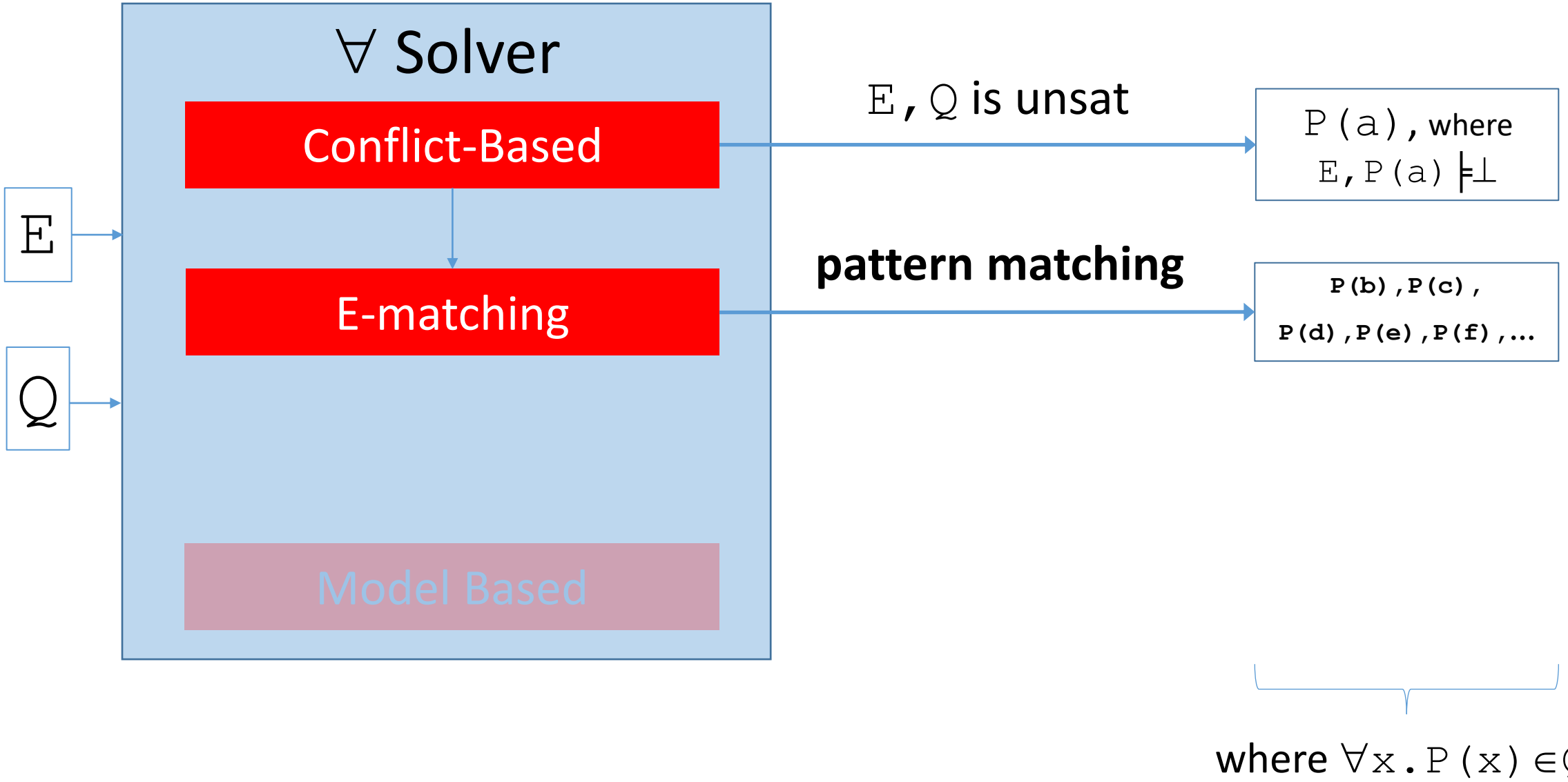


- Input:
  - Ground literals $\mathbb{E}$
  - Quantified formulas $\mathbb{Q}$

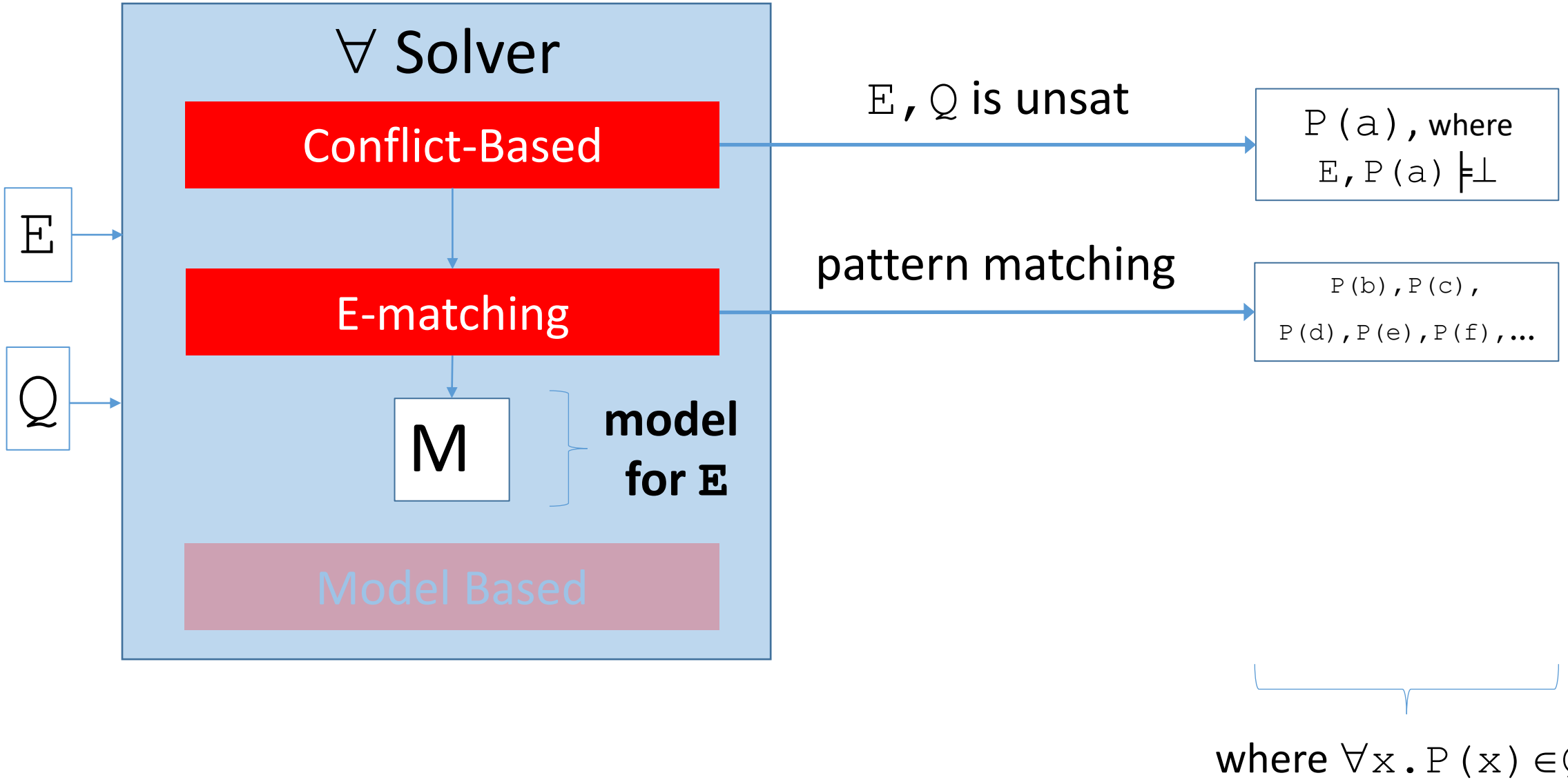
Putting it Together



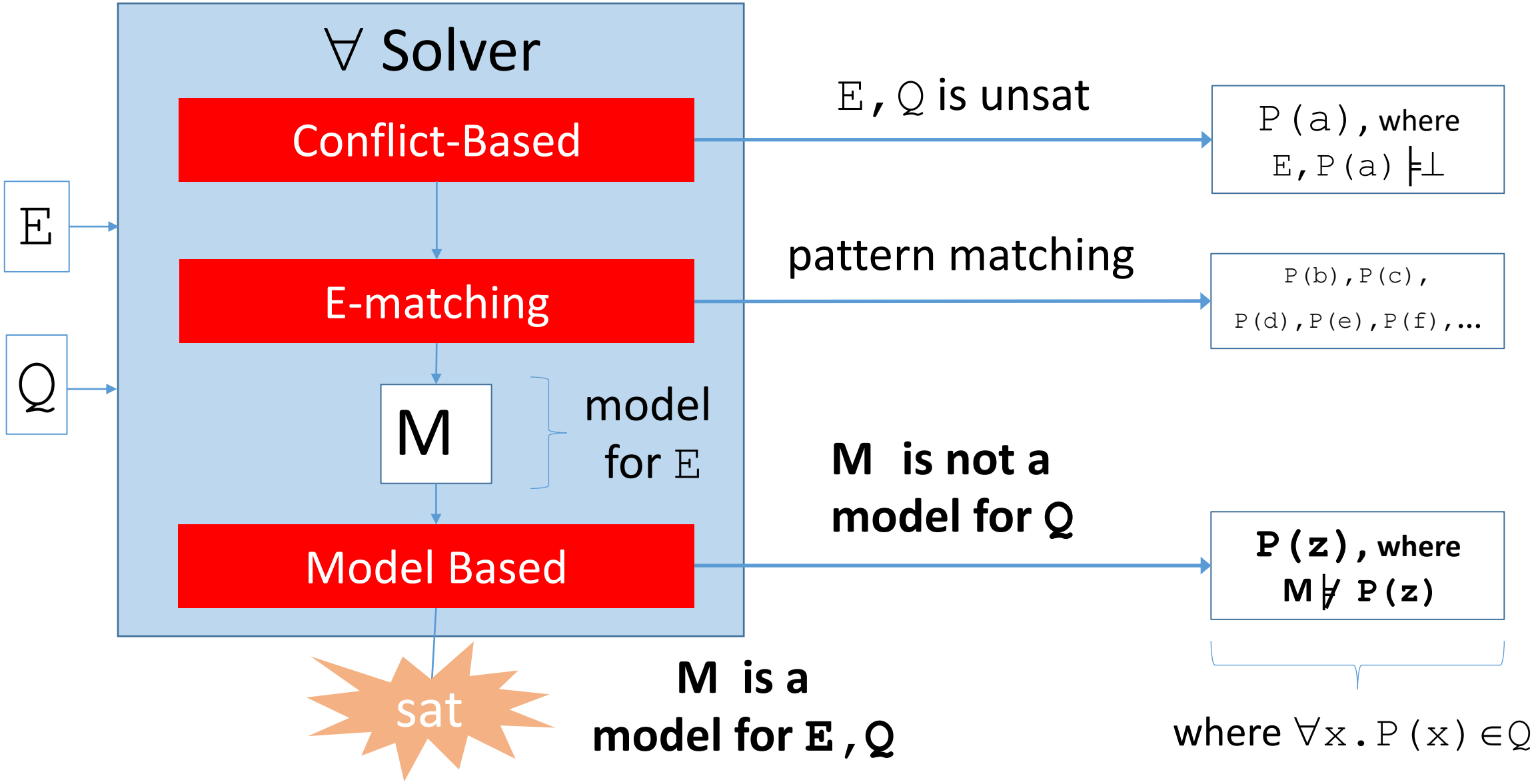
Putting it Together



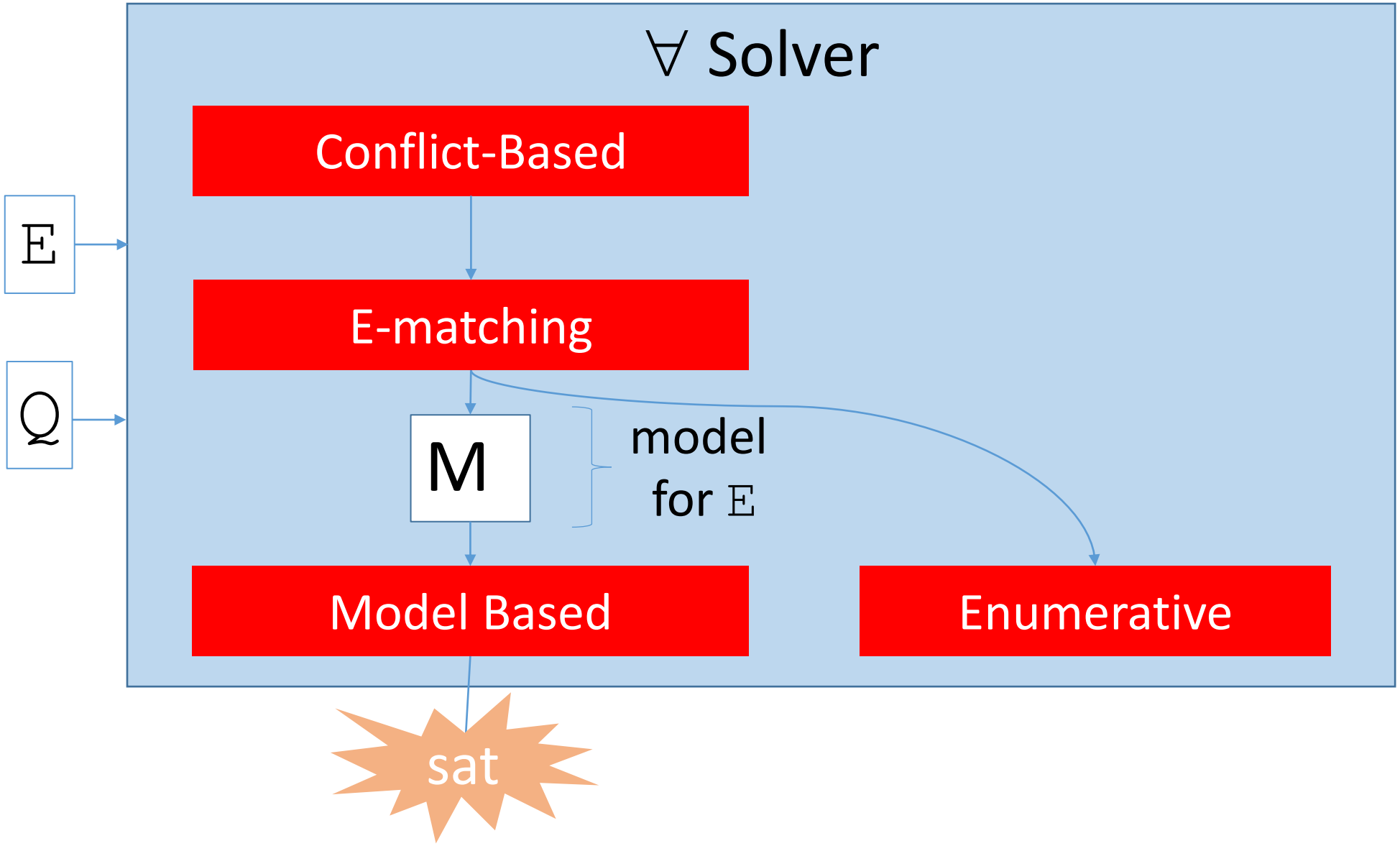
Putting it Together



Putting it Together



Putting it Together

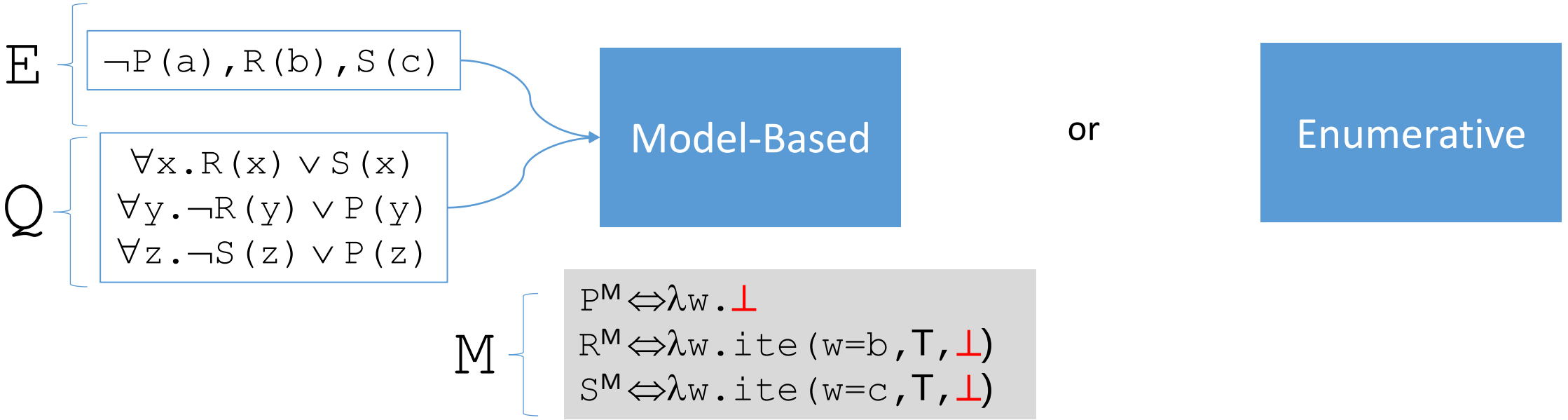


Enumerative Instantiation

- Alternative to model-based quantifier instantiation
- **Idea:** when E-matching saturates, generate further instances based on an arbitrary term ordering:
  - `(x,y)` $\rightarrow$ `(a,a)<(a,b)<(b,a)<(b,b)<(a,c)<(b,c)<` …
- Has advantages wrt model-based instantiation:
  - Better performance for unsat
    - Introduce fewer terms $\Leftrightarrow$ more likely to generate conflicts
  - Simpler to implement

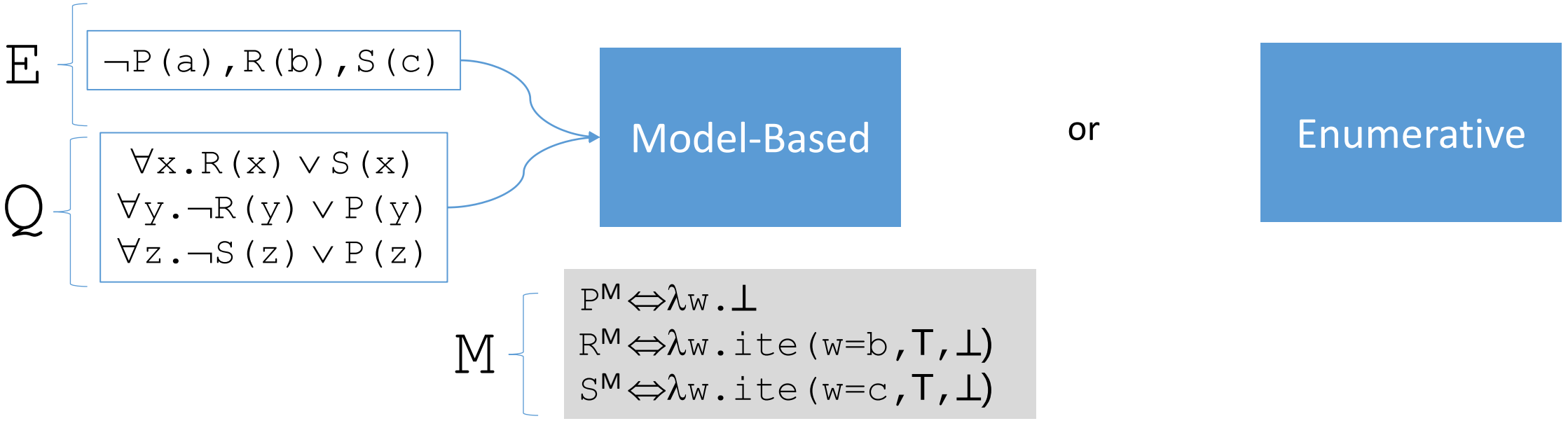
Enumerative vs Model-Based

Enumerative vs Model-Based



- …choose arbitrary default values

Enumerative vs Model-Based



The set of possible instantiations of Q for this round

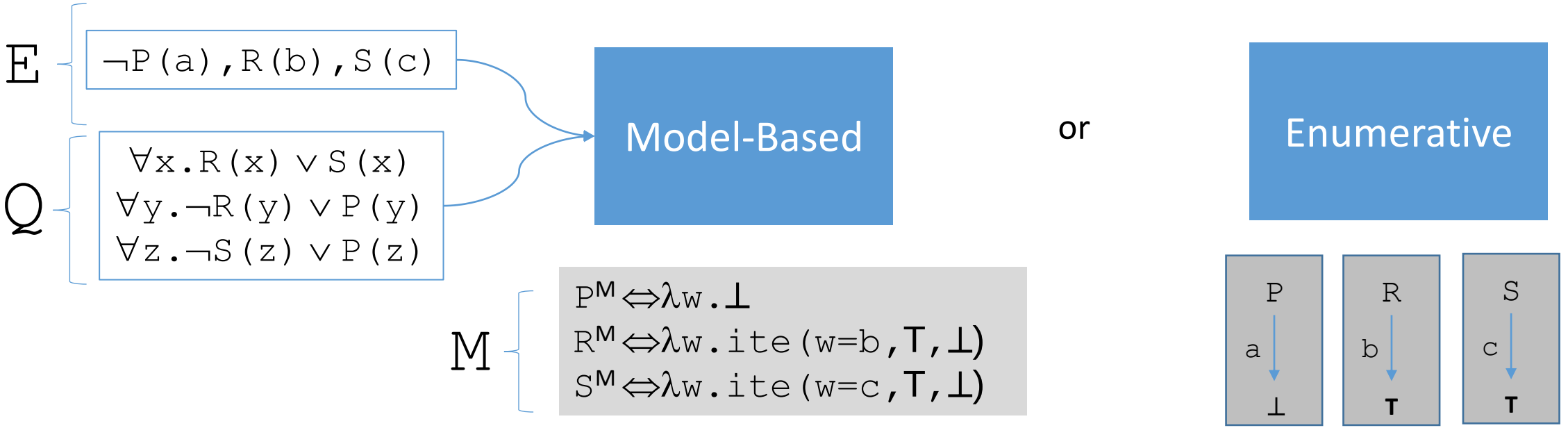
$x \rightarrow \{a\}$

$y \rightarrow \{b\}$

$z \rightarrow \{c\}$

… those not satisfied in model M

Enumerative vs Model-Based



The set of possible instantiations of Q for this round

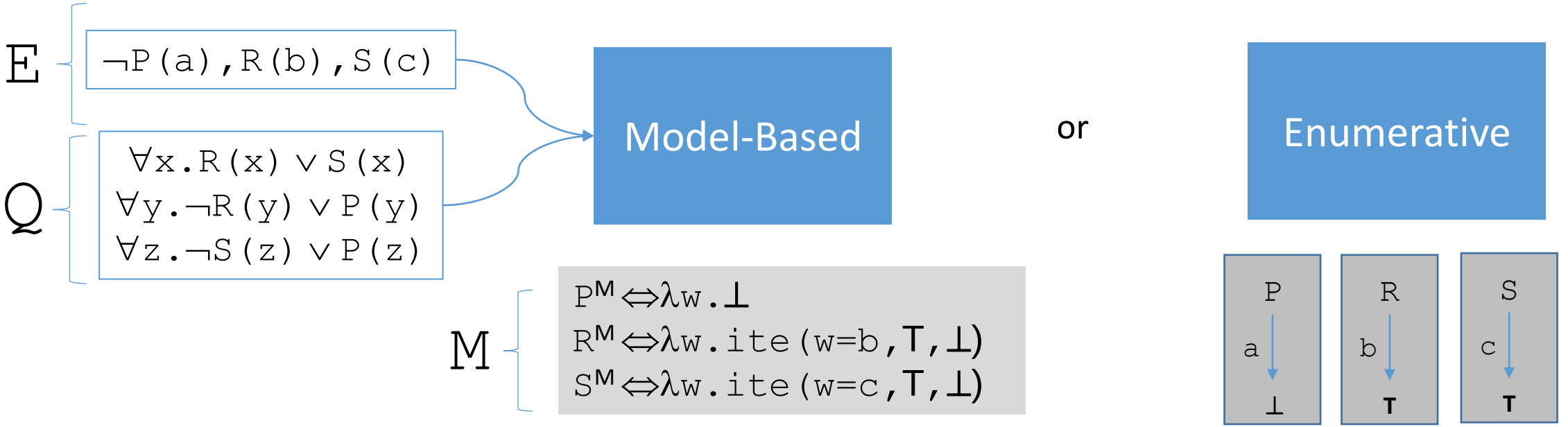
$x \rightarrow \{a\}$
$y \rightarrow \{b\}$
$z \rightarrow \{c\}$

… those not satisfied in model M

$x \rightarrow \{a\}$
$y \rightarrow \{a,b,c\}$
$z \rightarrow \{a,b,c\}$

… those not ***entailed*** by constraints E

Enumerative vs Model-Based



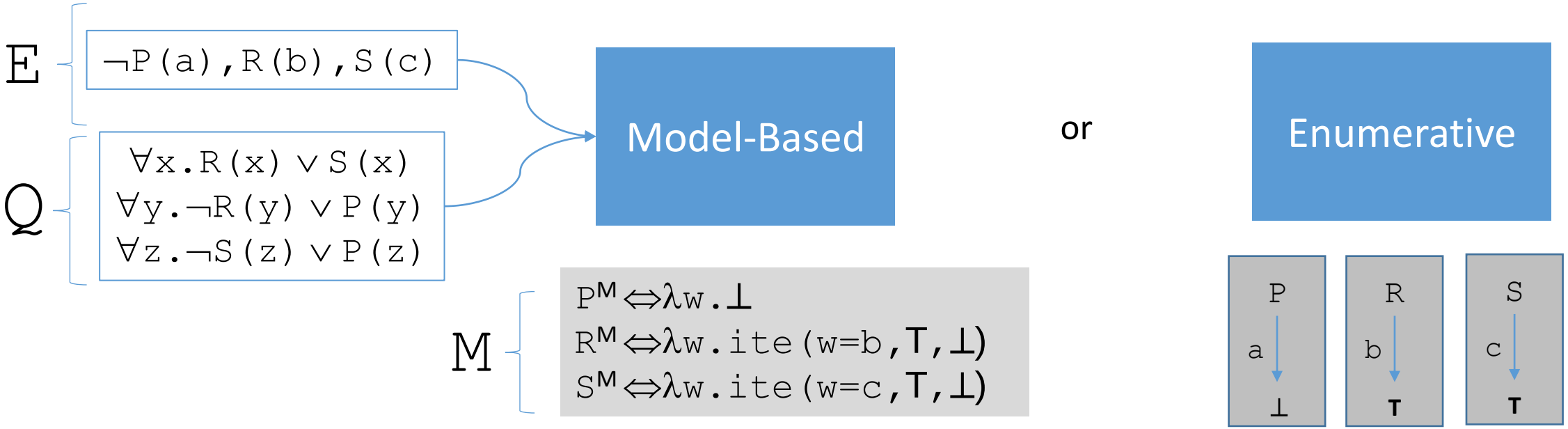
The set of possible instantiations of Q for this round

x → {**a**}	…	R(**a**) ∨ S(**a**)	x → {**a**}	… R(**a**) ∨ S(**a**)
y → {**b**}	…	¬R(**b**) ∨ P(**b**)	y → {**a**, b, c}	… ¬R(**a**) ∨ P(**a**)
z → {**c**}	…	¬S(**c**) ∨ P(**c**)	z → {**a**, b, c}	… ¬S(**a**) ∨ P(**a**)

… those not satisfied in model M

… those not *entailed* by constraints E

Enumerative vs Model-Based



$x \rightarrow \{a\} \ldots R(a) \vee S(a)$
$y \rightarrow \{b\} \ldots \neg R(b) \vee P(b)$
$z \rightarrow \{c\} \ldots \neg S(c) \vee P(c)$

$x \rightarrow \{a\} \ldots R(a) \vee S(a)$
$y \rightarrow \{a, \ldots\} \ldots R(a) \vee P(a)$
$z \rightarrow \{a, b, c\} \ldots \neg S(a) \vee P(a)$

unsat

Enumerative vs Model-Based

$\mathbf{m}(\mathsf{E}, \forall \bar{x}.\varphi)$: 1. Construct a model $\mathscr{M}$ for E.
2. Return $\{\{\bar{x} \mapsto \bar{t}\}\}$ where $\bar{t} \in \mathbf{T}(\mathsf{E})$ and $\mathscr{M} \not\models \varphi\{\bar{x} \mapsto \bar{t}\}$, or $\emptyset$ if none exists.

$\mathbf{u}(\mathsf{E}, \forall \bar{x}.\varphi)$: 1. Choose an ordering $\preceq$ on tuples of quantifier-free terms.
2. Return $\{\{\bar{x} \mapsto \bar{t}\}\}$ where $\bar{t}$ is a minimal tuple of terms w.r.t $\preceq$ such that $\bar{t} \in \mathbf{T}(\mathsf{E})$ and $\mathsf{E} \not\models \varphi\{\bar{x} \mapsto \bar{t}\}$, or $\emptyset$ if none exist.

[Reynolds et al, "Revisiting Enumerative Instantiation", TACAS 2018]

- Enumerative is better for unsat:
  - Model guesses may obscure useful instantiations
  - Term ordering simulates "breadth-first" search
- Model-based is better for sat:
  - Although enumerative instantiation may answer sat for some logics

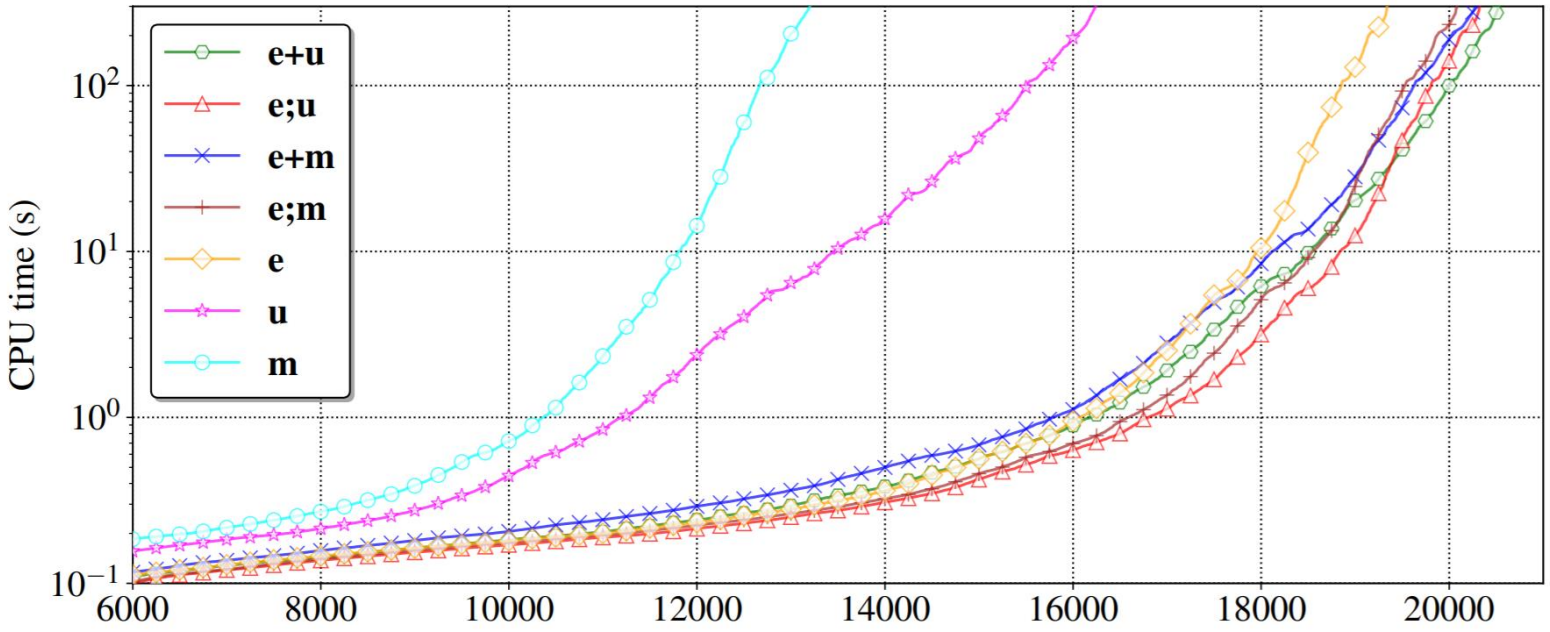
Results

[Reynolds et al, TACAS 2018]

- **e** = E-matching
- **m** = model-based
- **u** = enumerative

- **e;u** vs **e+u**
  - Priority vs interleaved
- Unsat benchmarks
  - SMT-LIB, TPTP

(Note: Conflict-based used as first priority in each configuration)



Library	#	u	e;u	e+u	e	m	e;m	e+m	uport	mport	port
TPTP	14731	4426	6125	6273	5396	4369	6066	6151	6674	6566	6859
UF	7293	2607	2906	2961	2862	2418	2898	2972	3119	3045	3159
UFDT	4384	1783	1977	1998	1958	1642	1954	1993	2091	2070	2113
UFLIA	7745	3622	5022	5037	4867	2638	4966	4989	5253	5132	5279
UFNIA	3213	1788	1947	1978	1937	1169	1860	1865	2107	2064	2138
Others	4699	2019	2348	2288	2320	966	2338	2312	2400	2363	2404
Total	42065	16245	20325	20535	19340	13202	20082	20282	21644	21240	21952

Quantifier Instantiation

- **Common thread:** satisfiability of ∀ + 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

Quantifier Instantiation

- **Common thread:** satisfiability of ∀ + 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

⇒ But reasoning about ∀ + theories without UF isn't as bad:

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

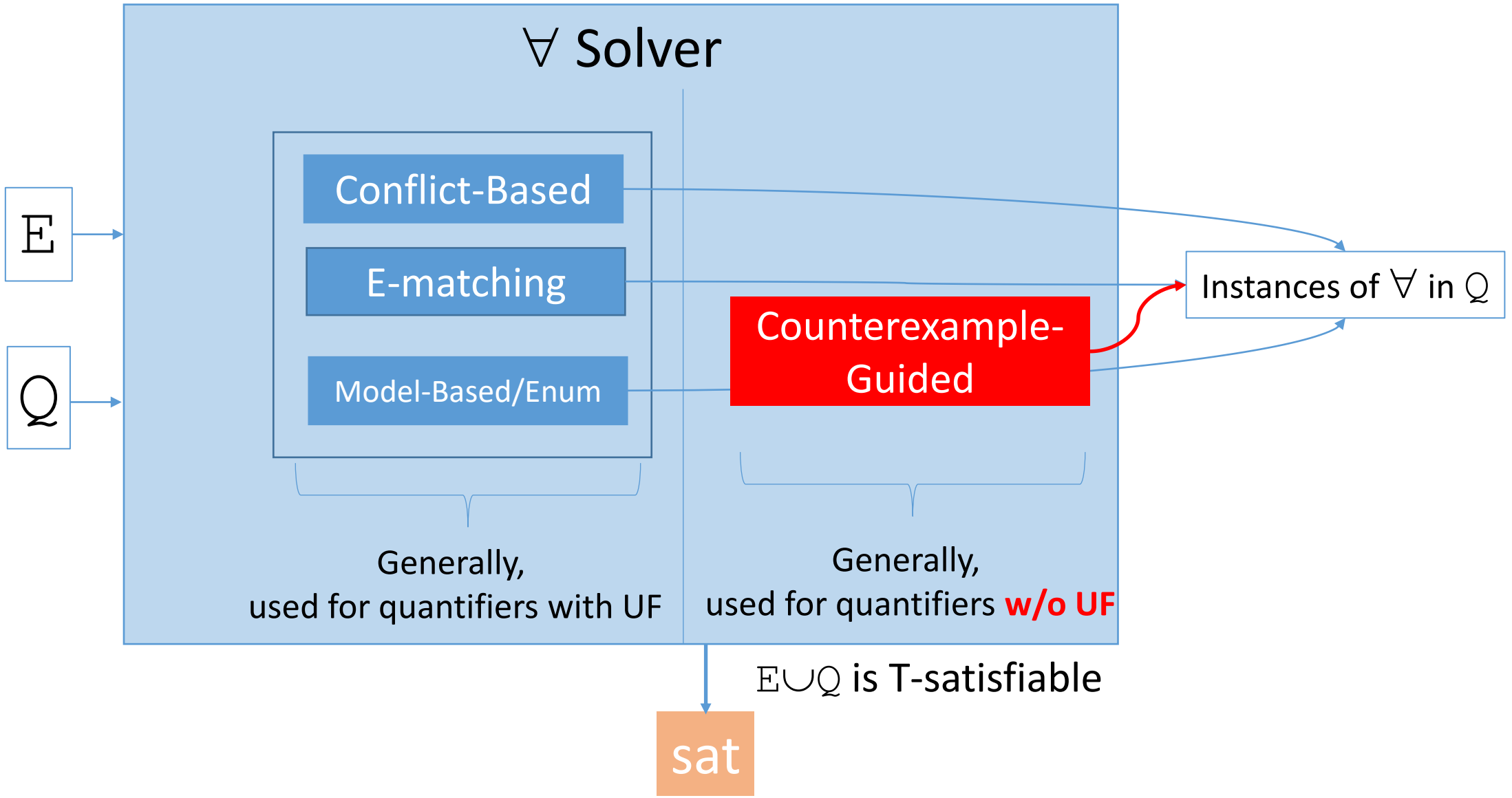
Quantifier Instantiation

- **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

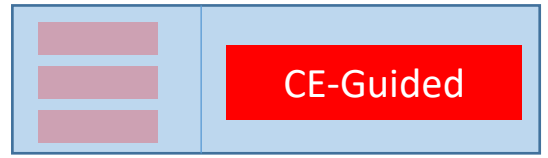
$\Rightarrow$ But reasoning about $\forall$ + theories without UF isn't as bad:

- 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

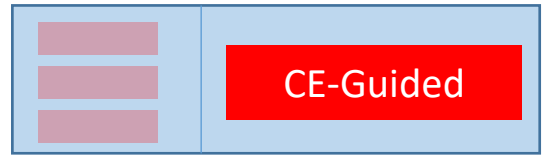


Counterexample-Guided Instantiation



- 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]

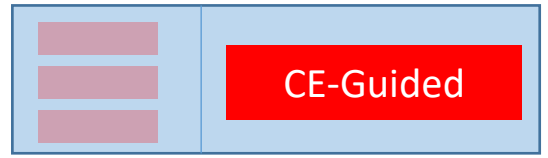
Counterexample-Guided Instantiation



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

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

Counterexample-Guided Instantiation

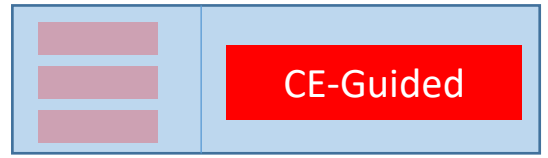


- High-level idea:
  - Quantifier elimination (e.g. for LIA) says:

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

(consider the dual)

Counterexample-Guided Instantiation

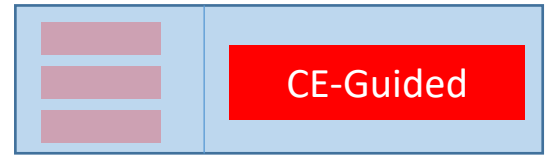


- High-level idea:
  - Quantifier elimination (e.g. for LIA) says:

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

  - 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

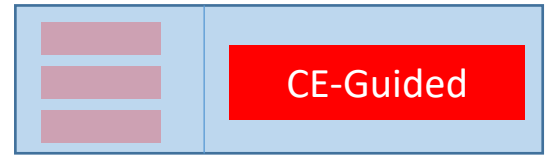


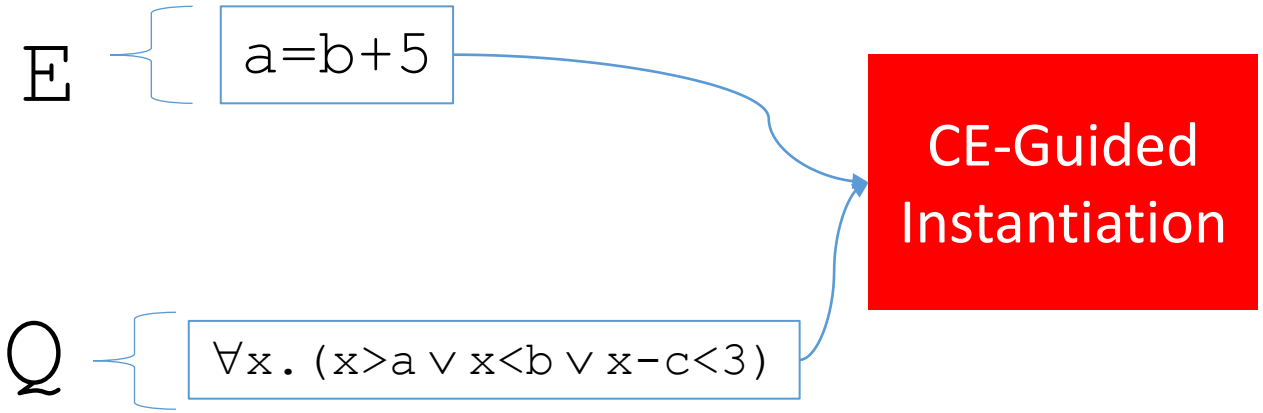
E { `a=b+5`

Q { `∀x.(x>a ∨ x<b ∨ x-c<3)`

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

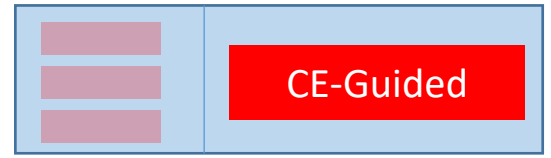
Counterexample-Guided Instantiation

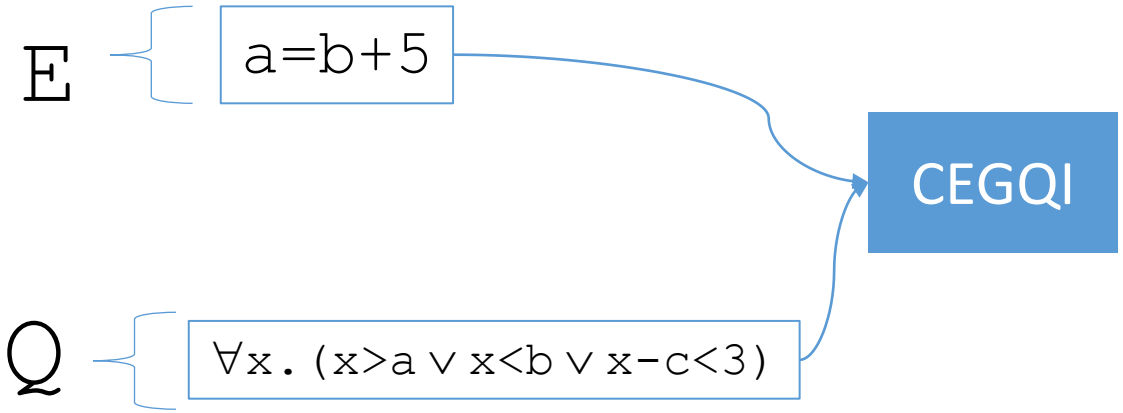




⇒Use *counterexample-guided instantiation*

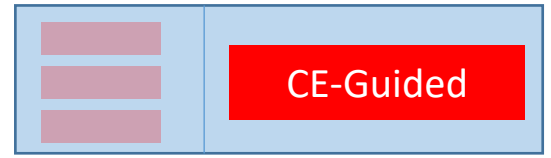
Counterexample-Guided Instantiation

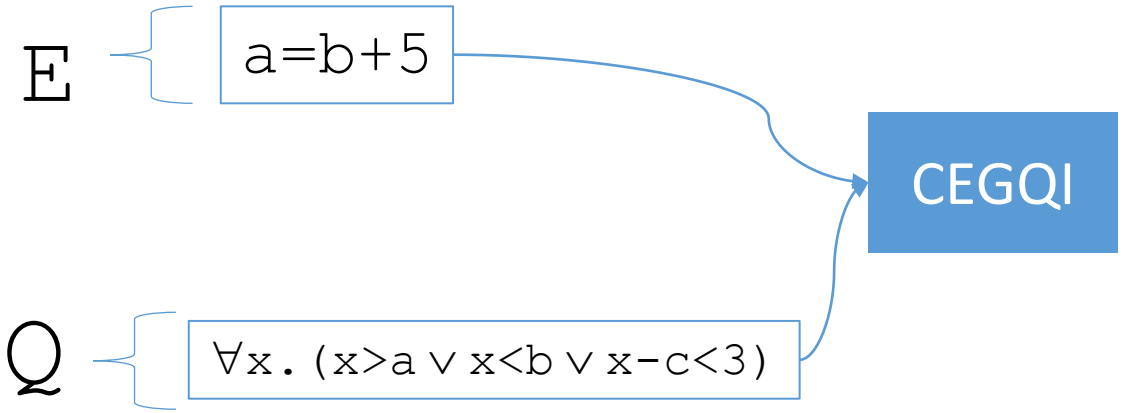




⇒Use *counterexample-guided instantiation*

Counterexample-Guided Instantiation

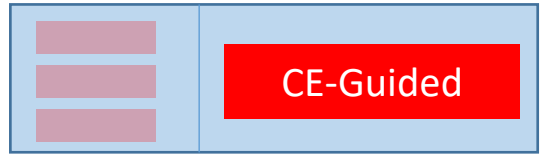


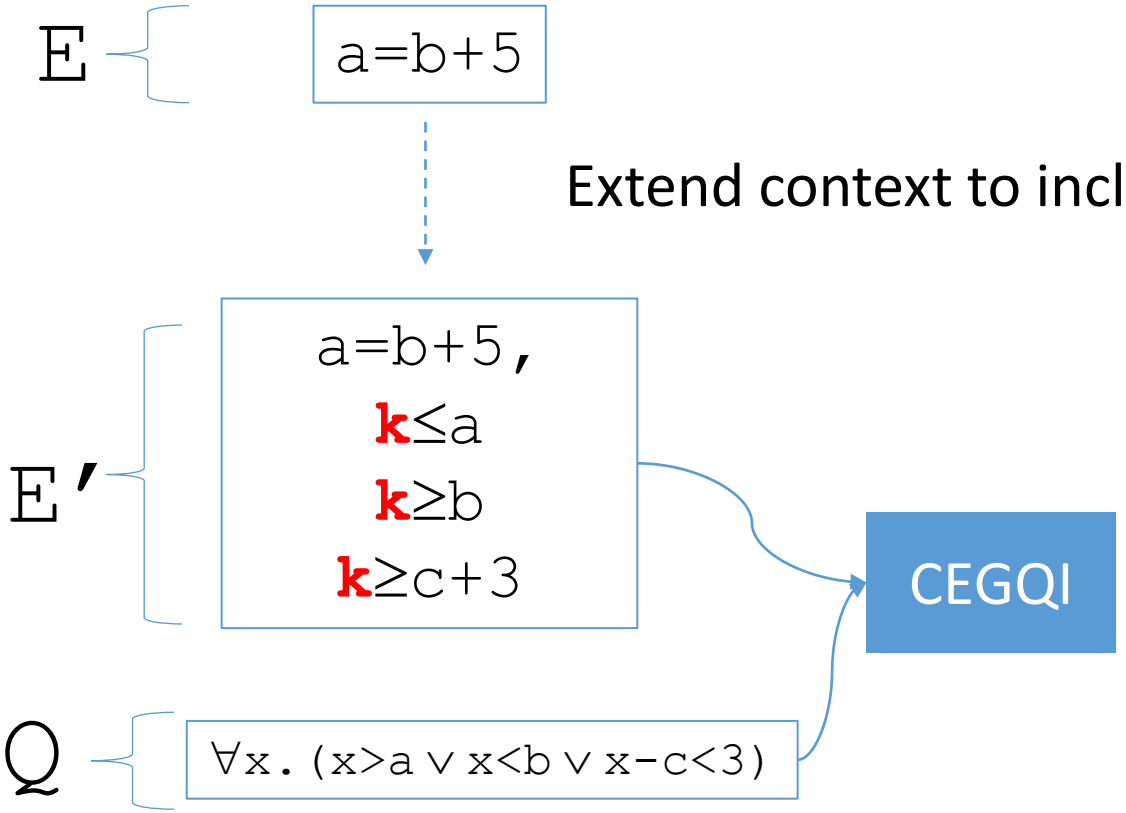


⇒With respect to *model-based instantiation*:

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

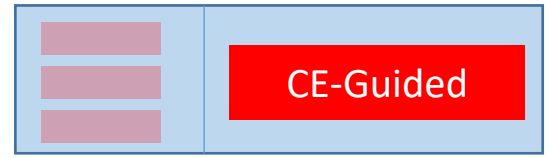
Counterexample-Guided Instantiation

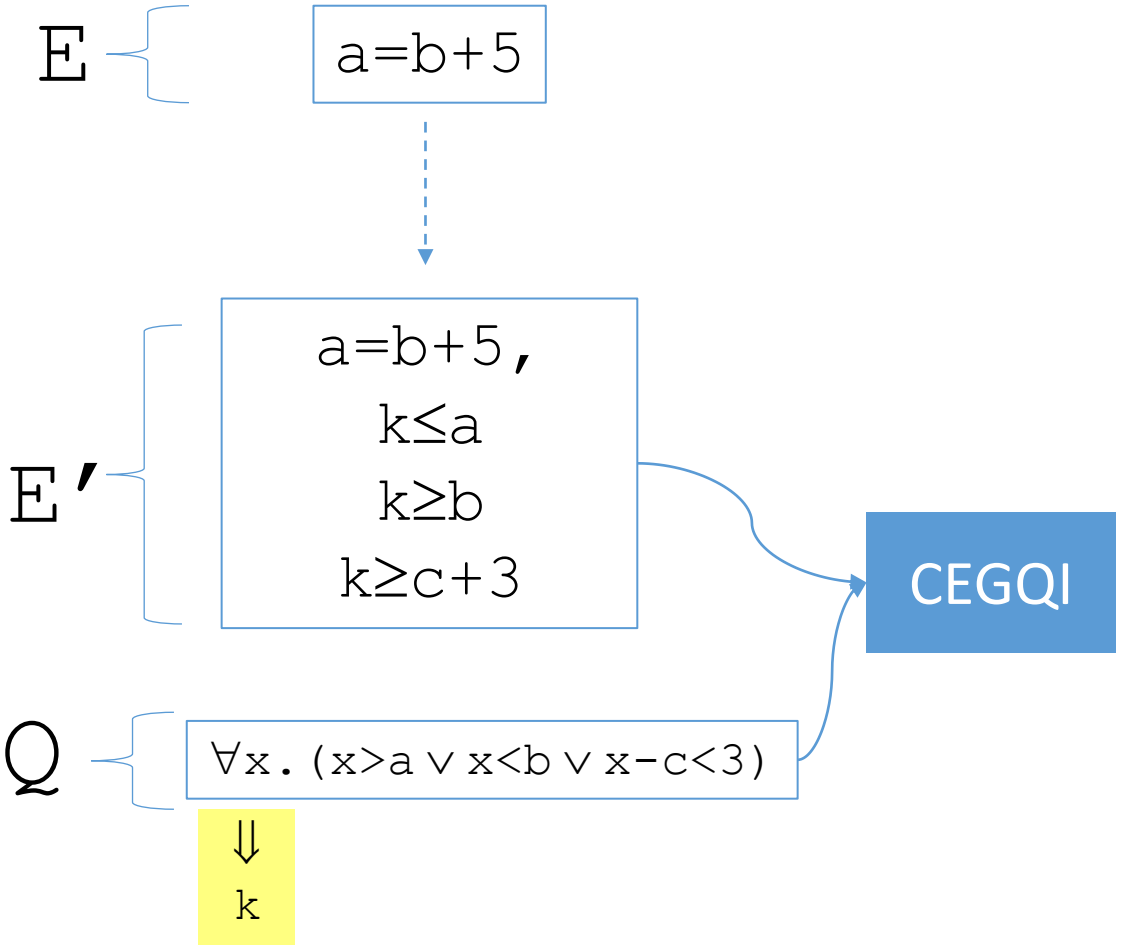




…has counterexample **k** iff ∃**k**.¬(**k**>a ∨ **k**<b ∨ **k**−c<3)

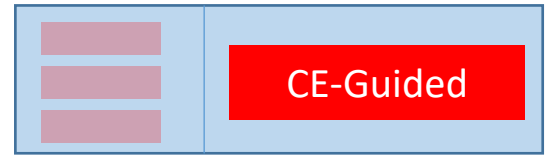
Counterexample-Guided Instantiation

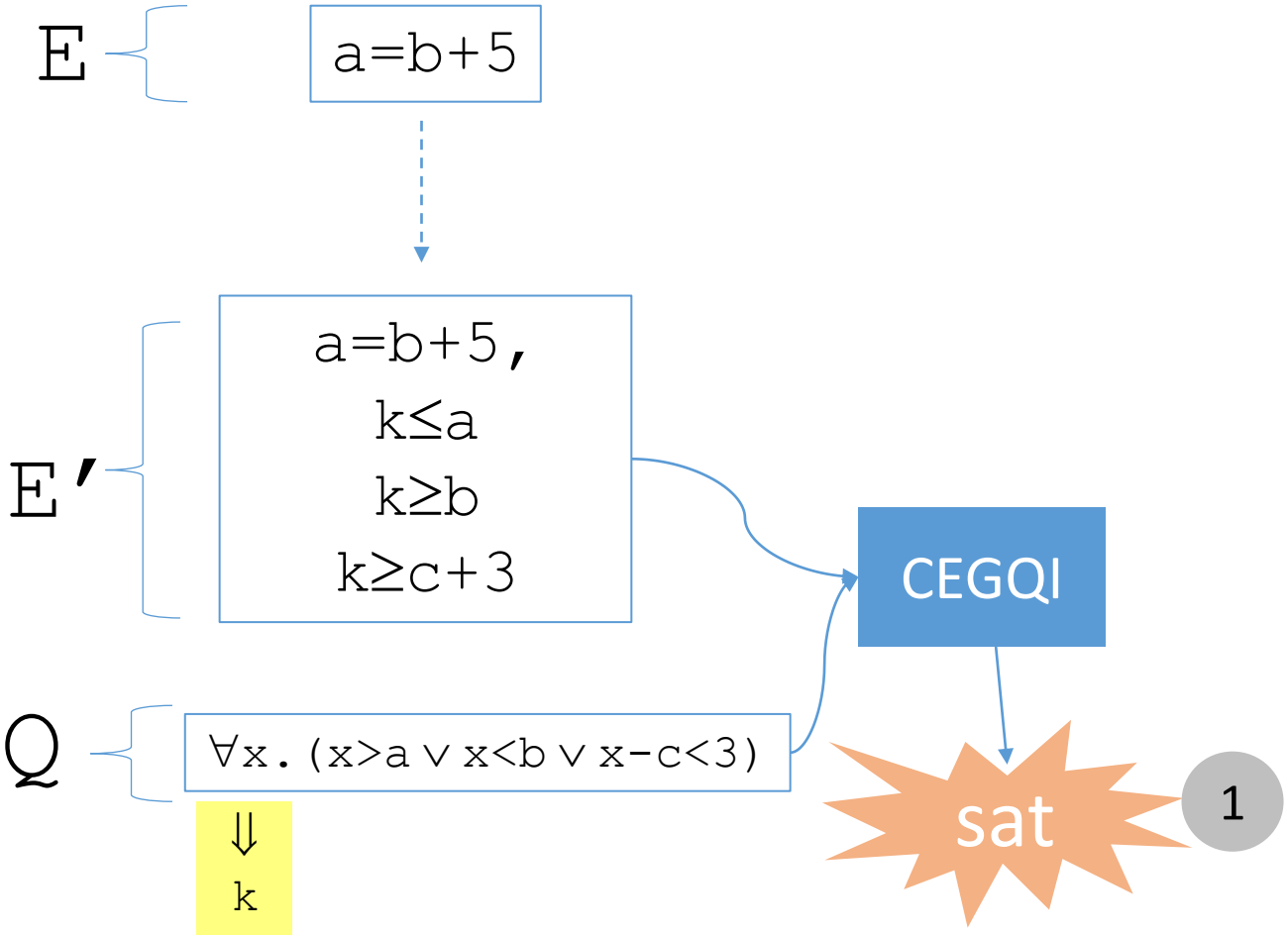




One of two cases…

Counterexample-Guided Instantiation

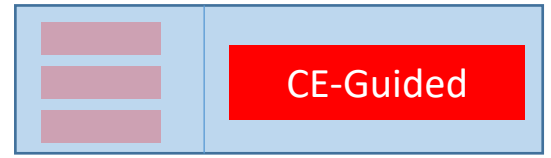


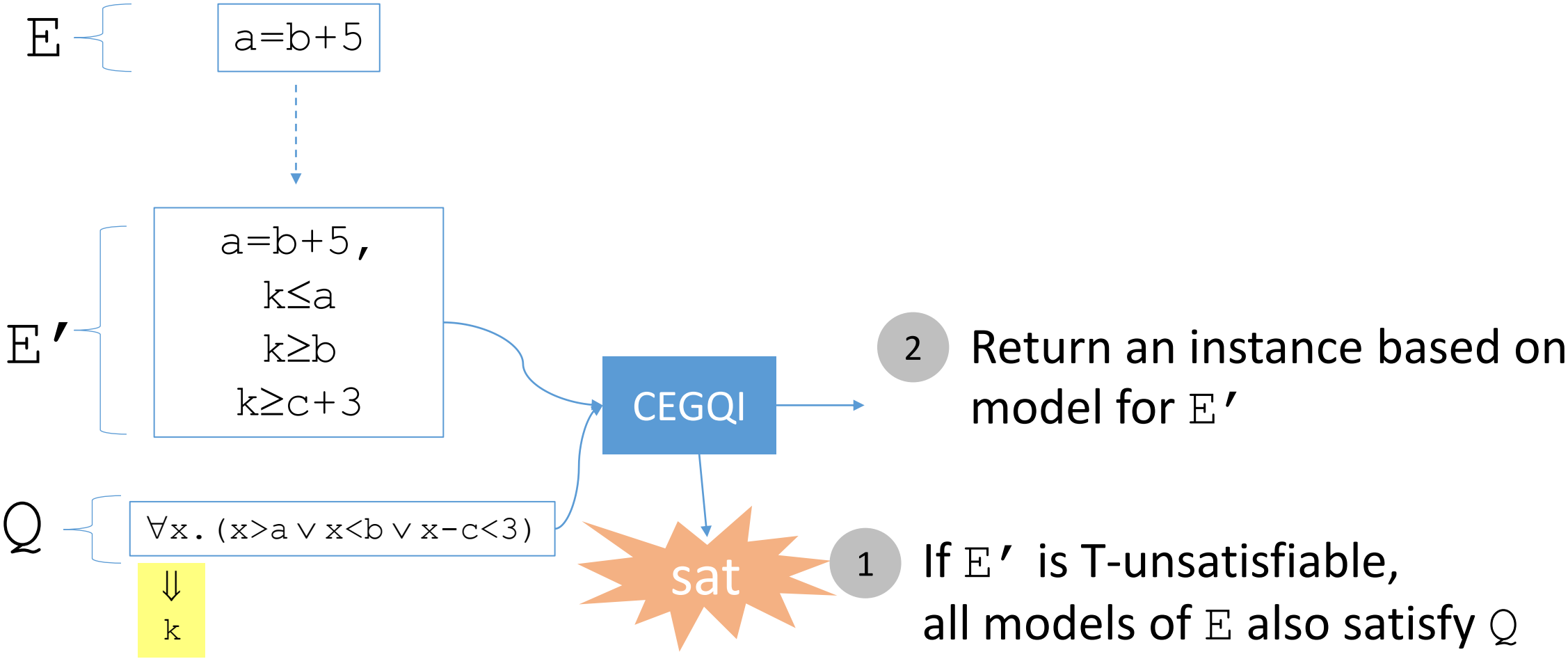


If E' is T-unsatisfiable,
all models of E also satisfy Q

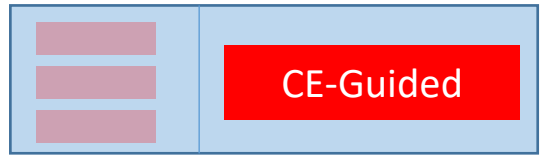
(since $\mathrm{E}' \equiv \mathrm{E} \cup \neg \mathrm{Q} \models_T \bot$ implies $\mathrm{E} \models_T \mathrm{Q}$)

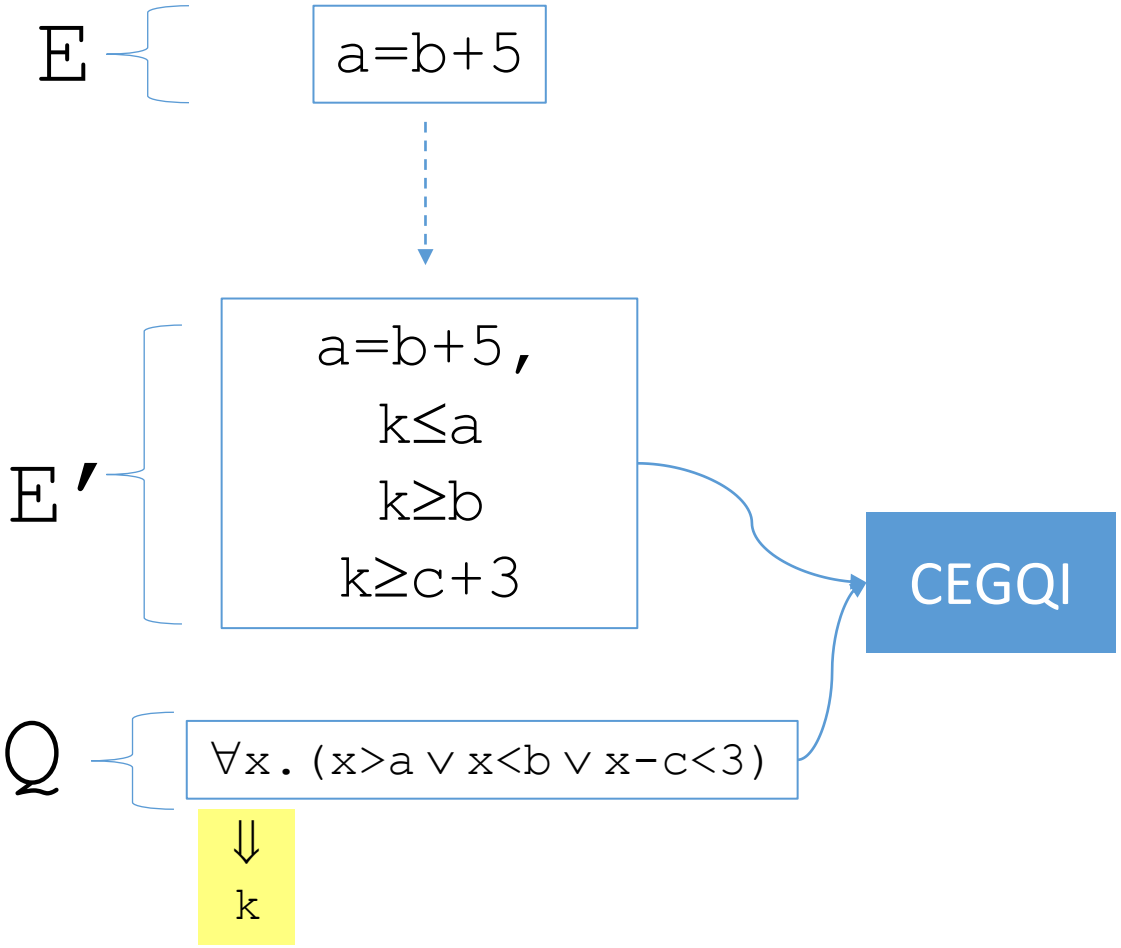
Counterexample-Guided Instantiation



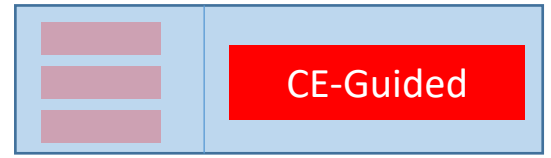


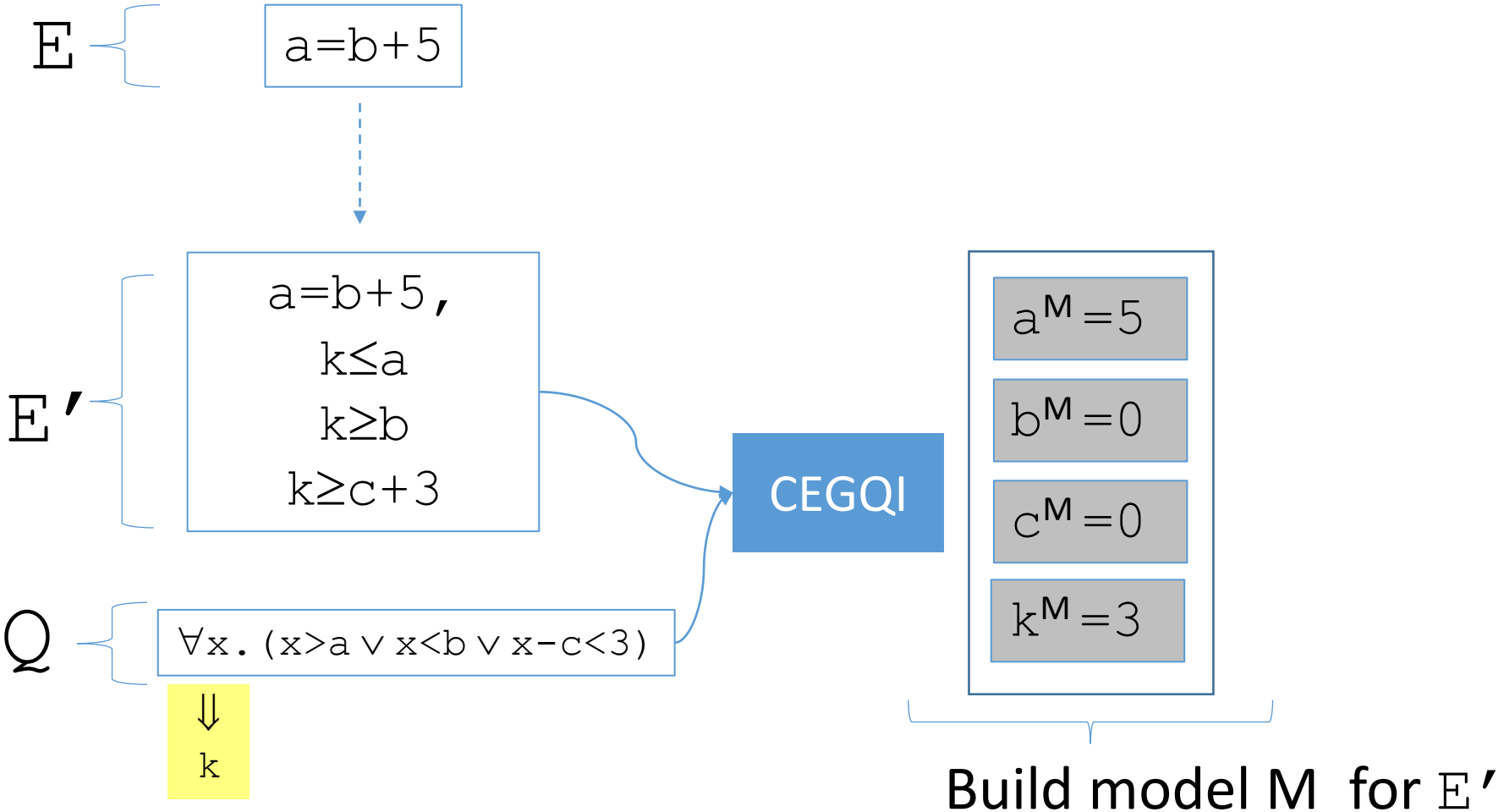
Counterexample-Guided Instantiation



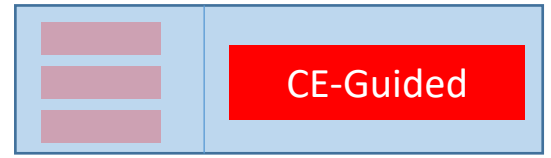


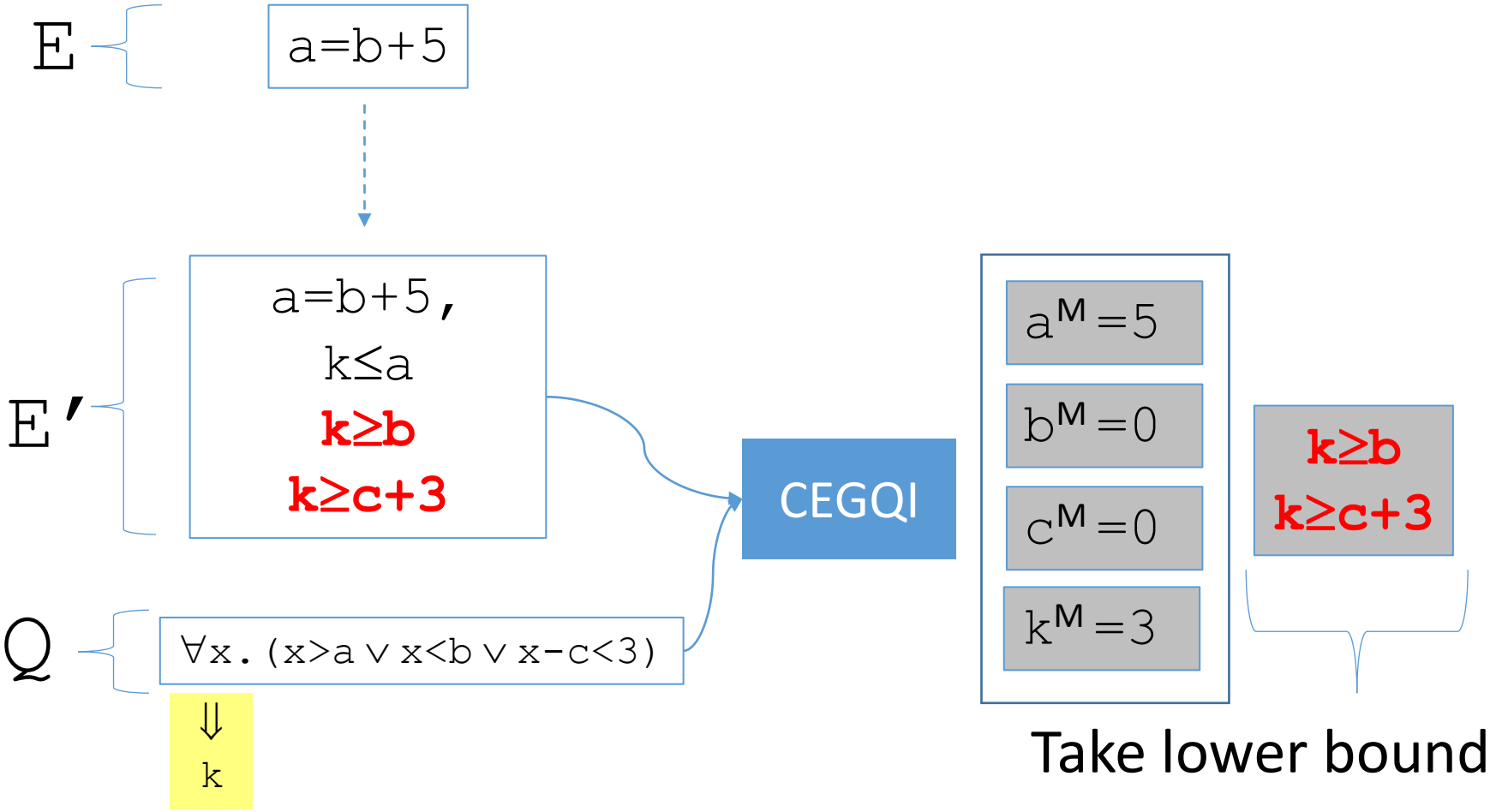
Counterexample-Guided Instantiation





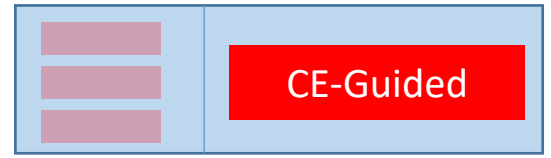
Counterexample-Guided Instantiation

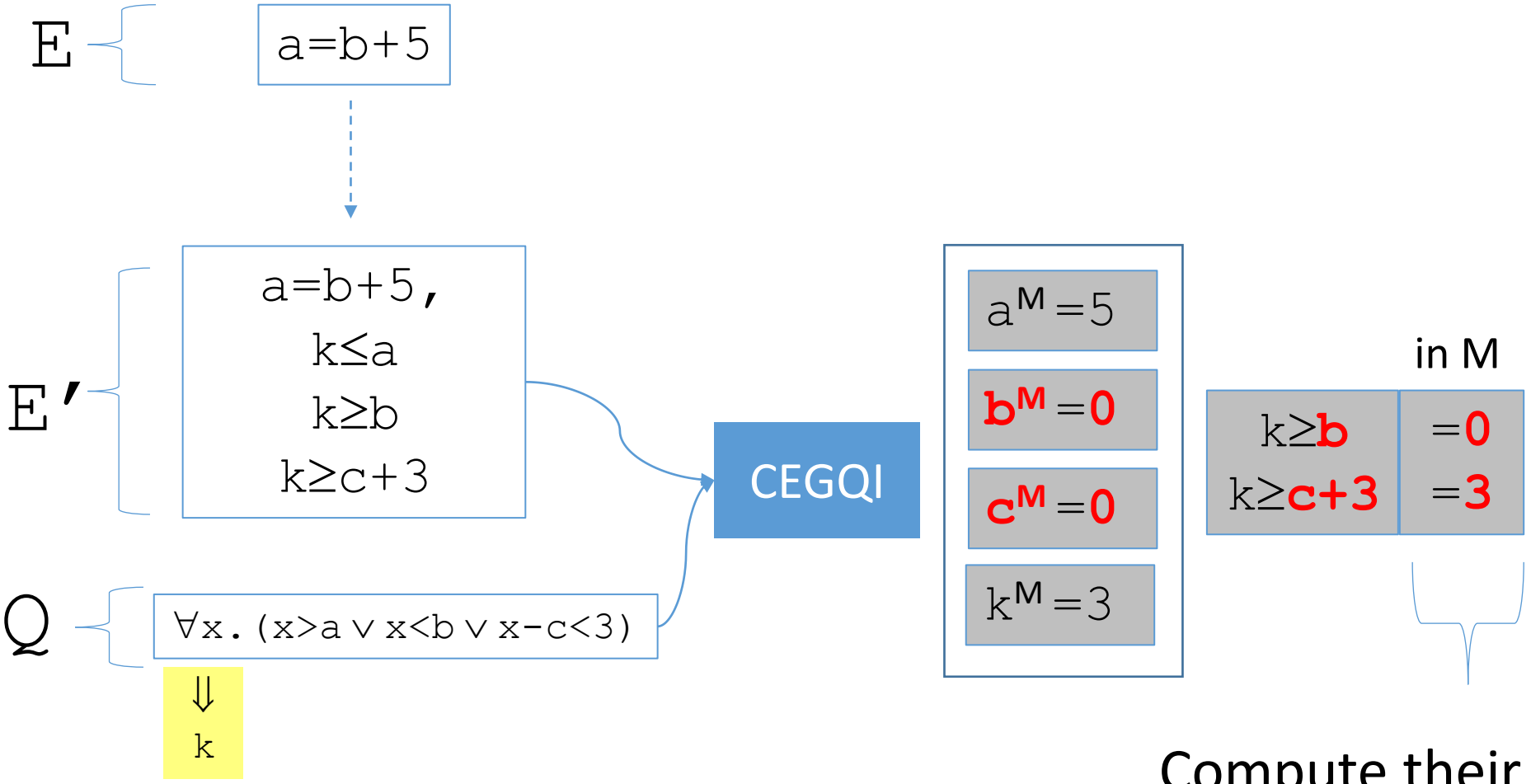




Take lower bounds of k in E'

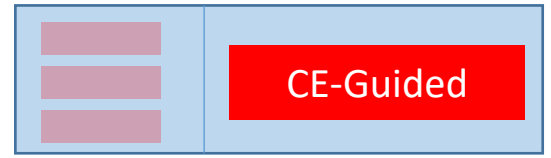
Counterexample-Guided Instantiation





Compute their value in 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

$a^M = 5$
$b^M = 0$
$c^M = 0$
$k^M = 3$

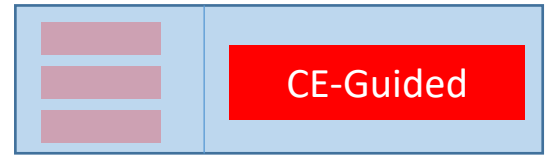
in M

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

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

Add instance for lower bound that is maximal in M

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

$a^M = 5$

$b^M = 0$

$c^M = 0$

$k^M = 3$

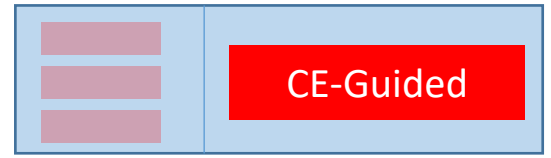
in M

k≥b	=0
k≥c+3	=3

∀x.(x>a ∨ x<b ∨ x-c<3) ⇒
c+3>a ∨ c+3<b

Simplify

Counterexample-Guided Instantiation



E

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

QF Solver

$a^M=5$

$b^M=0$

$c^M=0$

$k^M=3$

CEGQI

Q

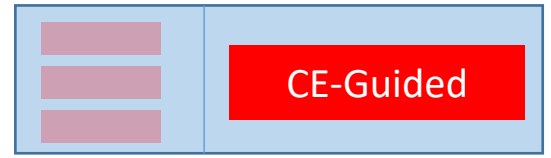
$\forall x.(x>a \vee x<b \vee x-c<3)$

in 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

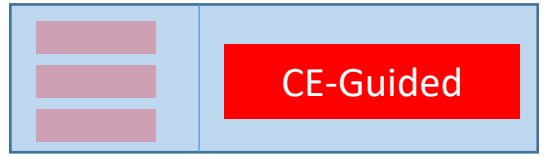


E { a=b+5, c+3<b

Q { ∀x.(x>a ∨ x<b ∨ x-c<3)

CEGQI

Counterexample-Guided Instantiation



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

Extend context to include counterexample

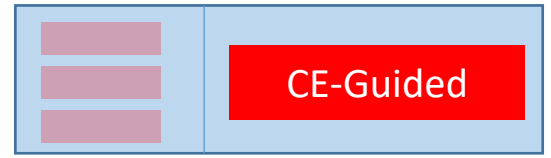
E′ $\{$ $a=b+5, c+3<b,$ **$k \leq a$** **$k \geq b$** **$k \geq c+3$**

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

$\Downarrow$ **k**

CEGQI

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

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

⇓
k

CEGQI

$a^M = 5$

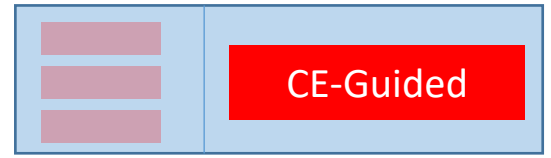
$b^M = 0$

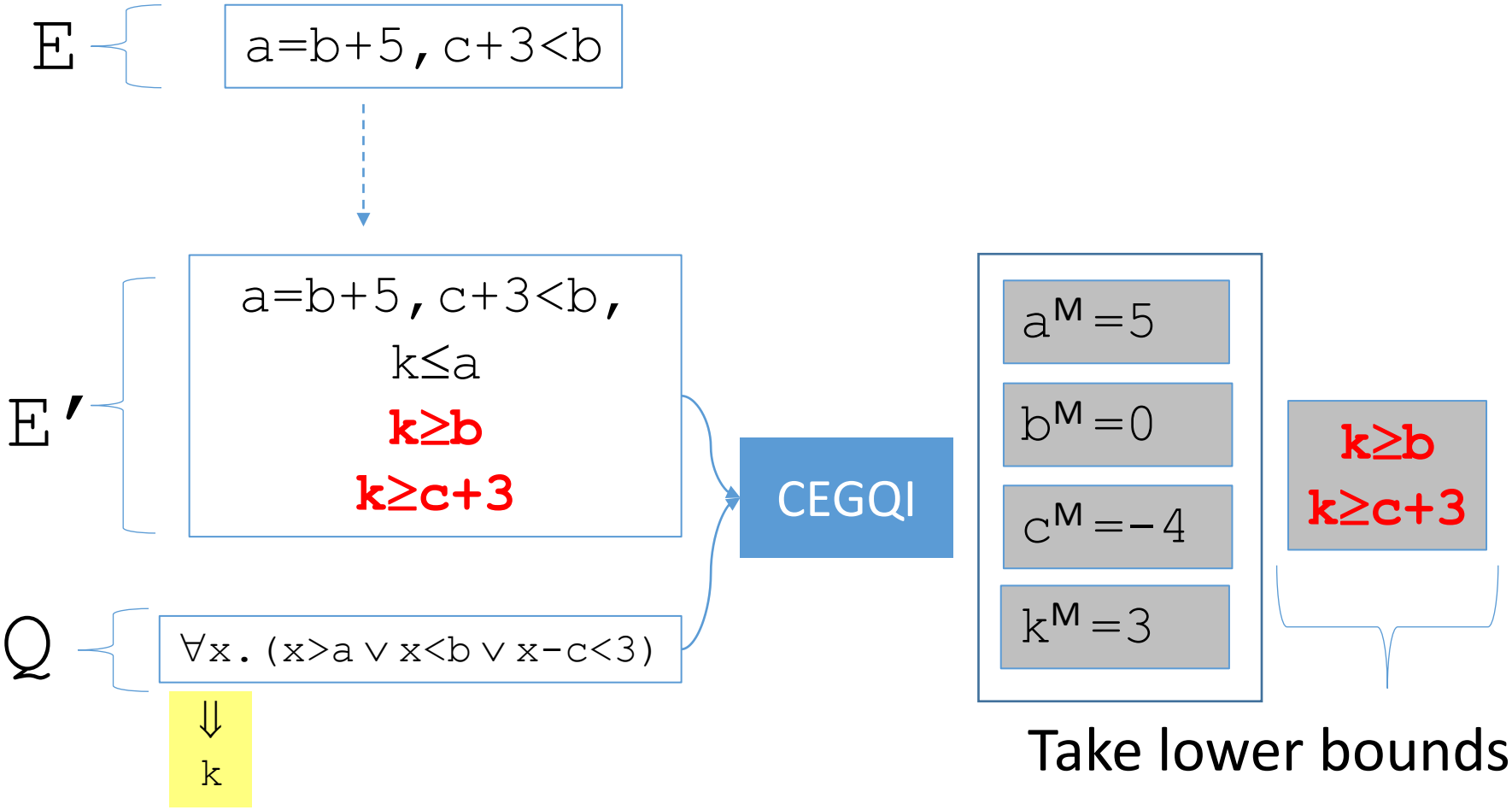
$c^M = -4$

$k^M = 3$

Build model M for E′

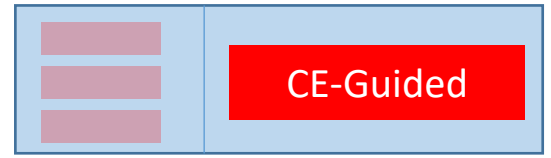
Counterexample-Guided Instantiation





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

$a^M = 5$

$b^M = 0$

$c^M = -4$

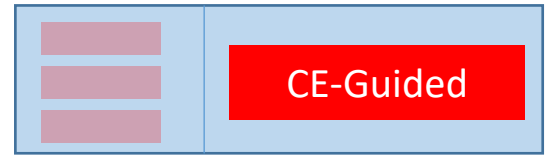
$k^M = 3$

in M

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

Compute their value in 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

$a^M=5$

$b^M=0$

$c^M=-4$

$k^M=3$

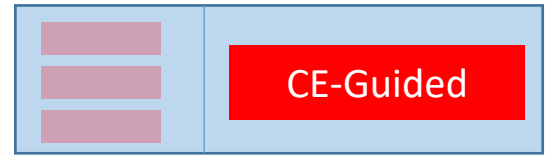
in M

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

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

Add instance for lower bound that is maximal in 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

$a^M=5$

$b^M=0$

$c^M=-4$

$k^M=3$

in M

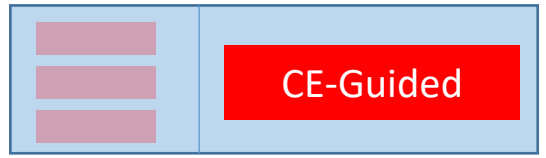
$k\geq b$	$=0$
$k\geq c+3$	$=-1$

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

$b>a \vee b<c+3$

Simplify

Counterexample-Guided Instantiation



E

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

QF Solver

CEGQI

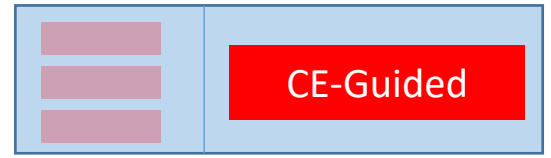
Q

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

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

$b>a \vee b<c+3$

Counterexample-Guided Instantiation



E

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

QF Solver

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

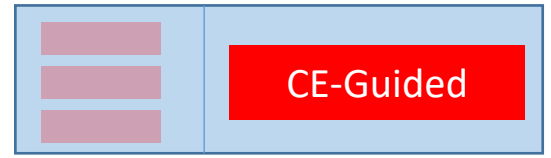
CEGQI

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, **c+3>a**

QF Solver

Backtrack previous decision **c+3>a** ∨ c+3<b

CEGQI

Q { ∀x.(x>a ∨ x<b ∨ x-c<3)

∀x.(x>a ∨ x<b ∨ x-c<3) ⇒
b>a ∨ b<c+3

Counterexample-Guided Instantiation

E

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

QF
Solver

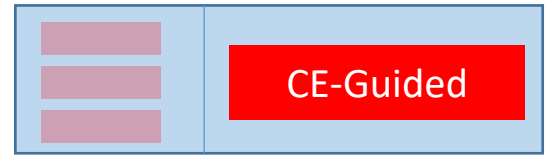
CEGQI

Q

∀x.(x>a ∨ x<b ∨ x-c<3)

∀x.(x>a ∨ x<b ∨ x-c<3) ⇒
b>a ∨ b<c+3

Counterexample-Guided Instantiation



E

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

Q

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

CEGQI

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
k≤a
k≥b
k≥c+3

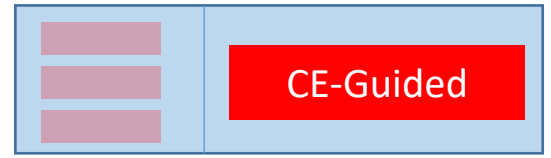
CEGQI

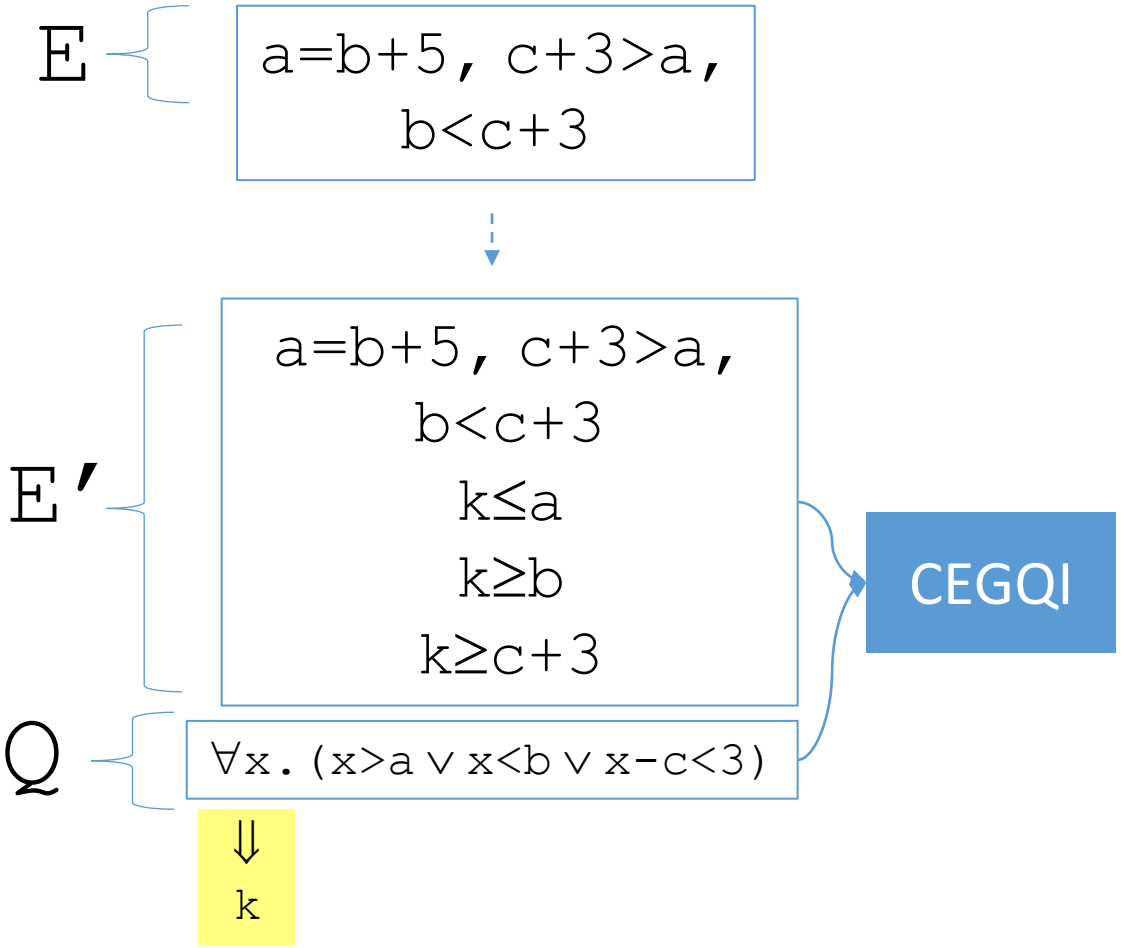
Q

∀x.(x>a ∨ x<b ∨ x-c<3)

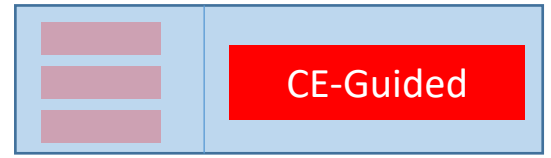
⇓
k

Counterexample-Guided Instantiation





Counterexample-Guided Instantiation



E

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

E′

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

$k \le a$

$k \ge b$

$k \ge c+3$

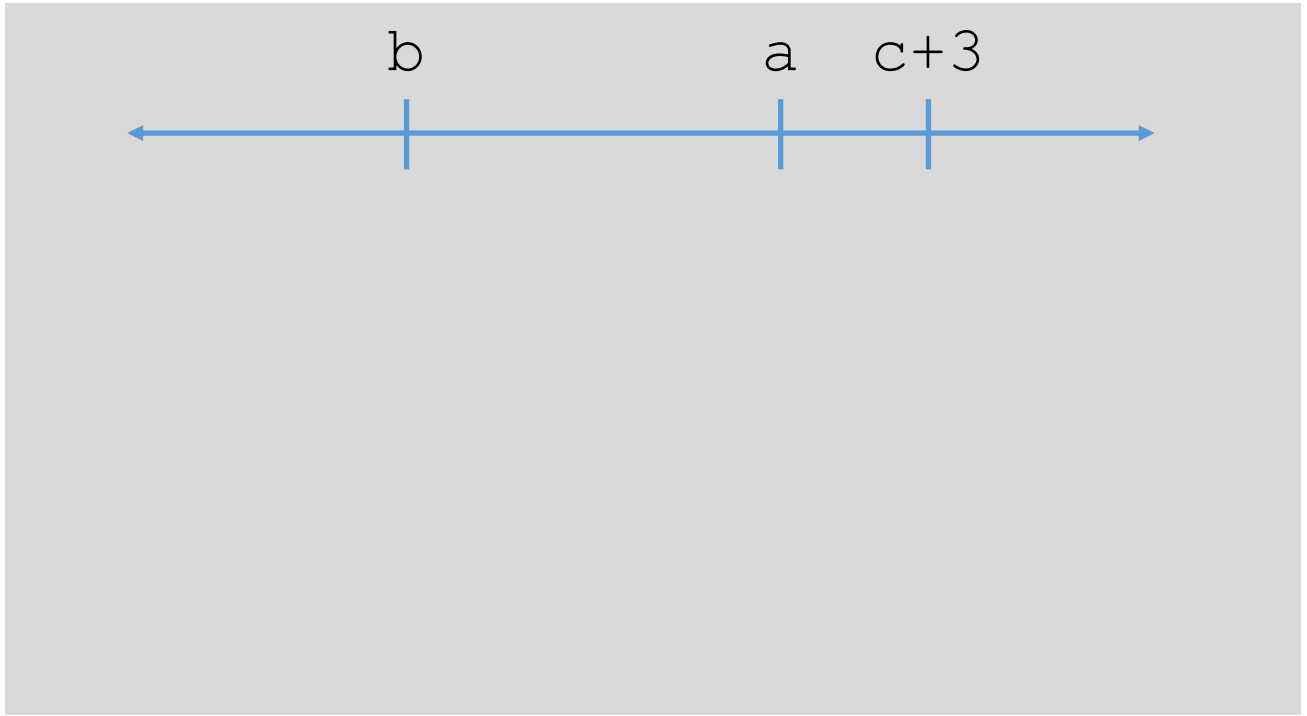
CEGQI

Q

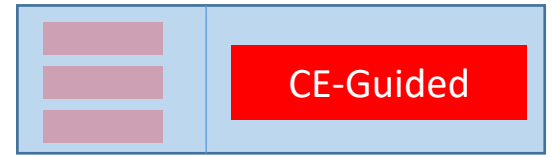
$\forall x.(x>a \vee x<b \vee x-c<3)$

$\Downarrow$

k



Counterexample-Guided Instantiation



E

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

E′

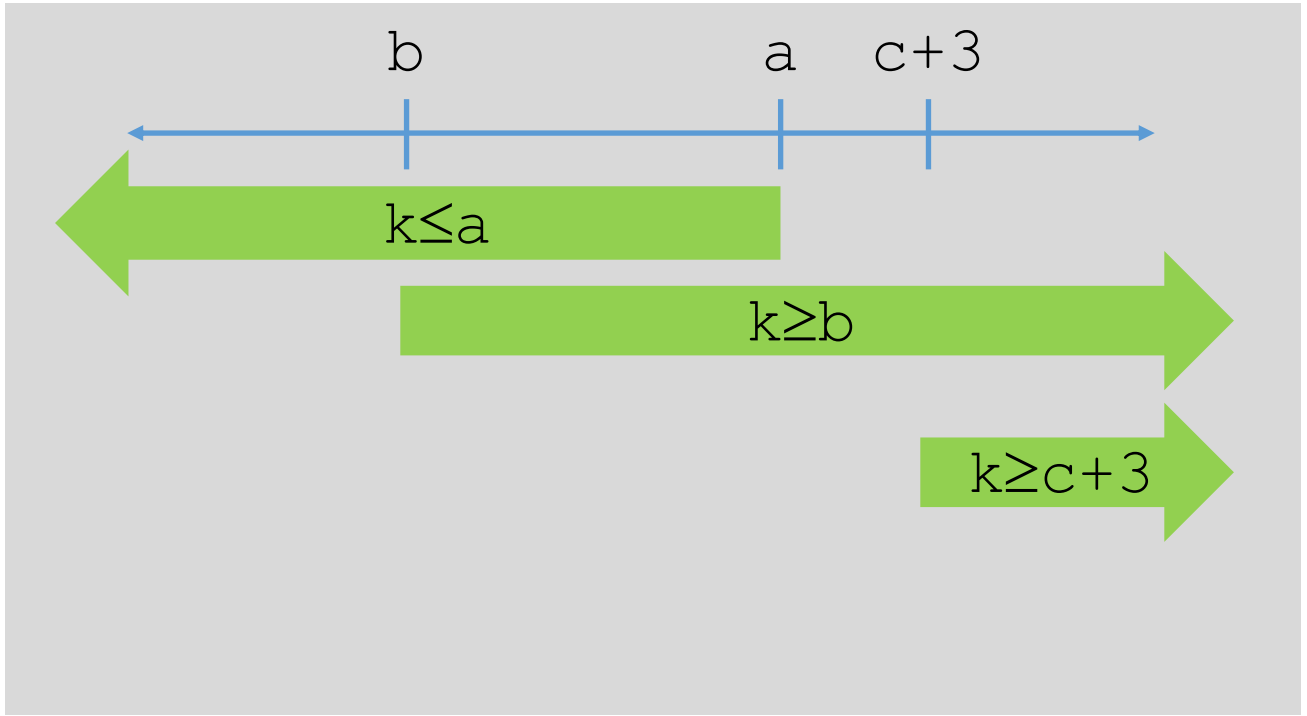
a=b+5, c+3>a,
b<c+3
k≤a
k≥b
k≥c+3

Q

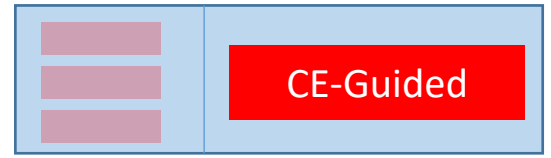
∀x.(x>a ∨ x<b ∨ x-c<3)

⇓
k

CEGQI



Counterexample-Guided Instantiation



E

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

E′

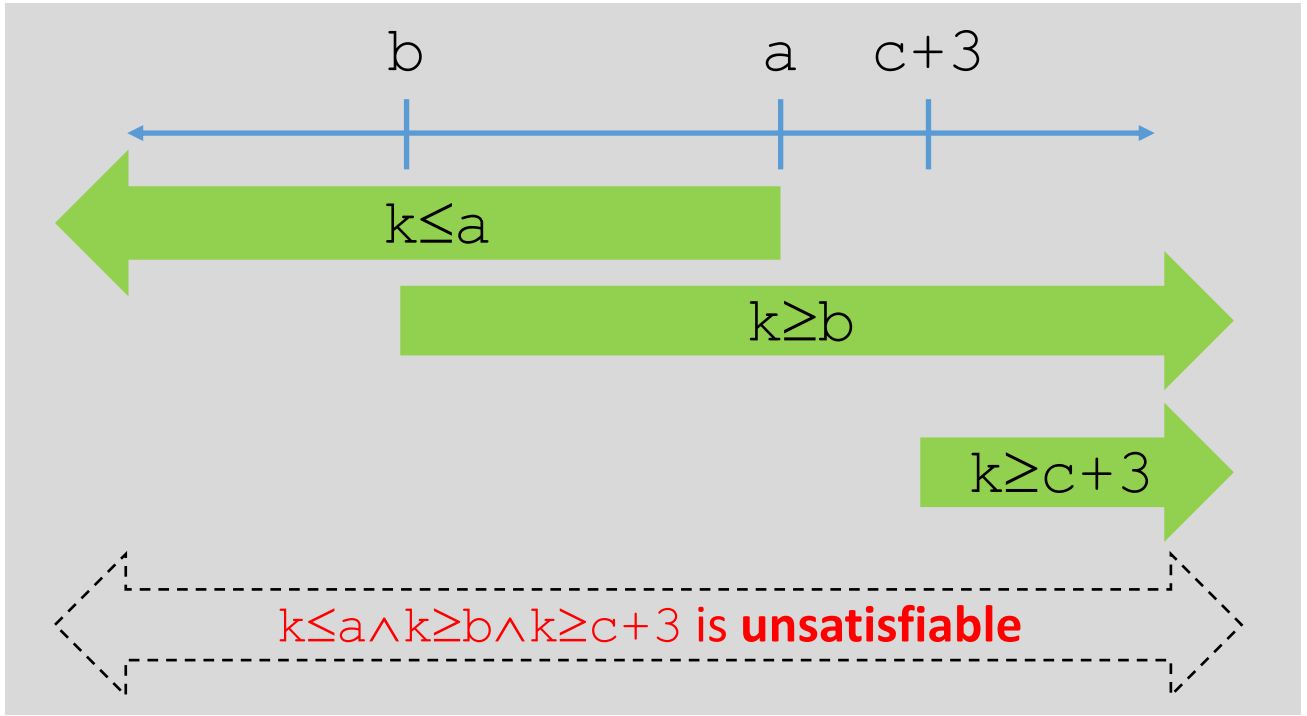
a=b+5, c+3>a,
b<c+3
k≤a
k≥b
k≥c+3

Q

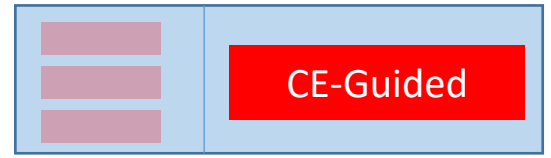
∀x.(x>a ∨ x<b ∨ x-c<3)

⇓
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≤a
k≥b
k≥c+3

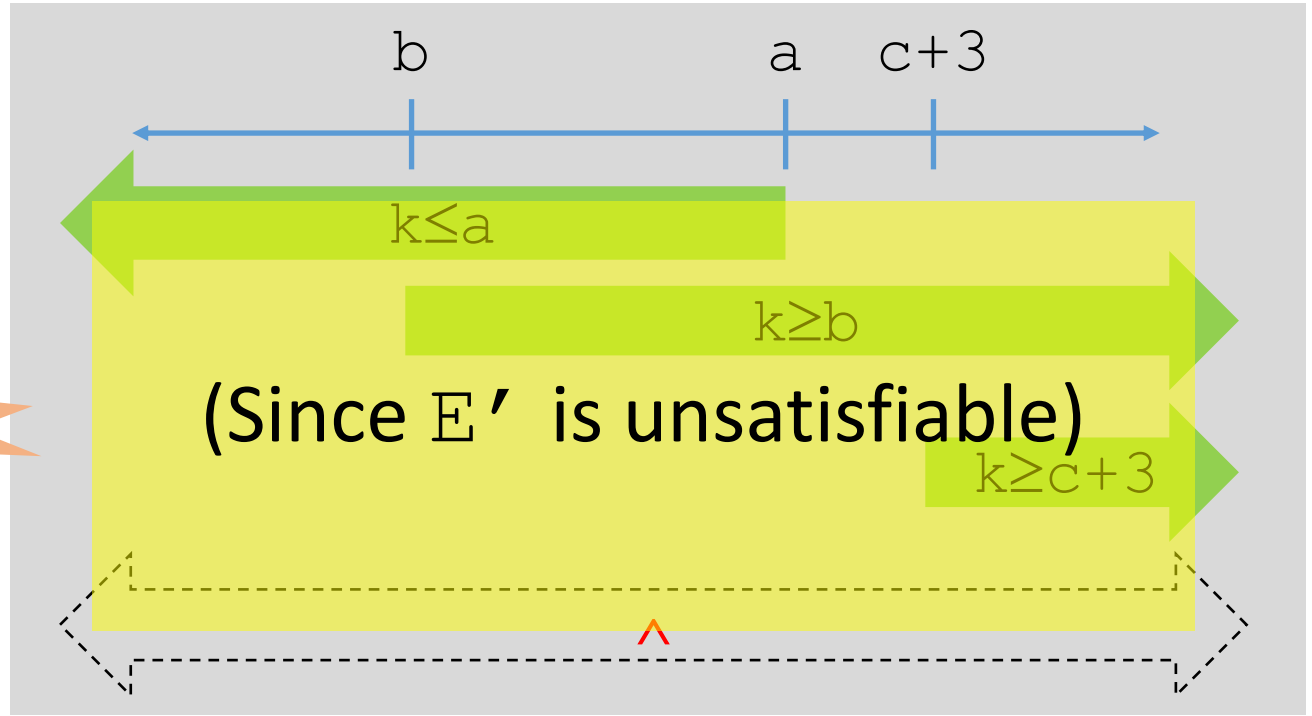
Q

∀x.(x>a ∨ x<b ∨ x-c<3)

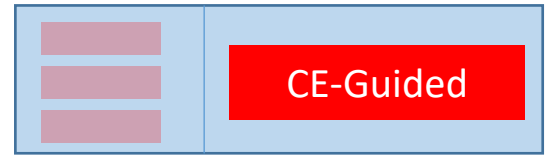
⇓
k

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

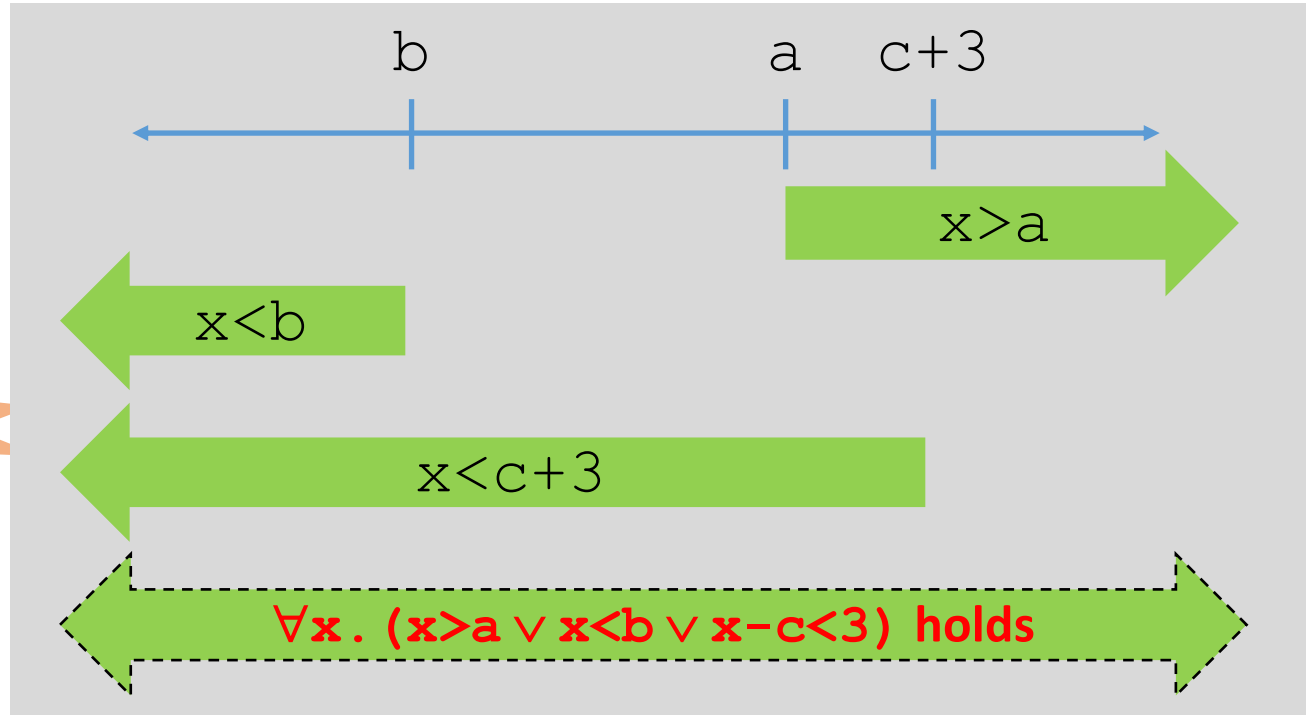
∀x. (x>a ∨ x<b ∨ x-c<3)

⇓
k

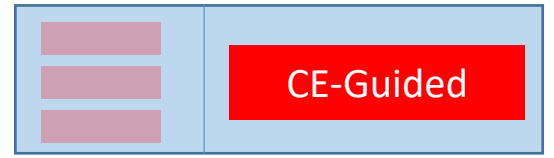
CEGQI

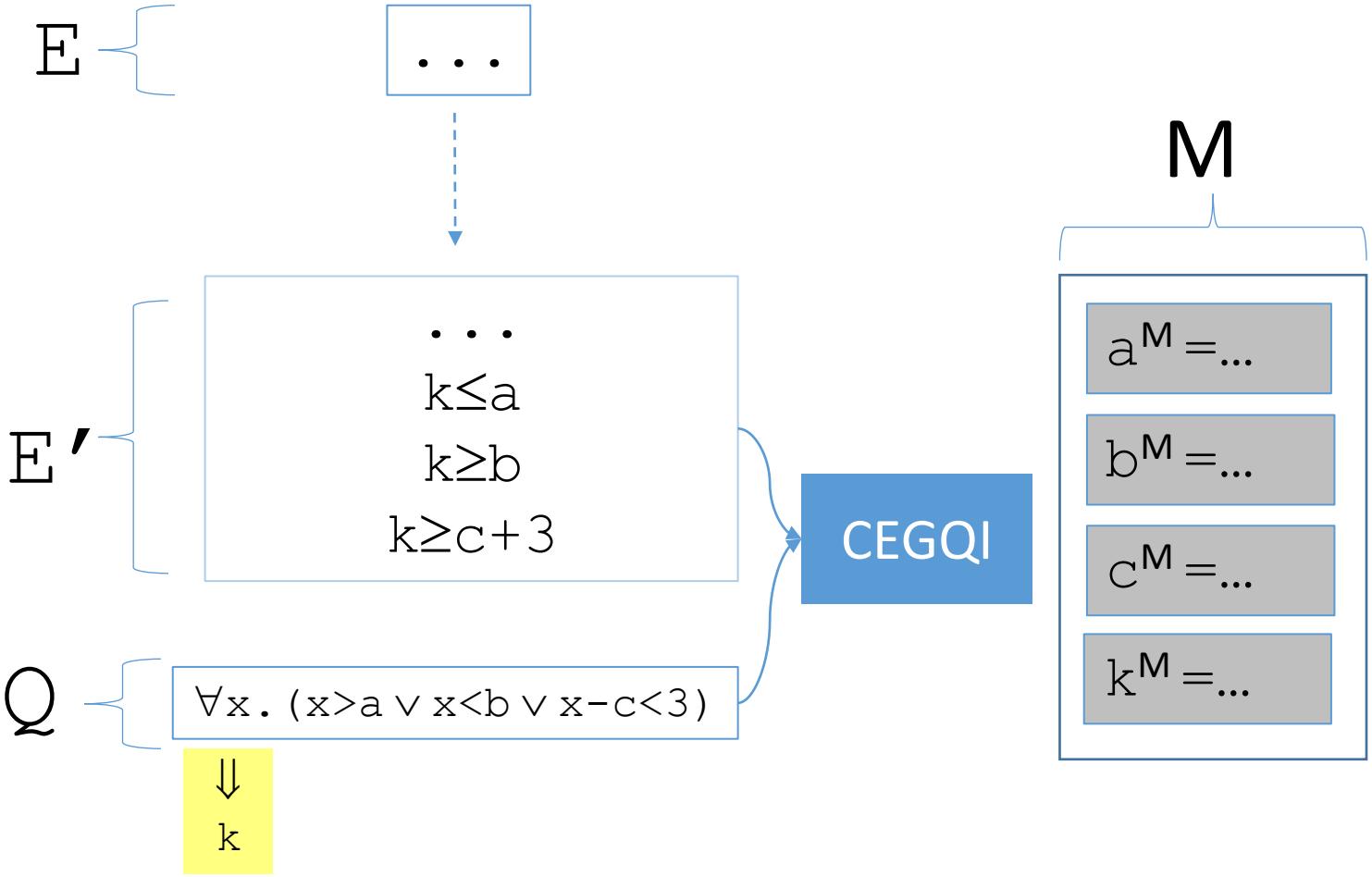
sat

Dually:



Summary





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

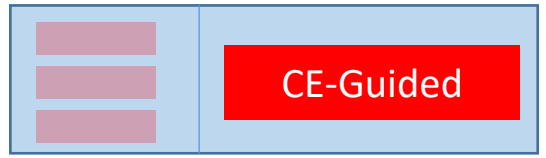
⇒ *In general:*

Requires a

selection function

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

Selection Functions

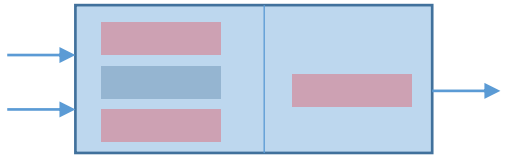


- Selection function gives decision procedure for $\forall$ in various theories:
  - Linear real arithmetic (LRA)
    - Maximal lower (minimal upper) bounds
      [Loos+Wiespfenning 93]
      $l_1 < k, \ldots, l_n < k \rightarrow \{x \rightarrow l_{max} + \delta\}$
      …may involve virtual terms δ, ∞
    - Interior point method:
      [Ferrante+Rackoff 79]
      $l_{max} < k < u_{min} \rightarrow \{x \rightarrow (l_{max} + u_{min}) / 2\}$
  - Linear integer arithmetic (LIA)
    - Maximal lower (minimal upper) bounds (+c)
      [Cooper 72]
      $l_1 < k, \ldots, l_n < k \rightarrow \{x \rightarrow l_{max} + c\}$
  - Bitvectors/finite domains
    - Value instantiations
    - Invertibility conditions **[Niemetz et al 2018]**
  - Datatypes, …

$\Rightarrow$ ***Termination argument for each****: enumerate at most a finite number of instances*

Summary

- Quantifier Instantiation Beyond E-matching, via:
  - **Conflict-based, Model-Based Instantiation**
    - Effective in practice for $\forall$ that highly involve UF
    - Conflict-Based is useful for “unsat”
    - Model-Based is useful for “sat”
      - Alternatively, enumerative instantiation, which is more tailored for “unsat”
  - **Counterexample-guided Instantiation**
    - Decision procedure for $\forall$+LRA, $\forall$+LIA, $\forall$+BV, …



Future Work

- How do we develop instantiations procedures for **∀+new theories**?
  - Ongoing work on ∀+bit-vectors
  - Also interested in ∀+strings, ∀+floating-points, ∀+finite sets, etc.
- How can we combine instantiation-based techniques with **superposition**?
- Other techniques for finding instantiations?
  - E.g. those based on syntax-guided synthesis

- Techniques in this talk implemented in CVC4
  - Open source
  - Available at : https://cvc4.github.io/

- …Thanks for listening!