Instantiation-Based Methods for First-Order Logic

Andrew Reynolds

April 4, 2017

THE UNIVERSITY OF IOWA

In this Talk

$$(\forall x.P(x) \vee f(b)=b+1) \wedge \exists y.(\neg P(y) \wedge f(y)<y)$$

- Focus on techniques for establishing *T-satisfiability* of formulas with:

In this Talk

$$(\forall x.P(x) \lor f(b)=b+1) \land \exists y.(\neg P(y) \land f(y)<y)$$

- Focus on techniques for establishing *T-satisfiability* of formulas with:
  - Boolean structure

In this Talk

$$(\forall x. P(x) \lor f(b)=b+1) \land \exists y. (\neg P(y) \land f(y)<y)$$

- Focus on techniques for establishing *T-satisfiability* of formulas with:
  - Boolean structure
  - Constraints in a background theory T, e.g. UFLIA

In this Talk

$$(\forall x . P(x) \lor f(b)=b+1) \land \exists y . (\neg P(y) \land f(y)<y)$$

- Focus on techniques for establishing *T-satisfiability* of formulas with:
  - Boolean structure
  - Constraints in a background theory T, e.g. UFLIA
  - ***Existential and Universal Quantifiers***

Outline

- Background
- Satisfiabilty Modulo Theories (SMT) solver architecture

…and how it extends to ∀ reasoning via quantifier instantiation

$$\forall x. \psi[x] \Rightarrow \psi[t]$$

- Recent strategies for quantifier instantiation:
  - E-matching, conflict-based, model-based, counterexample-guided

Quantified formulas ∀ in SMT

- Are of importance to applications:
  - 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: $\forall i:input.\exists o:output.R[o,i]$
  - Planning: $\exists p:plan.\forall t:time.F[P,t]$
  - Knowledge representation: $\forall xy:Person.sibling(x,y)\Rightarrow mother(x)=mother(y)$

Quantified formulas ∀ in SMT

- Are of importance to applications:
  - 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: $\forall i:input.\exists o:output.R[o,i]$
  - Planning: $\exists p:plan.\forall t:time.F[P,t]$
  - Knowledge representation: $\forall xy:Person.sibling(x,y)\Rightarrow mother(x)=mother(y)$
- Are very challenging in theory:
  - Establishing T-satisfiability of formulas with ∀ is generally undecidable

Quantified formulas ∀ in SMT

- Are of importance to applications:
  - Automated theorem proving:
    - Background axioms {∀x.g(e,x)=g(x,e)=x, ∀x.g(x,g(y,z))=g(g(x,y),x),∀x.g(x,i(x))=e}
  - Software verification:
    - Unfolding ∀x.foo(x)=bar(x+1), code contracts ∀x.pre(x)⇒post(f(x))
    - Frame axioms ∀x.x≠t ⇒ A'(x)=A(x)
  - Function Synthesis: ∀i:input.∃o:output.R[o,i]
  - Planning: ∃p:plan.∀t:time.F[P,t]
  - Knowledge representation: ∀xy:Person.sibling(x,y)⇒mother(x)=mother(y)
- Are very challenging in theory:
  - Establishing T-satisfiability of formulas with ∀ is generally undecidable
- Can be handled well in practice:
  - Efficient decision procedures for decidable fragments
  - Heuristic techniques have high success rates in the general case

Background: *Quantifiers*

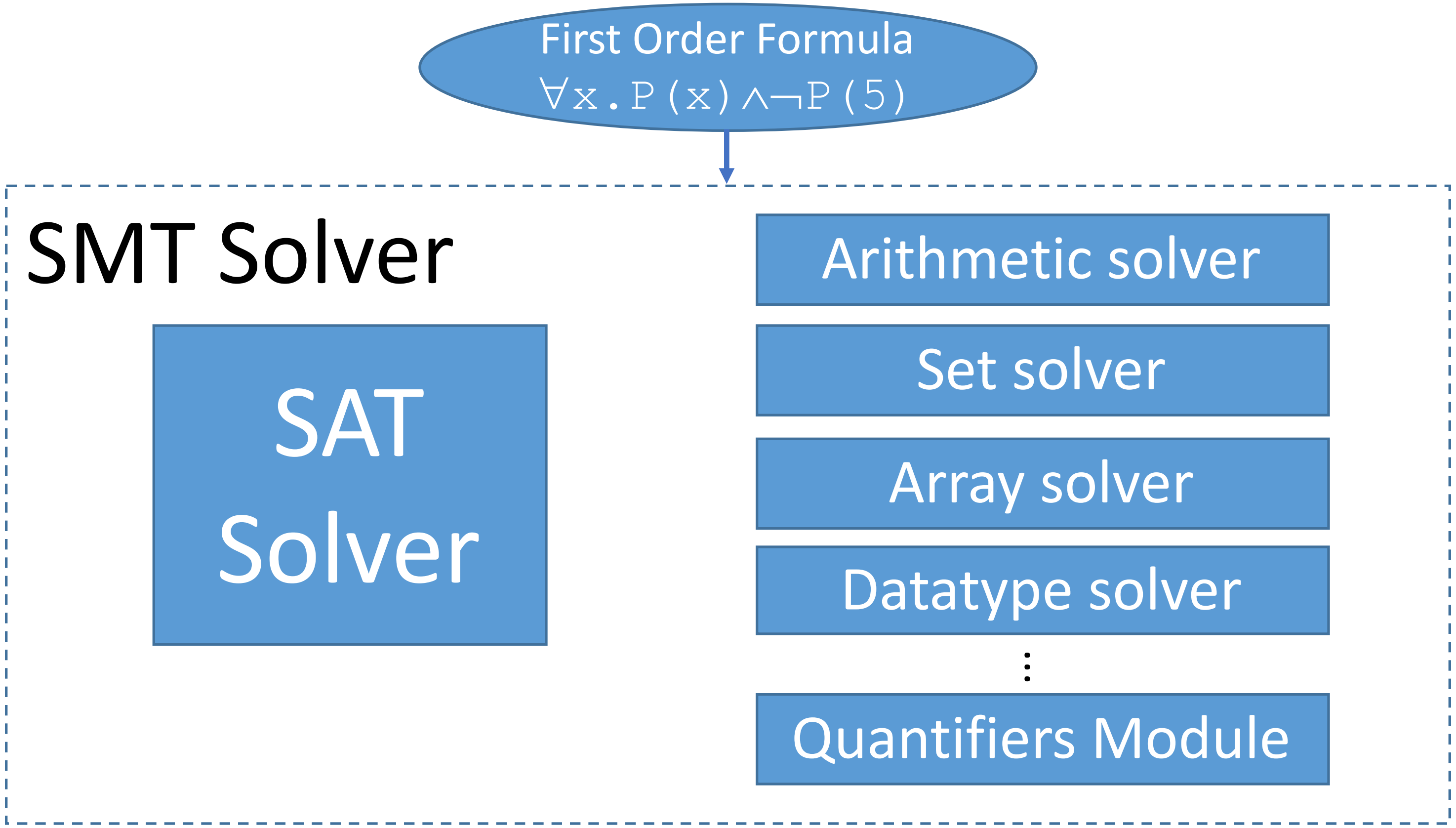
- Universal quantification:

$$\underbrace{\forall \mathtt{x{:}Int.P(x)}}_{\mathtt{P}\text{ is true for all integers }\mathtt{x}}$$

- Existential quantification:

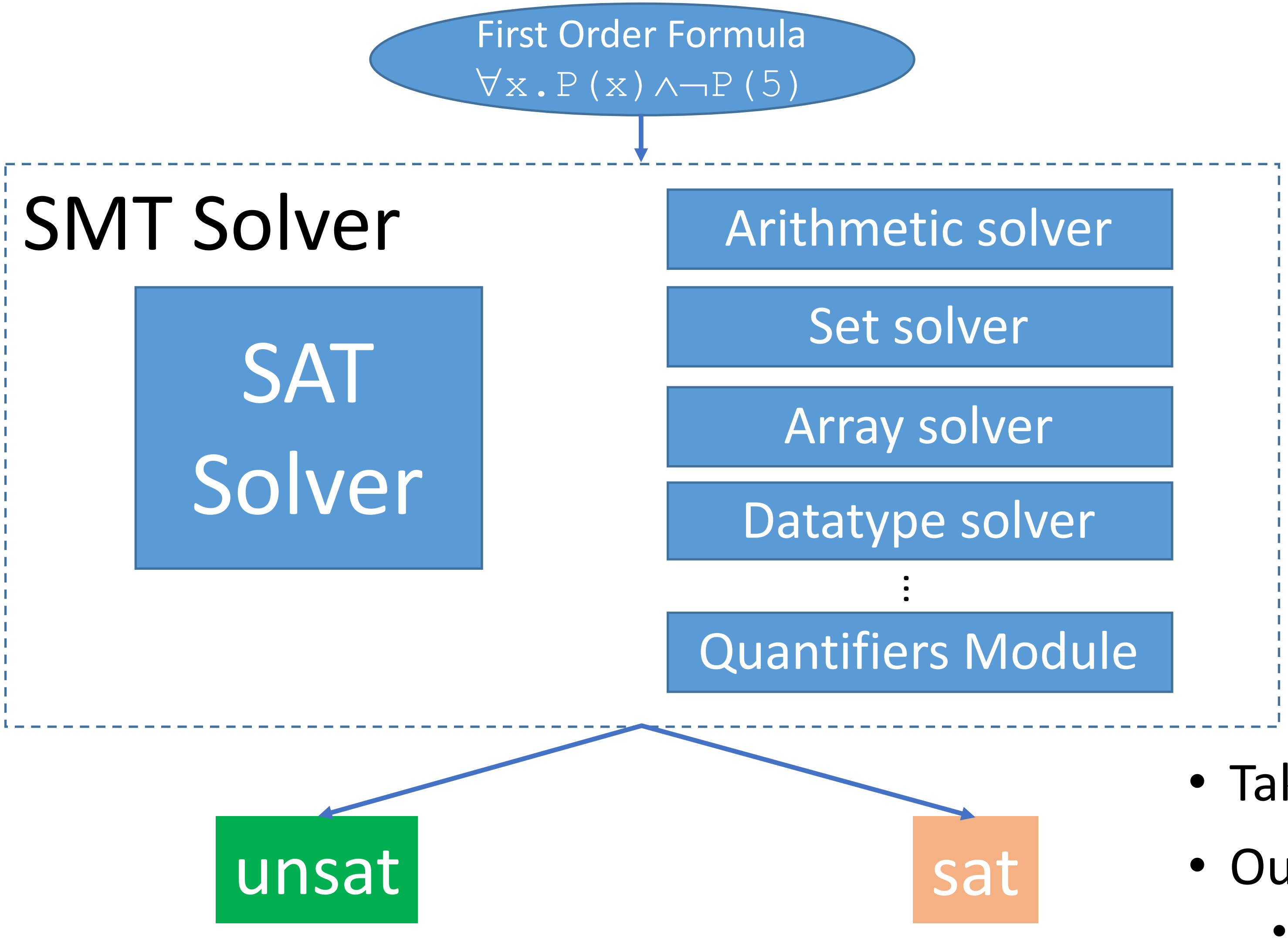
$$\underbrace{\exists \mathtt{x{:}Int.}\neg\mathtt{Q(x)}}_{\mathtt{Q}\text{ is false for some integer }\mathtt{x}}$$

Satisfiability Modulo Theories (SMT) Solvers



- Combination of propositional (SAT) solver, theory solver(s), quantifiers module

Satisfiability Modulo Theories (SMT) Solvers



First Order Formula

$\forall x. P(x) \land \neg P(5)$

SMT Solver

SAT Solver

Arithmetic solver

Set solver

Array solver

Datatype solver

⋮

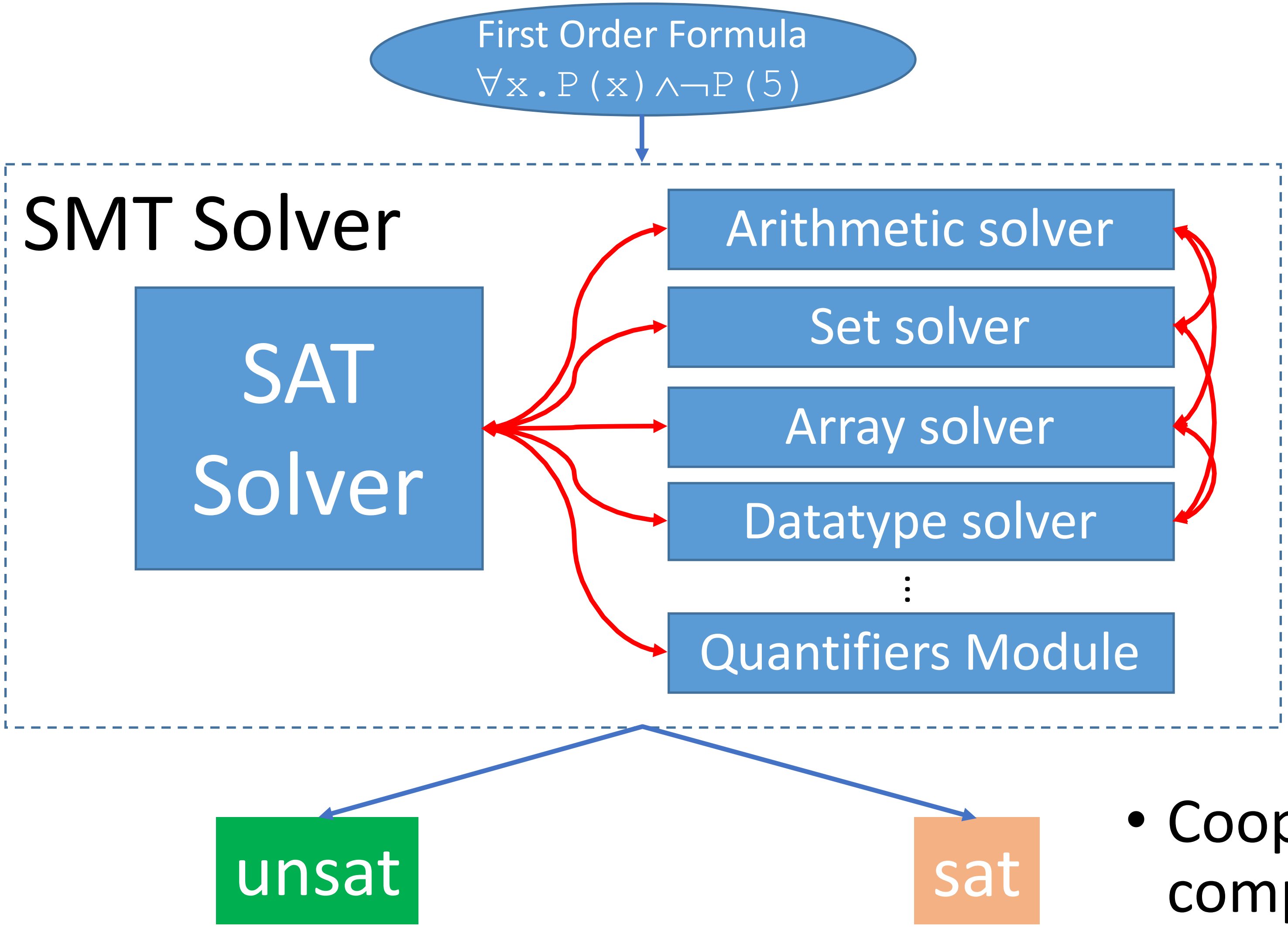
Quantifiers Module

unsat

sat

- Takes input first-order formula
- Outputs “sat” or “unsat”
  - “sat” if and only if input has a model

Satisfiability Modulo Theories (SMT) Solvers



- Cooperative interaction between components

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
-------------------------------
∀x.P(x)
¬P(5)
```

Signature

Initial input

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
∀x.P(x)
¬P(5)
∀x.P(x)⇒P(5)
```

Signature

Initial input

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

Signature

$\forall x.P(x)$

$\neg P(5)$

Initial input

$\forall x.P(\mathbf{x}) \Rightarrow P(\mathbf{5})$

Learned clauses

Instantiate x→5

Introductory Examples

`P : Int -> Bool`
`Q : Int -> Bool`
	Signature

`∀x.P(x)`
`¬P(5)`
	Initial input

`∀x.P(x)⇒P(5)`	Learned clauses

⇒ This set is *unsatisfiable* at the propositional level

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

A

¬ B

A ⇒ B

Signature

Initial input

Learned clauses

⇒ This set is *unsatisfiable* at the propositional level

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
∀x.P(x)
¬P(5)∨¬P(3)
```

Signature

Initial input

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
```

Signature

```
--------------------------------
```

$\forall x.P(x)$

$\neg P(5) \vee \neg P(3)$

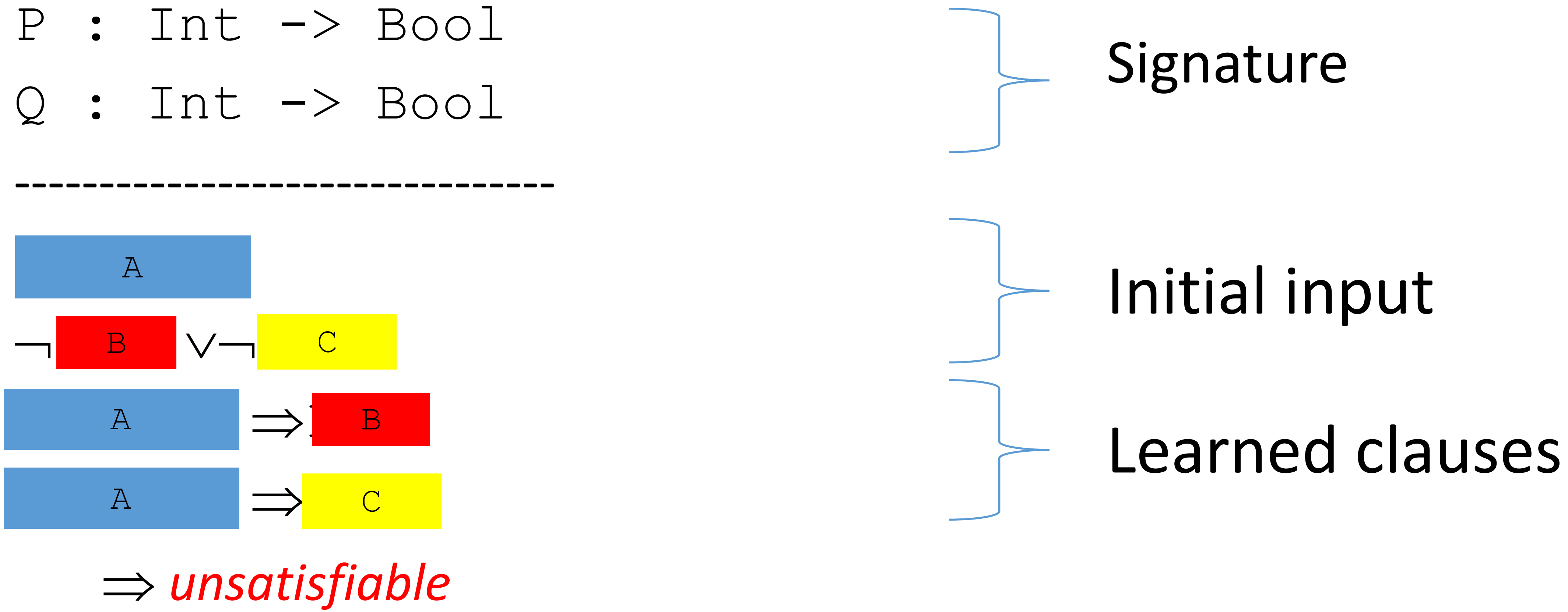
Initial input

$\forall x.P(\mathbf{x}) \Rightarrow P(\mathbf{5})$

$\forall x.P(\mathbf{x}) \Rightarrow P(\mathbf{3})$

Learned clauses

Introductory Examples



```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

A

¬ B ∨¬ C

A ⇒ B

A ⇒ C

⇒ *unsatisfiable*

Signature

Initial input

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
∀x.P(x)∨Q(x)
¬P(7)∧¬P(2)∧¬Q(7)
```

Signature

Initial input

- Is this satisfiable or unsatisfiable?
- If unsatisfiable, what instantiations do I need?

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

Signature

$\forall x. P(x) \lor Q(x)$

$\neg P(7) \land \neg P(2) \land \neg Q(7)$

Initial input

$\forall x. P(\mathbf{x}) \lor Q(\mathbf{x}) \Rightarrow (P(\mathbf{7}) \lor Q(\mathbf{7}))$

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

Signature

A

¬ B ∧¬ C) ∧¬(D

Initial input

A ⇒(B ∨ D)

Learned clauses

⇒ *unsatisfiable*

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
```

Signature

```
--------------------------------
∀x.¬P(x)∧∀y.Q(y)
(P(4)∨¬Q(5))∧P(6)∧Q(7)
```

Initial input

- Is this satisfiable or unsatisfiable?
- If unsatisfiable, what instantiations do I need?

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

Signature

$\forall x . \neg P(x) \wedge \forall y . Q(y)$

$(P(4) \vee \neg Q(5)) \wedge P(6) \wedge Q(7)$

Initial input

$\forall x . \neg P(\mathbf{x}) \Rightarrow \neg P(\mathbf{6})$

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

$A \wedge B$

$(C \vee \neg D) \wedge E \wedge F$

$A \Rightarrow \neg E$

$\Rightarrow$ *unsatisfiable*

Signature

Initial input

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
```

Signature

```
--------------------------------
∀x.P(x)∨Q(x)
(¬P(5)∨¬P(3))∧¬Q(5)
```

Initial input

- Is this satisfiable or unsatisfiable?
- If unsatisfiable, what instantiations do I need?

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
```

Signature

```
--------------------------------
```

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

$(\neg P(5) \vee \neg P(3)) \wedge \neg Q(5)$

Initial input

$\forall x. P(\mathbf{x}) \vee Q(\mathbf{x}) \Rightarrow (P(\mathbf{5}) \vee Q(\mathbf{5}))$

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
---------------------------------
```

Signature

A

$(\neg B \vee \neg C) \wedge \neg D$

Initial input

$A \Rightarrow (B \vee D)$

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
-------------------------------
```

Signature

A

$(\neg B \vee \neg C) \wedge \neg D$

Initial input

$A \Rightarrow (B \vee D)$

Learned clauses

⇒ *satisfiable*

A	= true	C	= false
B	= true	D	= false

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
```

Signature

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

$(\neg P(5) \vee \neg P(3)) \wedge \neg Q(5)$

Initial input

$\forall x. P(x) \vee Q(x) \Rightarrow (P(5) \vee Q(5))$

Learned clauses

Introductory Examples

$$\left.\begin{array}{l}\texttt{P : Int -> Bool}\\ \texttt{Q : Int -> Bool}\end{array}\right\}\ \text{Signature}$$

$$\left.\begin{array}{l}\forall x. P(x) \vee Q(x)\\ (\neg P(5) \vee \neg P(3)) \wedge \neg Q(5)\end{array}\right\}\ \text{Initial input}$$

$$\left.\begin{array}{l}\forall x. P(x) \vee Q(x) \Rightarrow (P(5) \vee Q(5))\\ \forall x. P(\mathbf{x}) \vee Q(\mathbf{x}) \Rightarrow (P(\mathbf{3}) \vee Q(\mathbf{3}))\end{array}\right\}\ \text{Learned clauses}$$

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

A

$(\neg B \vee \neg C) \wedge \neg D$

$A \Rightarrow (B \vee D)$

$A \Rightarrow (C \vee E)$

Signature

Initial input

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

Signature

A

$(\neg B \lor \neg C) \land \neg D$

Initial input

$A \Rightarrow (B \lor D)$

$A \Rightarrow (C \lor E)$

Learned clauses

$\Rightarrow$ *satisfiable*

A = true

B = true

C = false

D = false

E = true

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
--------------------------------
```

Signature

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

$(\neg P(5) \vee \neg P(3)) \wedge \neg Q(5)$

Initial input

$\forall x. P(x) \vee Q(x) \Rightarrow (P(5) \vee Q(5))$

$\forall x. P(x) \vee Q(x) \Rightarrow (P(3) \vee Q(3))$

...

Learned clauses

⇒ This is input is *satisfiable*, no matter how many instantiations we consider

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
a : Int
--------------------------------
∀x.P(x)∨Q(x+3)
¬P(a-3)∧¬Q(a)
```

Signature

Initial input

- Is this satisfiable or unsatisfiable?
- If unsatisfiable, what instantiations do I need?

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
a : Int
```

Signature

$\forall x. P(x) \vee Q(x+3)$

$\neg P(a-3) \wedge \neg Q(a)$

Initial input

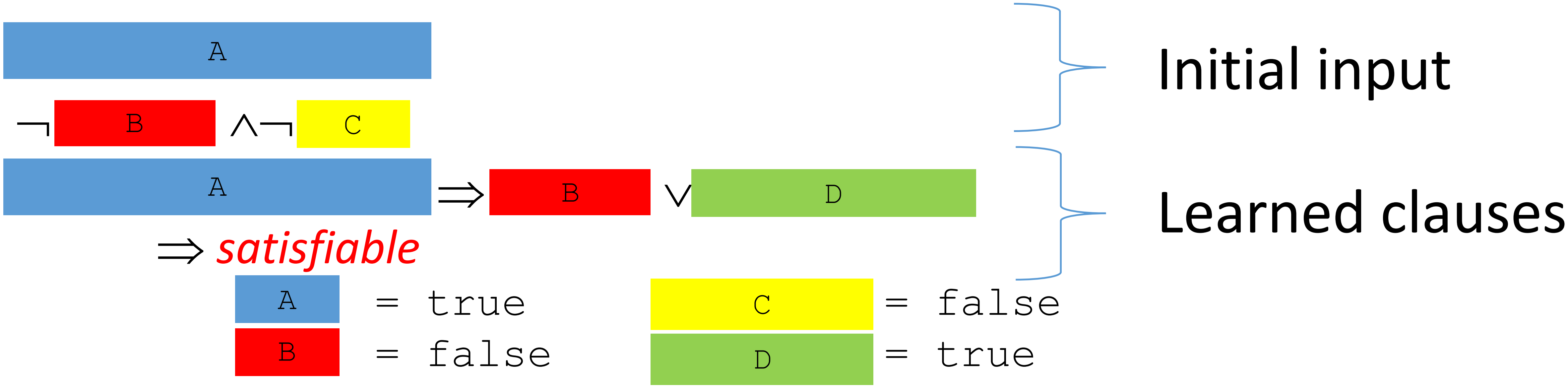
$\forall x. P(\mathbf{x}) \vee Q(\mathbf{x}+3) \Rightarrow P(\mathbf{a-3}) \vee Q((\mathbf{a-3})+3)$

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
a : Int
```

Signature



A

¬ B ∧¬ C

A ⇒ B ∨ D

⇒ *satisfiable*

A = true   C = false

B = false   D = true

Initial input

Learned clauses

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
a : Int
--------------------------------
```

Signature

A

¬B ∧¬Q(a)

Initial input

A ⇒ B ∨Q((a−3)+3)

Learned clauses

⇒ *satisfiable*

A = true   Q(a) = false

B = false   Q((a−3)+3) = true

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
a : Int
```

Signature

```
--------------------------------
```

$\forall x.P(x) \vee Q(x+3)$

$\neg P(a-3) \wedge \neg Q(a)$

Initial input

$\forall x.P(x) \vee Q(x+3) \Rightarrow P(a-3) \vee Q((a-3)+3)$

$Q((a-3)+3) \Rightarrow Q(a)$

Learned clauses

…since $(a-3)+3=a$

Introductory Examples

```
P : Int -> Bool
Q : Int -> Bool
a : Int
---------------------------------
```

Signature

A

¬ B ∧¬ C

Initial input

A ⇒ B ∨ D

D ⇒ C

Learned clauses

⇒ *unsatisfiable*

Quantified Formulas in DPLL(T): Basics

$$(P(a) \lor f(b)=a+1)$$
$$(\neg\forall x.P(x) \lor \forall y.\neg P(y) \lor R(y))$$
$$(\forall x.f(x)=g(x)+h(x) \lor \neg R(a))$$

$\Rightarrow$ Given the above input

Quantified Formulas in DPLL(T): Basics

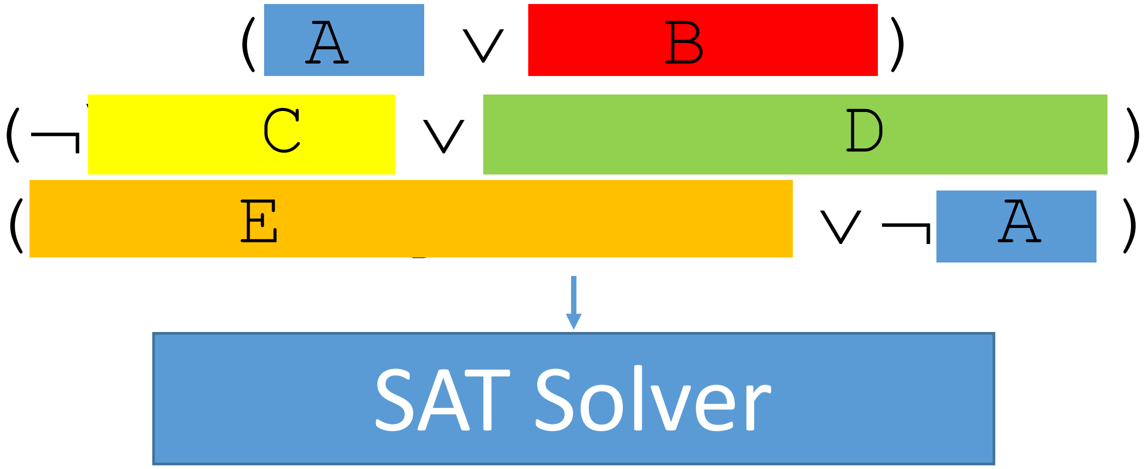
$$(P(a) \lor f(b)>a+1)$$

$$(\neg \forall x.P(x) \lor \forall y.\neg P(y) \lor R(y))$$

$$(\forall x.f(x)=g(x)+h(x) \lor \neg P(a))$$

- Consider the propositional abstraction of the formula
  - Atoms may encapsulate quantified formulas with Boolean structure
    - E.g. $\forall y.\neg P(y) \lor R(y)$

Quantified Formulas in DPLL(T): Basics



$(A \lor B)$

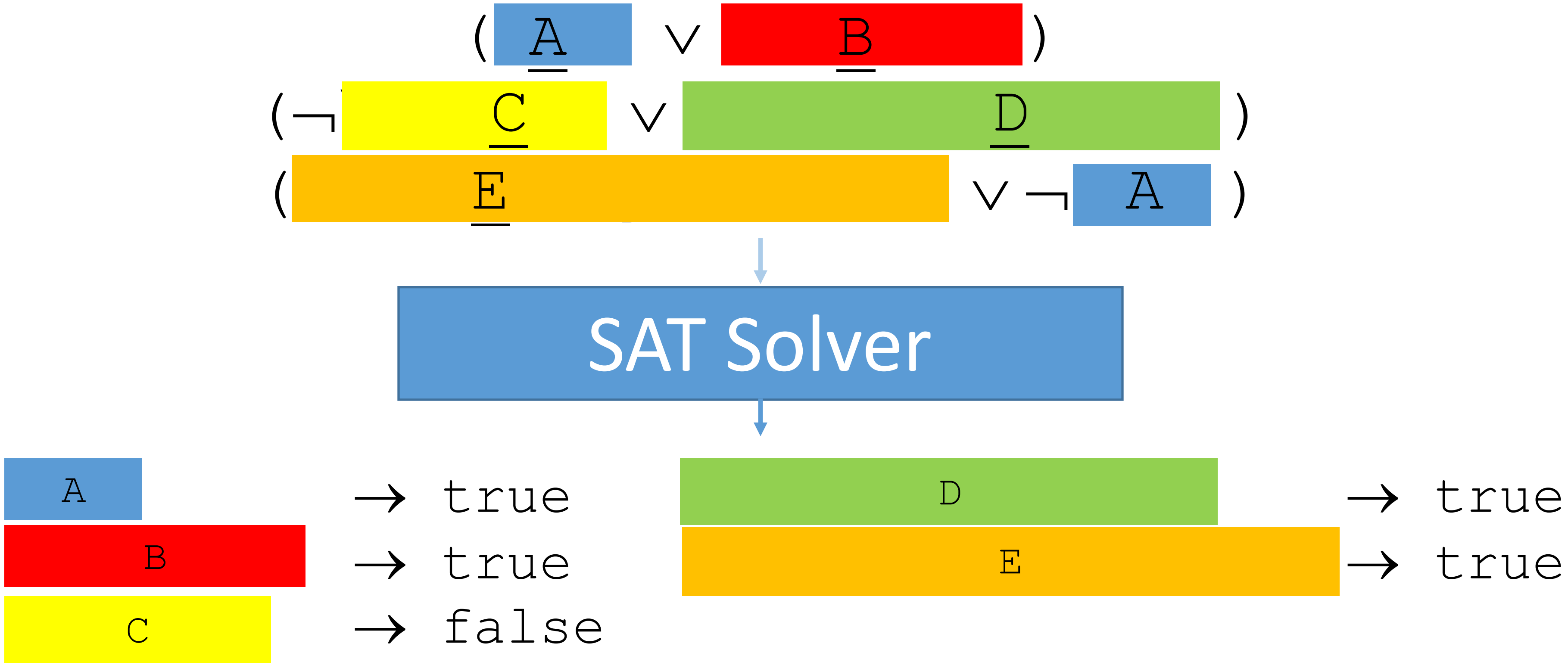
$(\neg C \lor D)$

$(E \lor \neg A)$

SAT Solver

- Find propositional satisfying assignment via off-the-shelf SAT solver

Quantified Formulas in DPLL(T): Basics



$(A \vee B)$

$(\neg C \vee D)$

$(E \vee \neg A)$

SAT Solver

A → true

B → true

C → false

D → true

E → true

- Find propositional satisfying assignment via off-the-shelf SAT solver

Quantified Formulas in DPLL(T): Basics

$$(\underline{P(a)} \lor \underline{f(b)>a+1})$$
$$(\underline{\neg\forall x.P(x)} \lor \underline{\forall y.\neg P(y) \lor R(y)})$$
$$(\underline{\forall x.f(x)=g(x)+h(x)} \lor \neg P(a))$$

↓

SAT Solver

↓

$P(a)$ → true

$f(b)>a+1$ → true

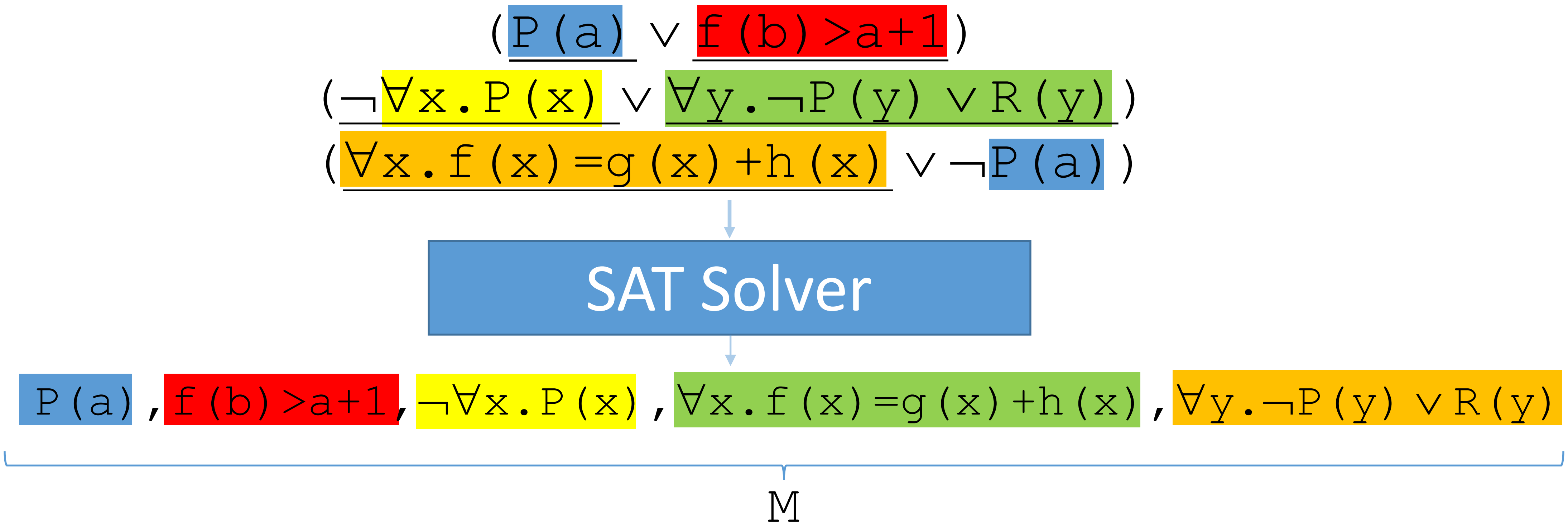
$\forall x.P(x)$ → false

$\forall y.\neg P(y) \lor R(y)$ → true

$\forall x.f(x)=g(x)+h(x)$ → true

⇒ Consider original atoms

Quantified Formulas in DPLL(T): Basics



$(P(a) \lor f(b)>a+1)$

$(\neg\forall x.P(x) \lor \forall y.\neg P(y) \lor R(y))$

$(\forall x.f(x)=g(x)+h(x) \lor \neg P(a))$

SAT Solver

$\underbrace{P(a),\ f(b)>a+1,\ \neg\forall x.P(x),\ \forall x.f(x)=g(x)+h(x),\ \forall y.\neg P(y) \lor R(y)}_{M}$

⇒ Propositional assignment can be seen as a set of T-literals M

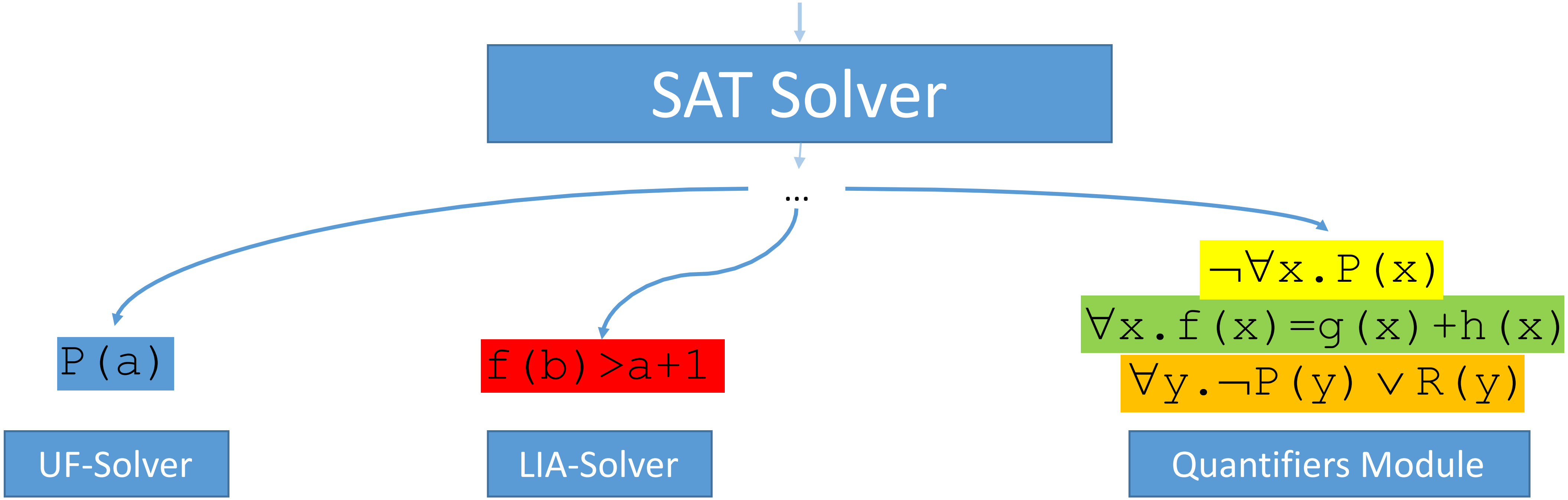
- Must check if M is T-satisfiable

Quantified Formulas in DPLL(T): Basics

$$(P(a) \lor f(b)>a+1)$$

$$(\neg\forall x.P(x) \lor \forall y.\neg P(y) \lor R(y))$$

$$(\forall x.f(x)=g(x)+h(x) \lor \neg P(a))$$



SAT Solver

…

$P(a)$

$f(b)>a+1$

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

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

$\forall y.\neg P(y) \lor R(y)$

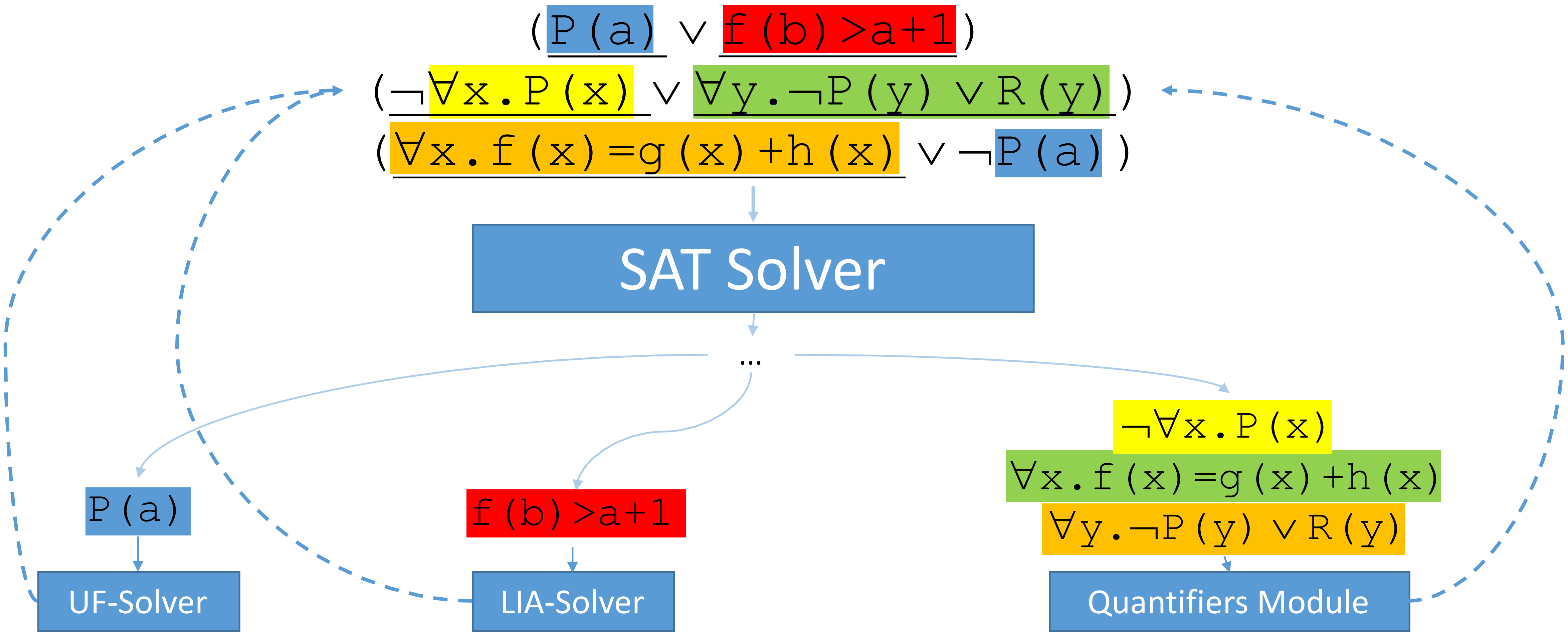
UF-Solver

LIA-Solver

Quantifiers Module

⇒ Distribute ground literals to T-solvers, ∀ literals to quantifiers module

Quantified Formulas in DPLL(T): Basics



$(P(a) \lor f(b)>a+1)$

$(\neg\forall x.P(x) \lor \forall y.\neg P(y) \lor R(y))$

$(\forall x.f(x)=g(x)+h(x) \lor \neg P(a))$

SAT Solver

…

$P(a)$

$f(b)>a+1$

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

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

$\forall y.\neg P(y) \lor R(y)$

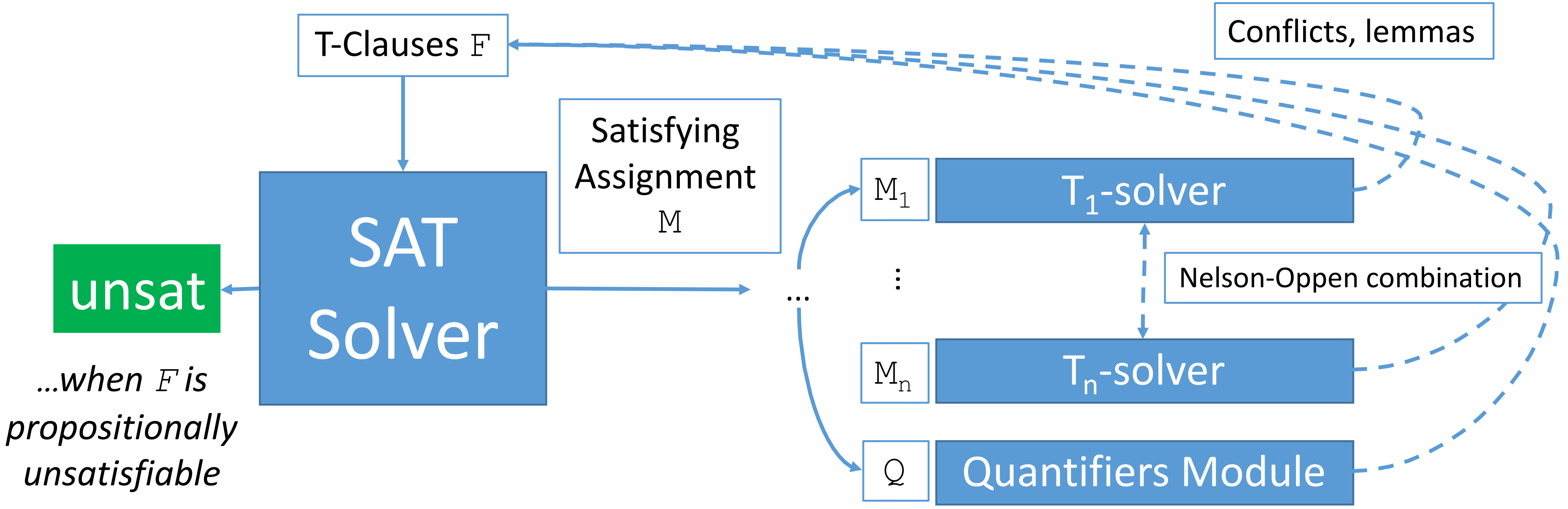
UF-Solver

LIA-Solver

Quantifiers Module

⇒ These solvers may choose to add conflicts/lemmas to clause set

DPLL(T_1+..+T_n)+Quantifiers: Overview

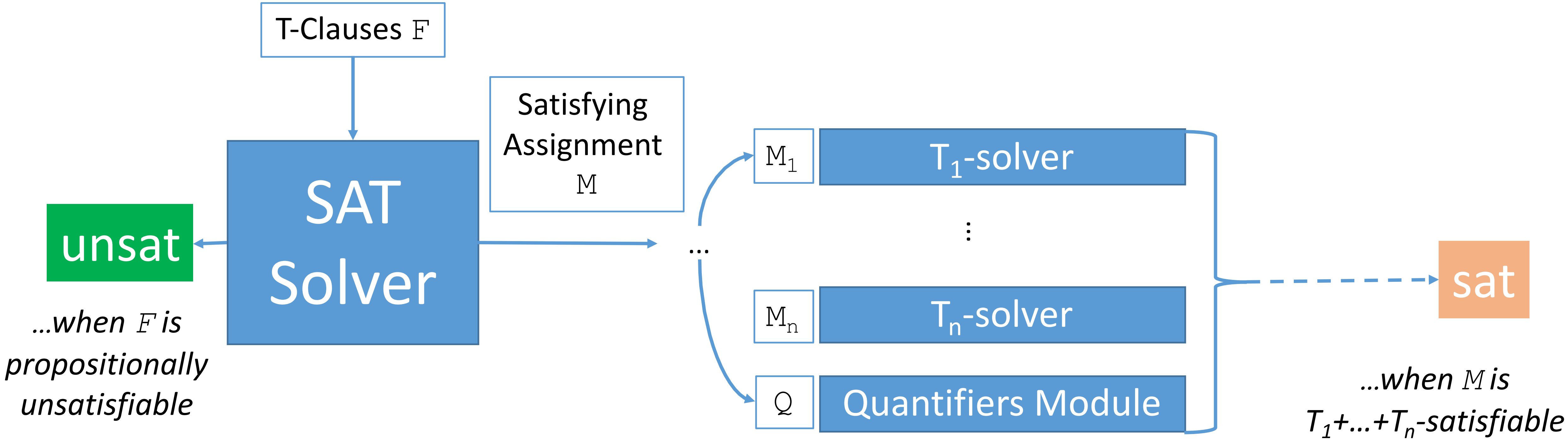


⇒ Each of these components may:

- Report M is T-unsatisfiable by reporting conflict clauses
- Report lemmas if they are unsure

[Nieuwenhuis/Oliveras/Tinelli 06]

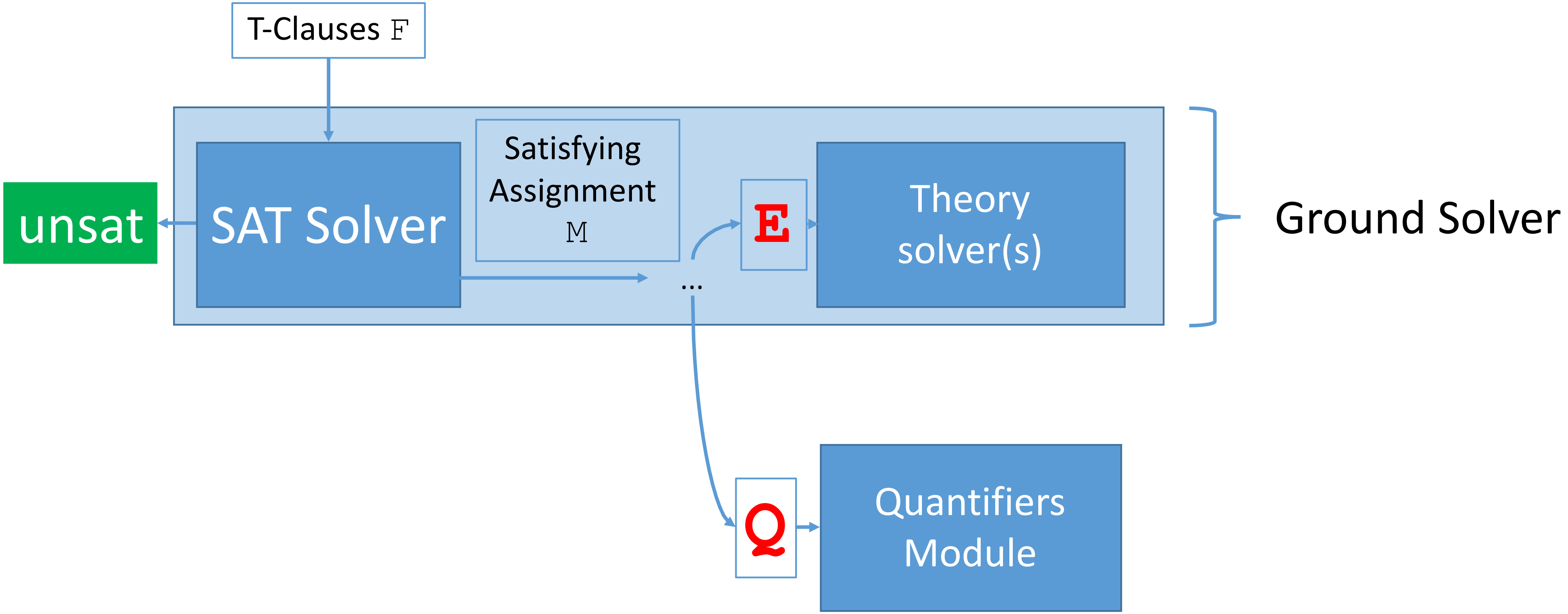
DPLL(T_1+..+T_n)+Quantifiers: Overview



⇒ If no component adds a lemma, then it must be the case that $\mathtt{M}$ is T_1+…+T_n-satisfiable

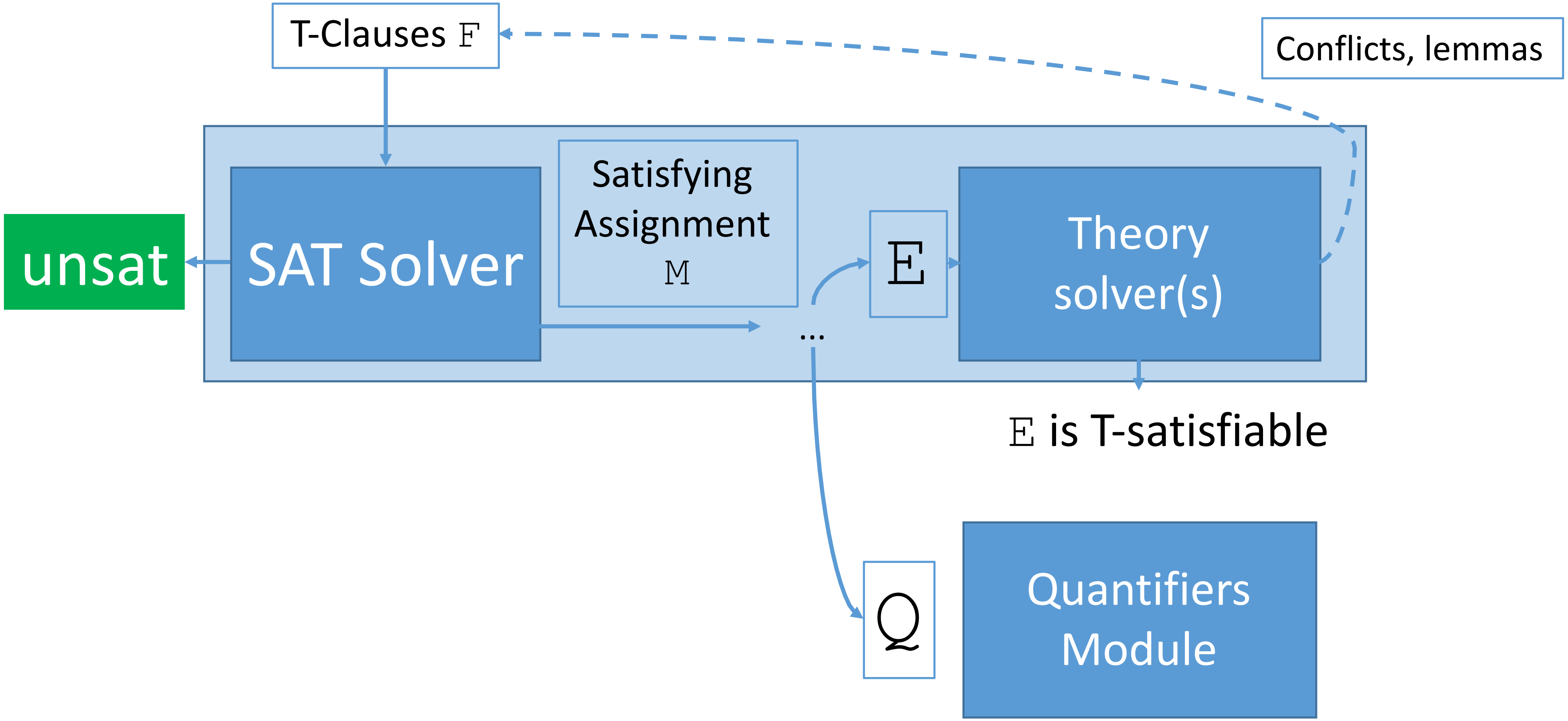
[Nieuwenhuis/Oliveras/Tinelli 06]

In this talk: DPLL(T)+Quantifiers, simplified



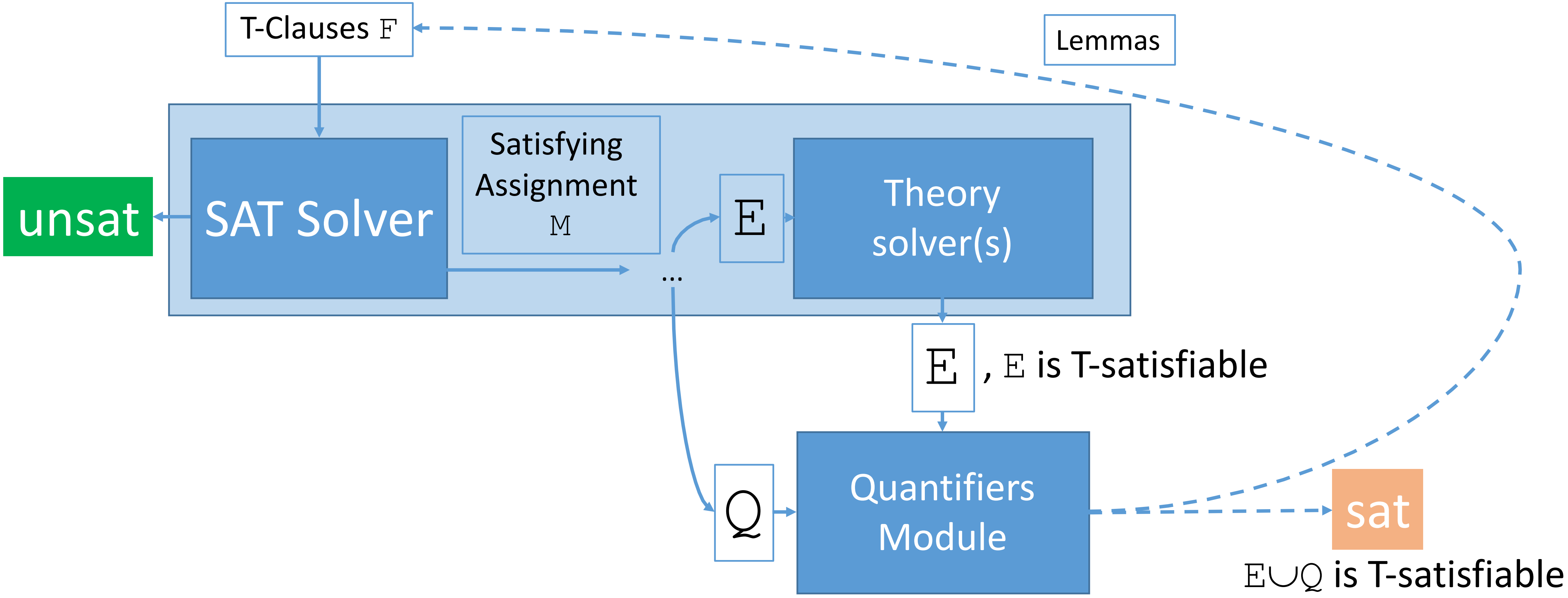
⇒ For purposes of this talk, partition M into quantifier-free part **E**, and set of ∀ formulas **Q**

In this talk: DPLL(T)+Quantifiers, simplified



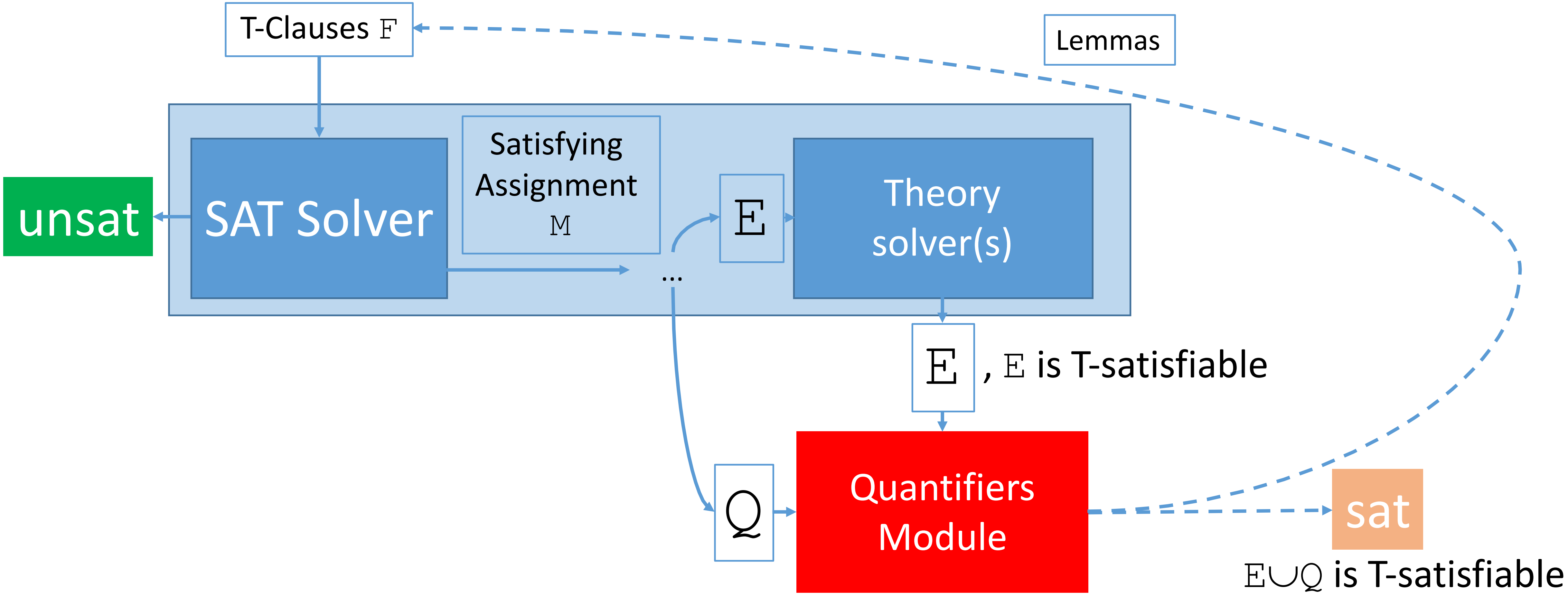
⇒ Theory solvers determine whether E is T-(un)satisfiable

In this talk: DPLL(T)+Quantifiers, simplified



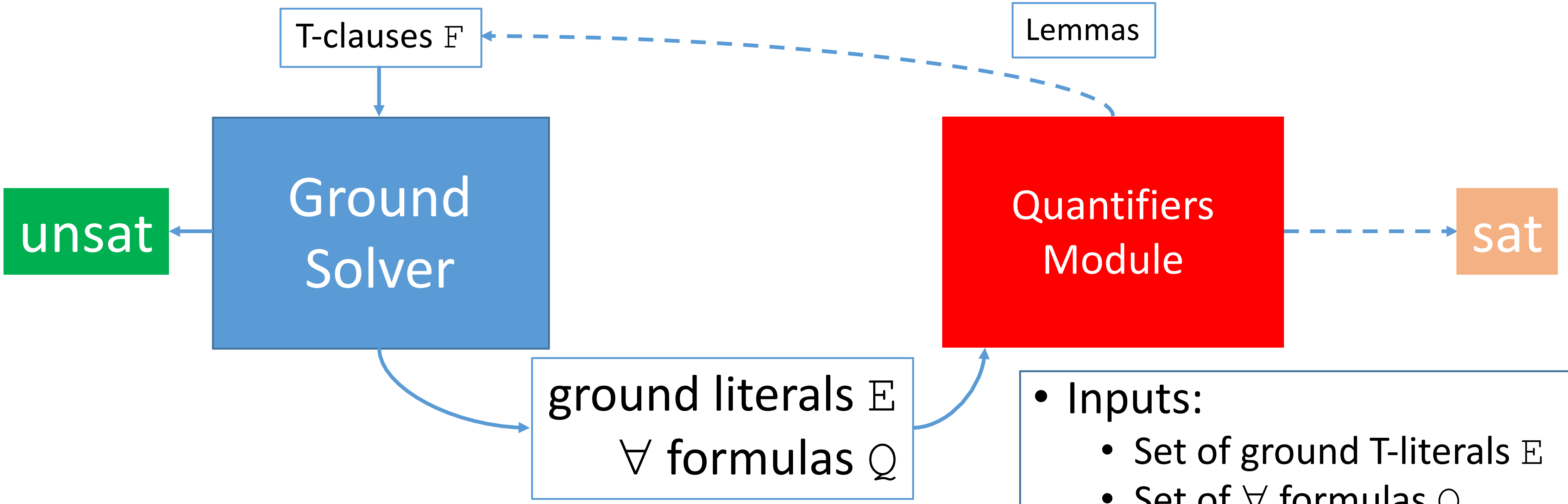
⇒ If E is T-satisfiable, quantifiers module may be invoked

In this talk: DPLL(T)+Quantifiers, simplified



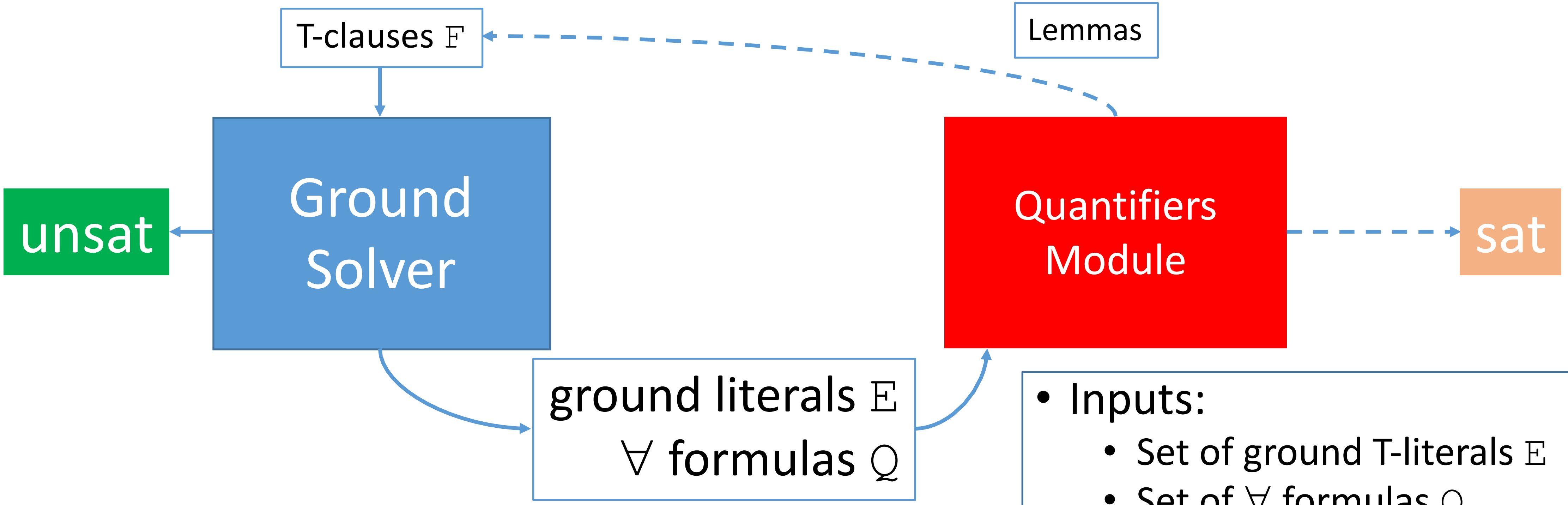
⇒ *The remainder of the talk will discuss how the quantifiers module is implemented*

DPLL(T)+Quantifiers, further simplified



- Inputs:
  - Set of ground T-literals $\mathtt{E}$
  - Set of $\forall$ formulas $\mathtt{Q}$
- Outputs:
  - "$\mathtt{E} \cup \mathtt{Q}$ is T-satisfiable", or
    $\Rightarrow$ $\mathtt{F}$ is T-satisfiable
  - Set of lemmas to add to $\mathtt{F}$

DPLL(T)+Quantifiers, further simplified



T-clauses F

Lemmas

unsat

Ground Solver

Quantifiers Module

sat

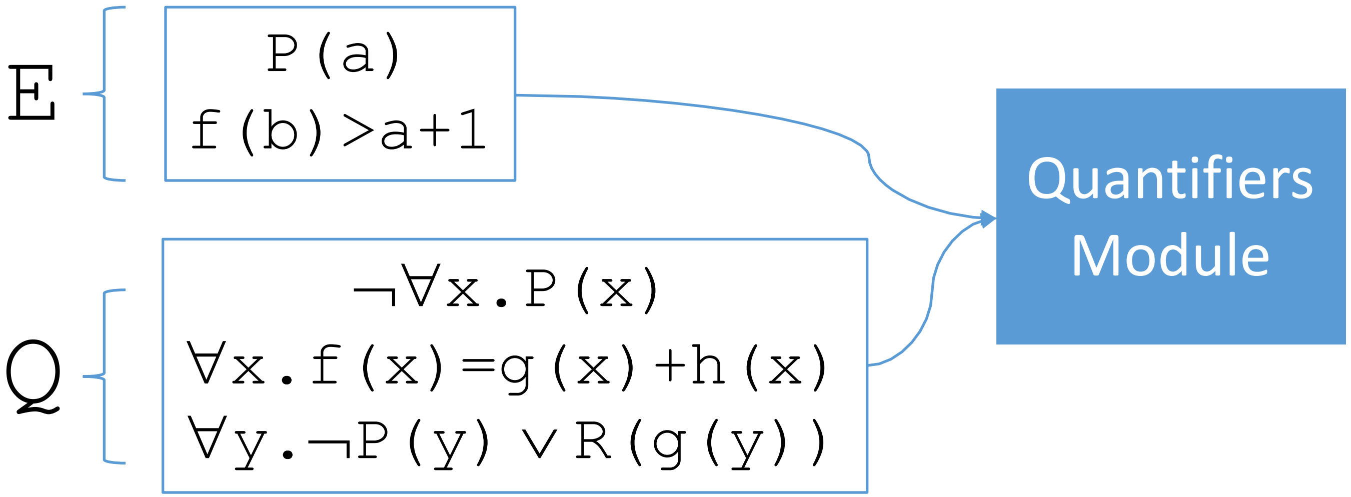
ground literals E

$\forall$ formulas Q

- Inputs:
  - Set of ground T-literals E
  - Set of $\forall$ formulas Q
- Outputs:
  - "$E \cup Q$ is T-satisfiable", or
    $\Rightarrow$ F is T-satisfiable
  - Set of lemmas to add to F

- Recurrent Questions:
  - Which lemmas do we add?
  - How do we know $E \cup Q$ is T-satisfiable?
  - When do we invoke it?

Which lemmas do we add: Basics



E

$P(a)$

$f(b)>a+1$

Q

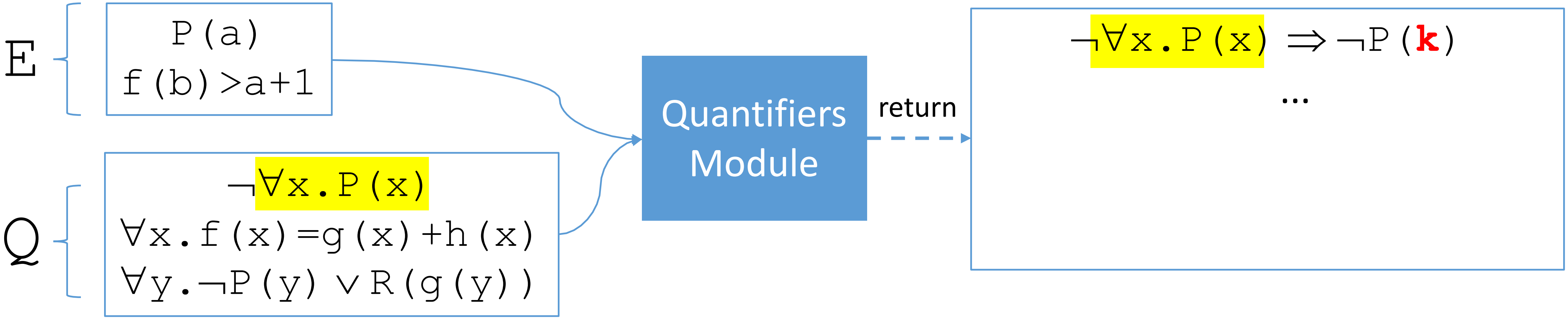
$\neg\forall x.P(x)$

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

$\forall y.\neg P(y) \vee R(g(y))$

Quantifiers Module

Which lemmas do we add: Basics



E

$P(a)$

$f(b)>a+1$

Q

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

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

$\forall y.\neg P(y) \vee R(g(y))$

Quantifiers Module

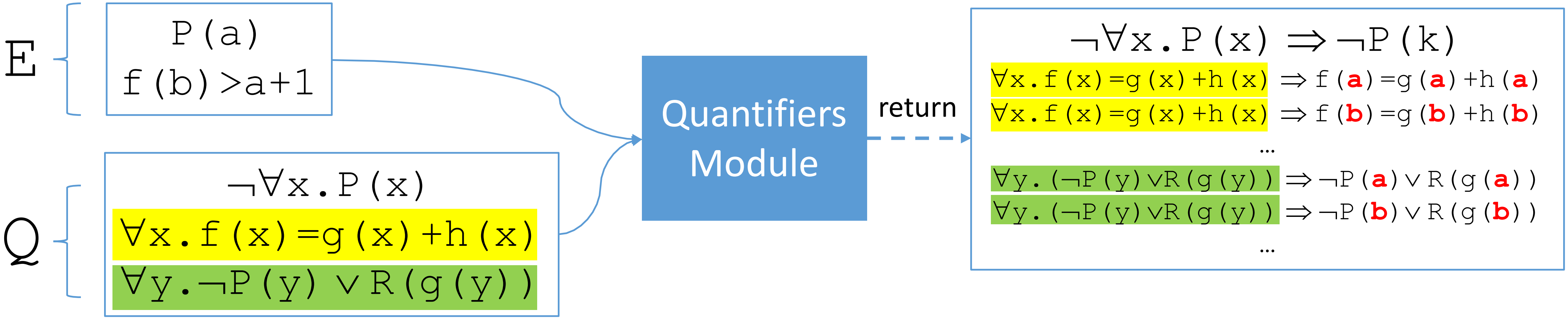
return

$\neg\forall x.P(x) \Rightarrow \neg P(\mathbf{k})$

…

- Existential quantification (negated universals) handled by Skolemization
  - Introduce a fresh witness **k**, lemma says $\exists x.\neg P(x)$ implies $\neg P(\mathbf{k})$
  - Need only be applied once

Which lemmas do we add: Basics



E

$P(a)$

$f(b)>a+1$

Q

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

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

$\forall y.\neg P(y) \vee R(g(y))$

Quantifiers Module

return

$\neg\forall x.P(x) \Rightarrow \neg P(k)$

$\forall x.f(x)=g(x)+h(x) \Rightarrow f(\mathbf{a})=g(\mathbf{a})+h(\mathbf{a})$

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

…

$\forall y.(\neg P(y) \vee R(g(y))) \Rightarrow \neg P(\mathbf{a}) \vee R(g(\mathbf{a}))$

$\forall y.(\neg P(y) \vee R(g(y))) \Rightarrow \neg P(\mathbf{b}) \vee R(g(\mathbf{b}))$

…

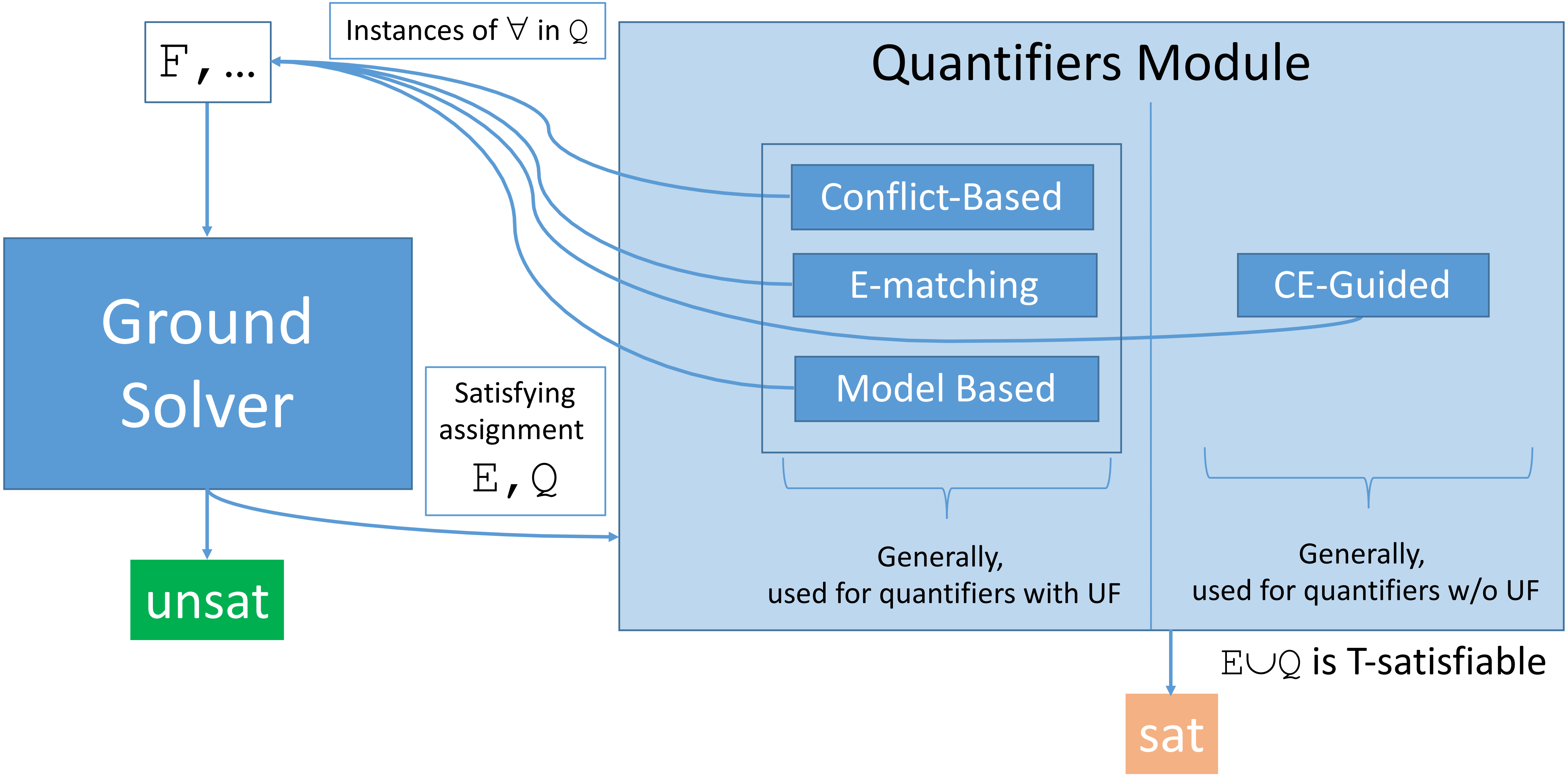
- Universal quantification handled by Instantiation
  - Choose ground term(s) **t**, lemma(s) say $\forall x.f(x)=g(x)+h(x)$ implies $f(\mathbf{t})=g(\mathbf{t})+h(\mathbf{t})$

$\Rightarrow$ May be applied ad infinitum!

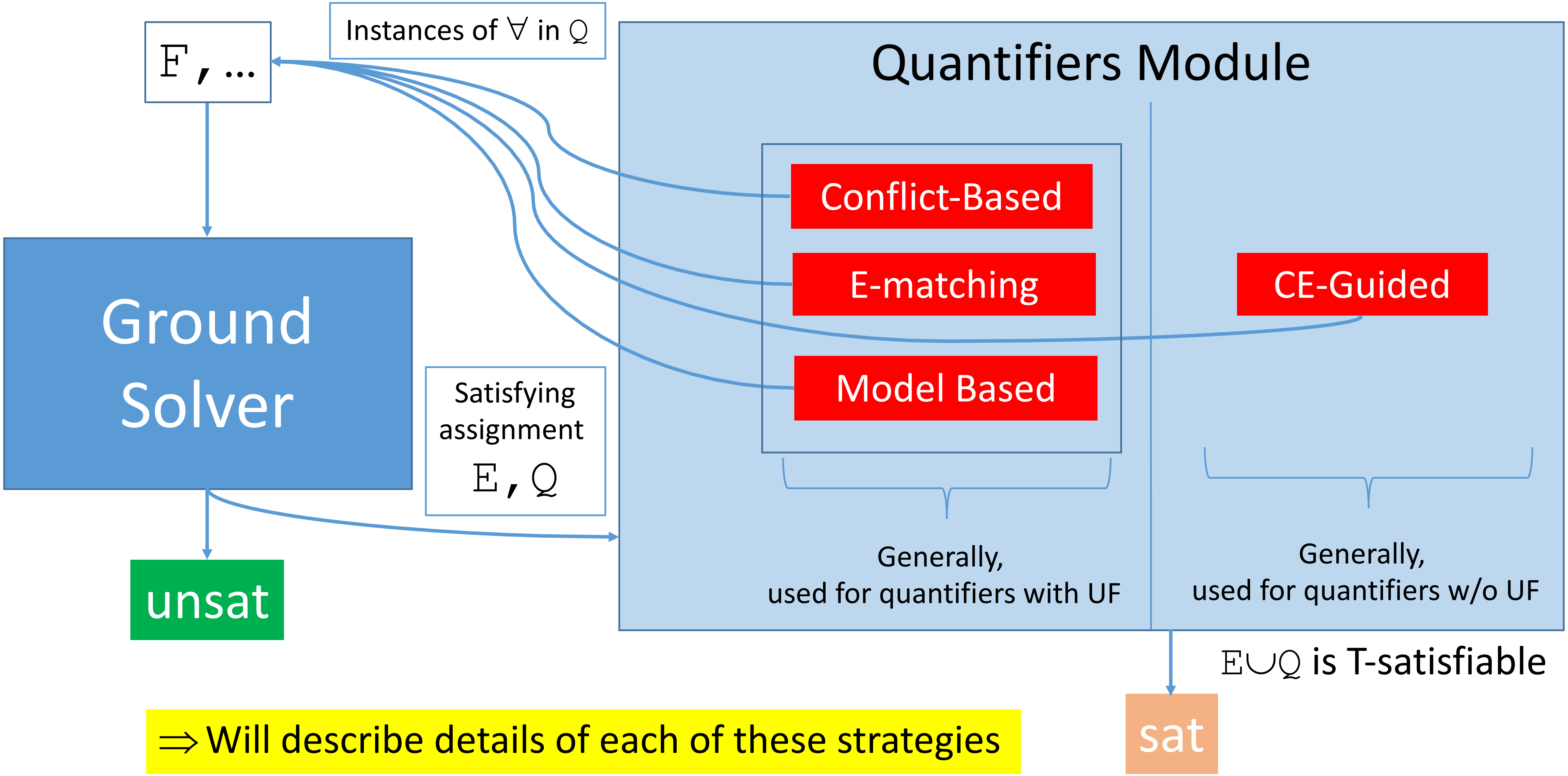
Quantifiers Module : Recurrent Questions

- Which *instances* do we add?
  - E-matching **[Detlefs et al 03]**
  - Conflict-based quantifier instantiation **[Reynolds et al FMCAD14]**
  - Model-based quantifier instantiation **[Ge,de Moura CAV09]**
  - Counterexample-guided quantifier instantiation **[Reynolds et al CAV15]**

Techniques for Quantifier Instantiation: Overview

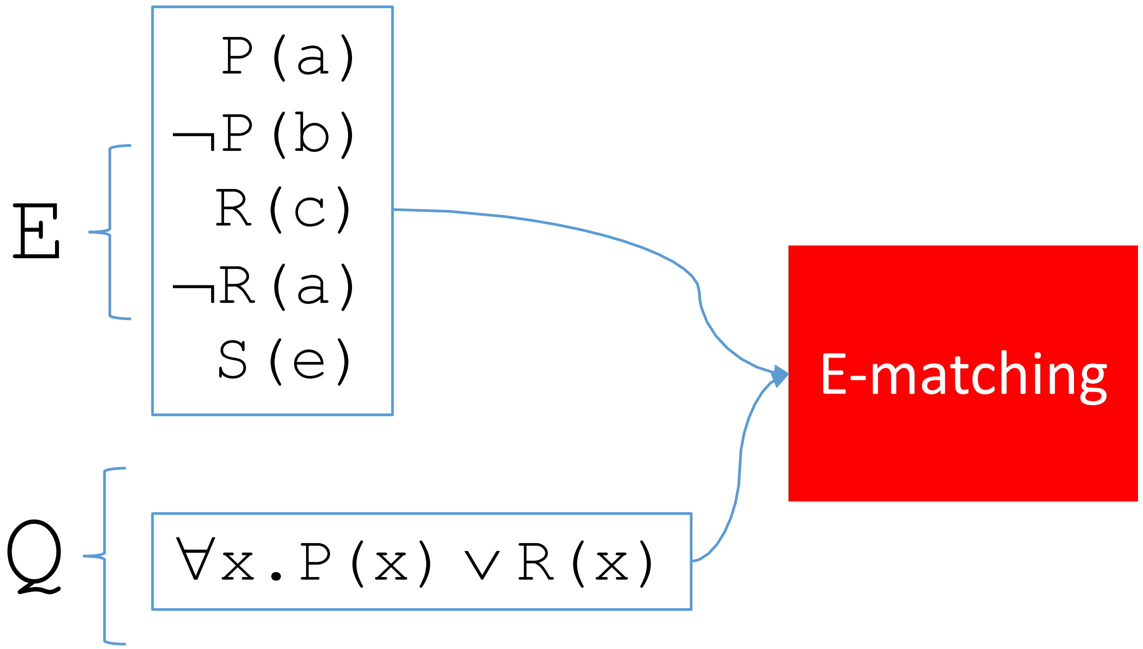


Techniques for Quantifier Instantiation: Overview



⇒ Will describe details of each of these strategies

E-matching

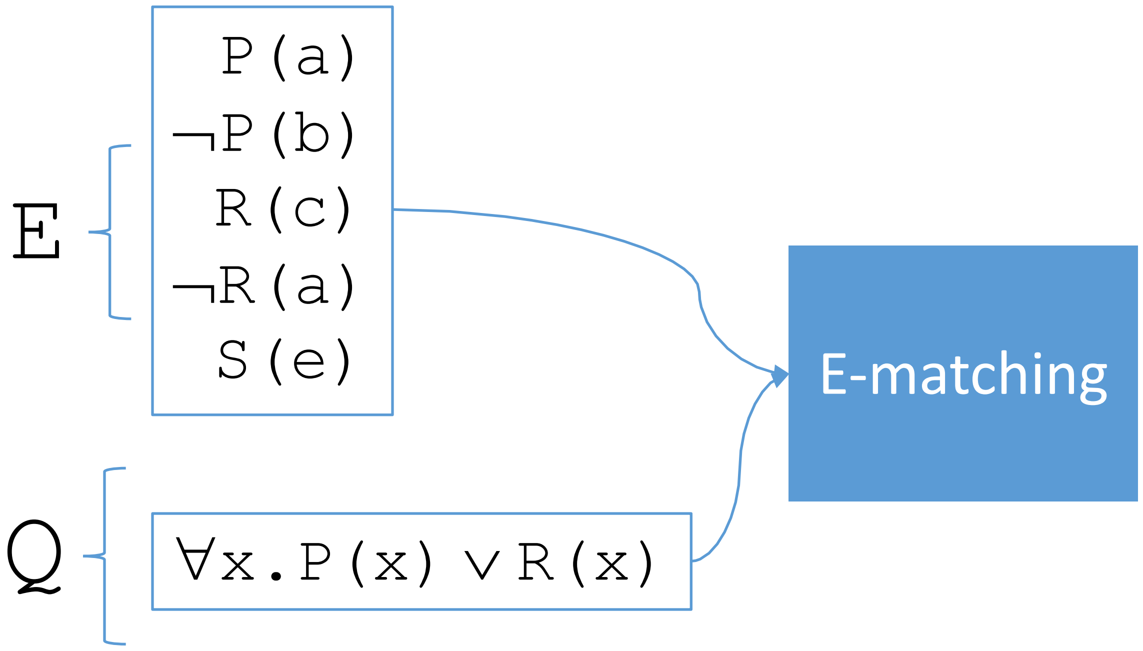


Conflict-Based

E-matching

Model Based

E-matching



E

$P(a)$

$\neg P(b)$

$R(c)$

$\neg R(a)$

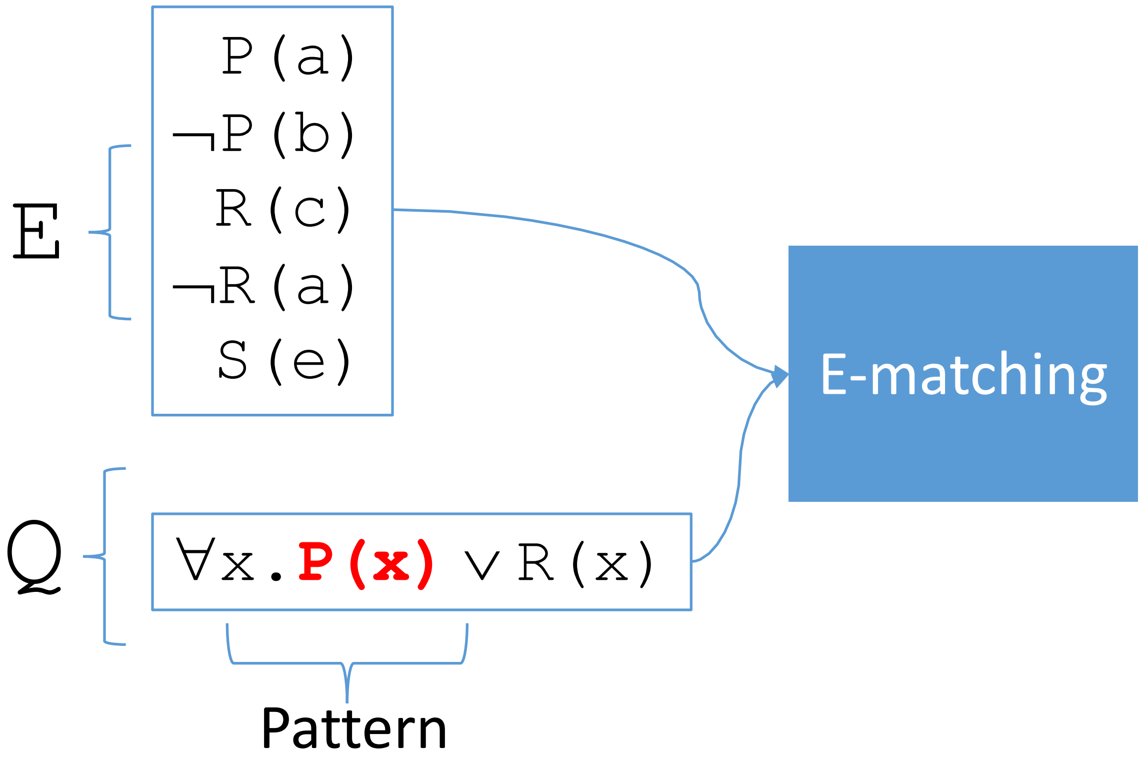
$S(e)$

Q

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

E-matching

E-matching



E

$P(a)$

$\neg P(b)$

$R(c)$

$\neg R(a)$

$S(e)$

Q

$\forall x. \underbrace{\mathbf{P(x)}}_{\text{Pattern}} \lor R(x)$

E-matching

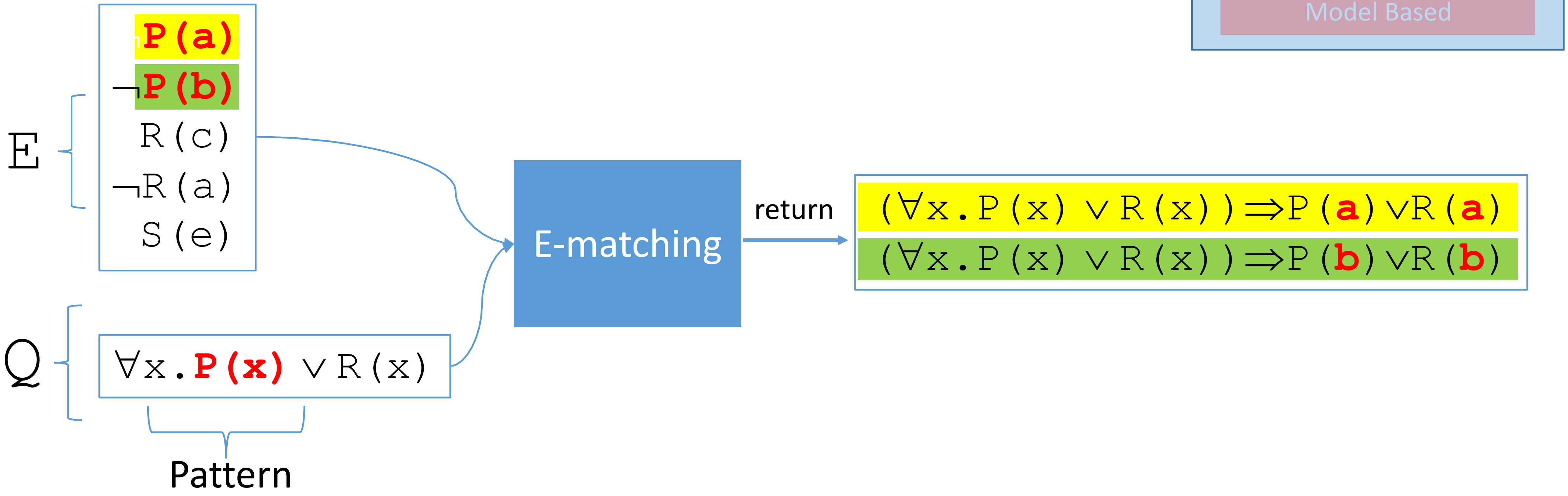
⇒ **Idea**: choose instances based on pattern matching

Conflict-Based

E-matching

Model Based

E-matching



E

$P(a)$

$\neg P(b)$

$R(c)$

$\neg R(a)$

$S(e)$

Q

$\forall x.\underbrace{P(x)}_{\text{Pattern}} \vee R(x)$

E-matching

return

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

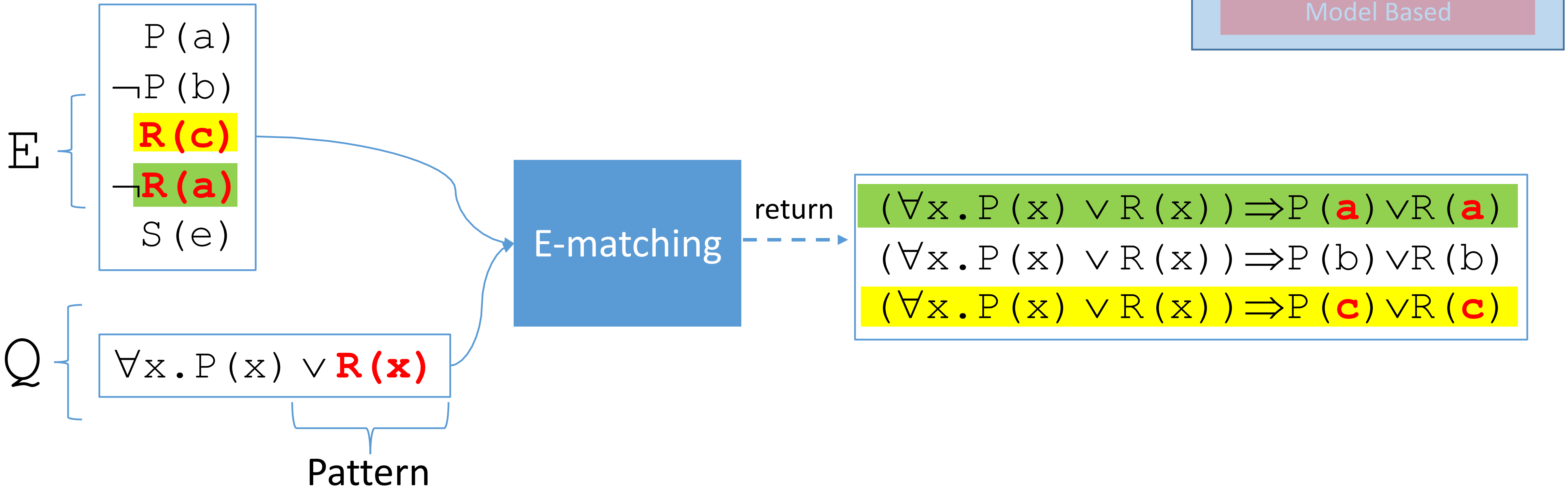
$(\forall x.P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$

E-matching

Conflict-Based

E-matching

Model Based



E

$P(a)$

$\neg P(b)$

$\mathbf{R(c)}$

$\neg \mathbf{R(a)}$

$S(e)$

Q

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

Pattern

E-matching

return

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

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

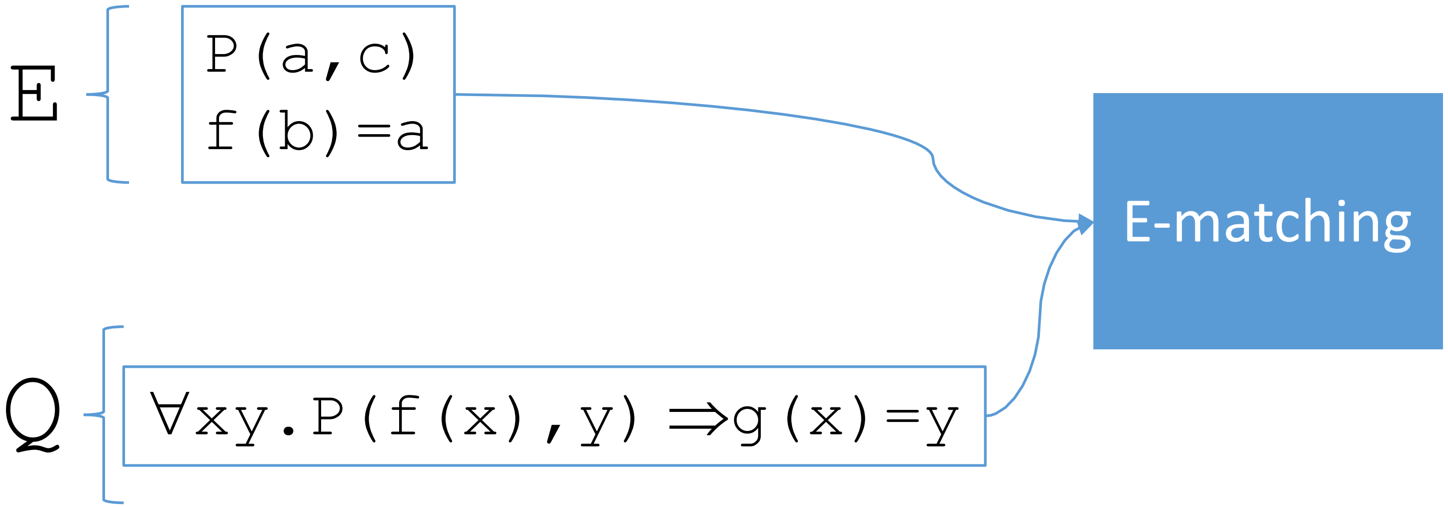
$(\forall x. P(x) \vee R(x)) \Rightarrow P(\mathbf{c}) \vee R(\mathbf{c})$

E-matching: Functions, Equality

Conflict-Based

E-matching

Model Based



E

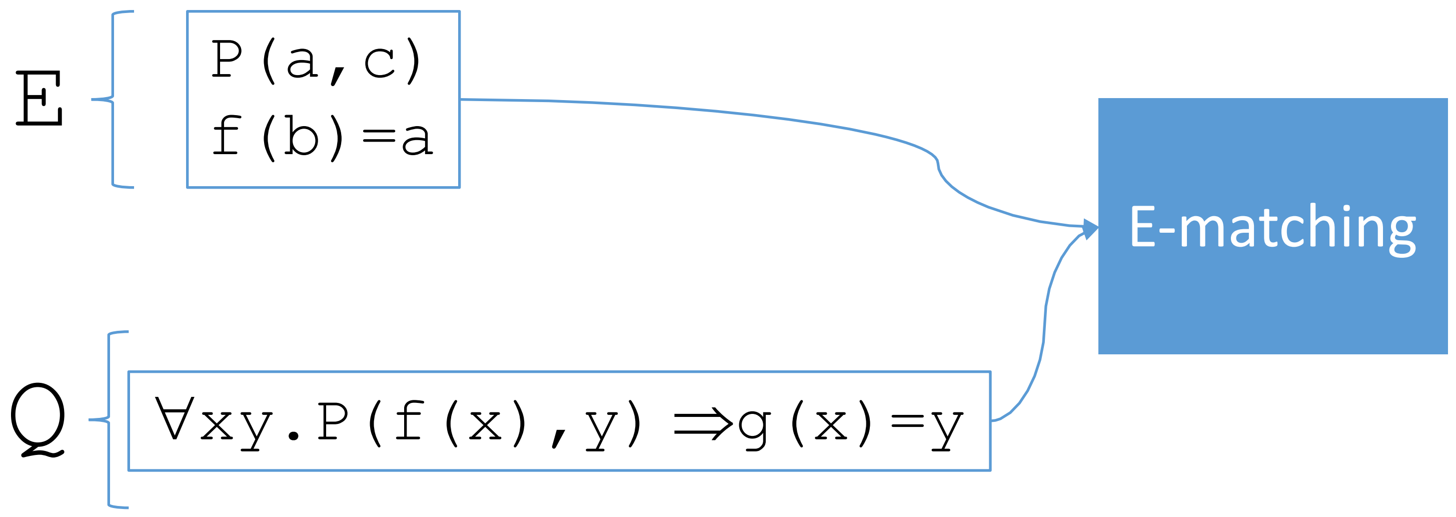
P(a,c)
f(b)=a

Q

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

E-matching

E-matching: Functions, Equality



E

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

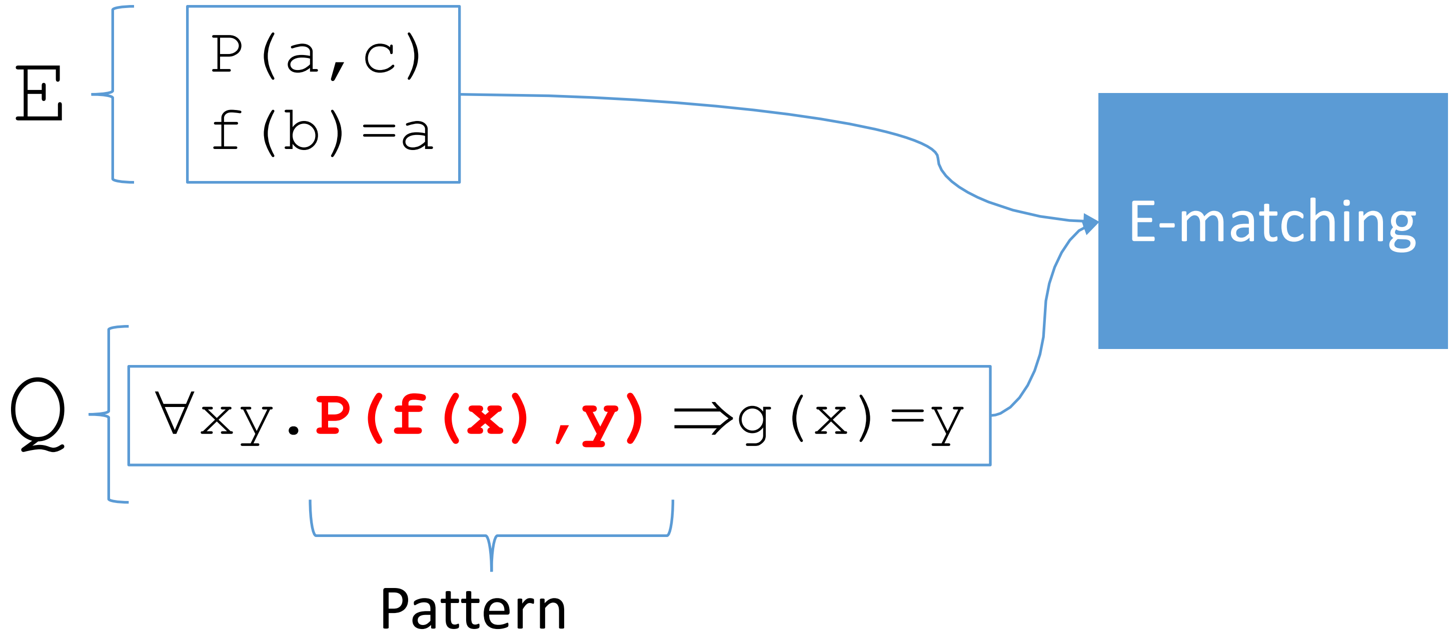
Q

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

E-matching

$\Rightarrow$ In E-matching, Pattern ***matching*** takes into account equalities in ***E***

E-matching: Functions, Equality



E

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

Q

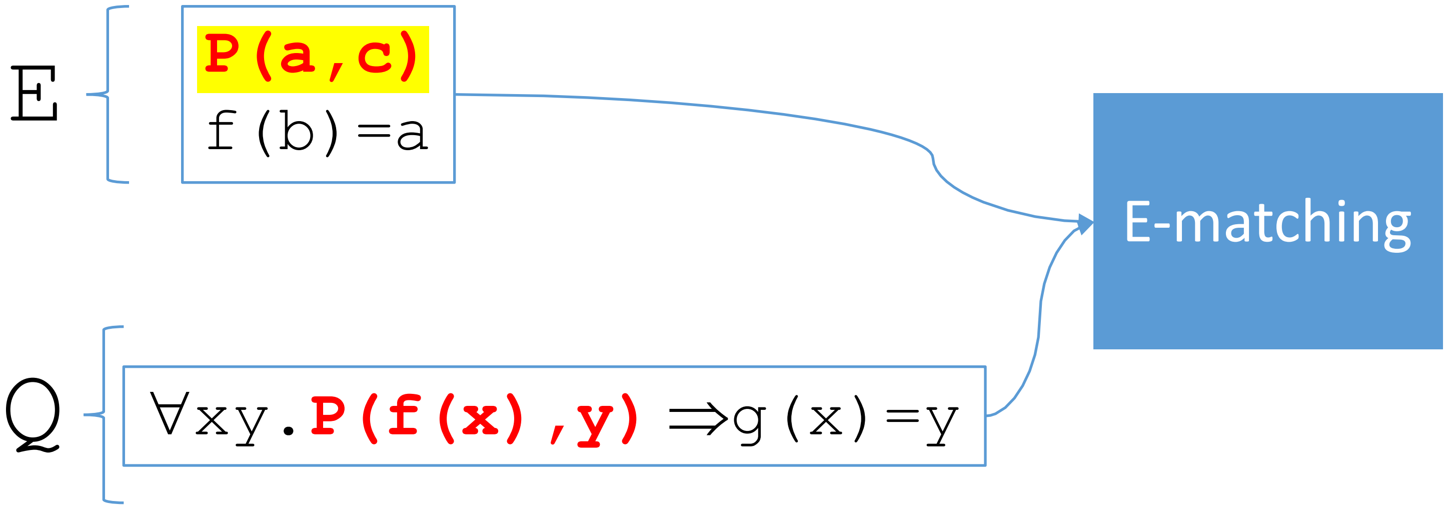
$\forall xy.\underbrace{\mathbf{P(f(x),y)}}_{\text{Pattern}} \Rightarrow g(x)=y$

Conflict-Based

E-matching

Model Based

E-matching: Functions, Equality



E

P(a,c)

f(b)=a

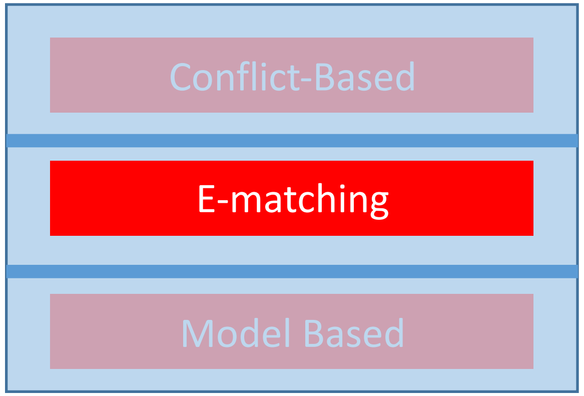
Q

∀xy.P(f(x),y) ⇒g(x)=y

E-matching

P(a ,c)

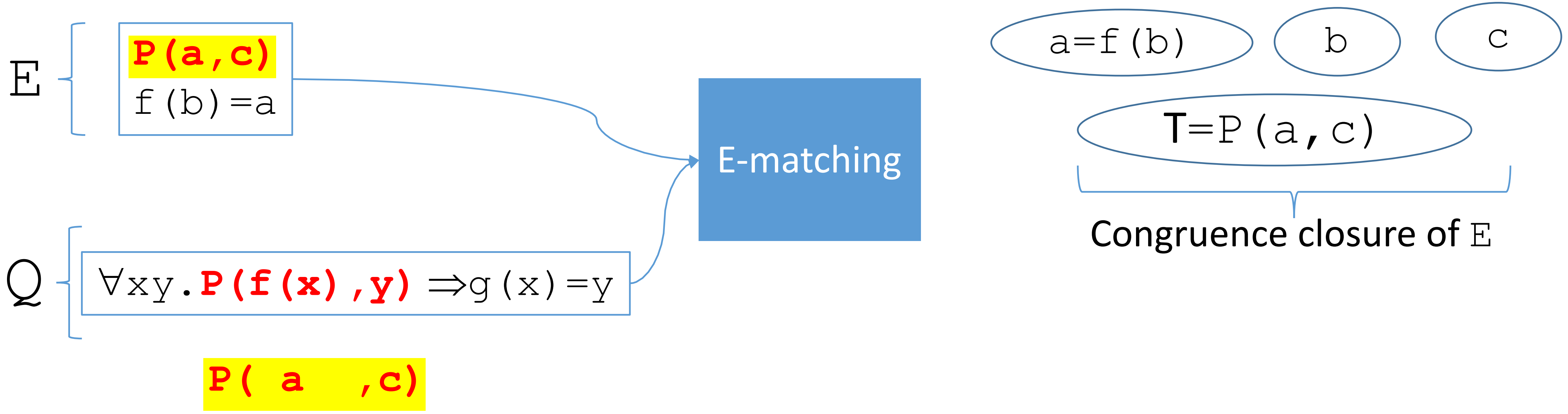
E-matching: Functions, Equality



Conflict-Based

E-matching

Model Based



E

P(a,c)

f(b)=a

Q

∀xy.P(f(x),y) ⇒g(x)=y

P(a ,c)

E-matching

a=f(b)

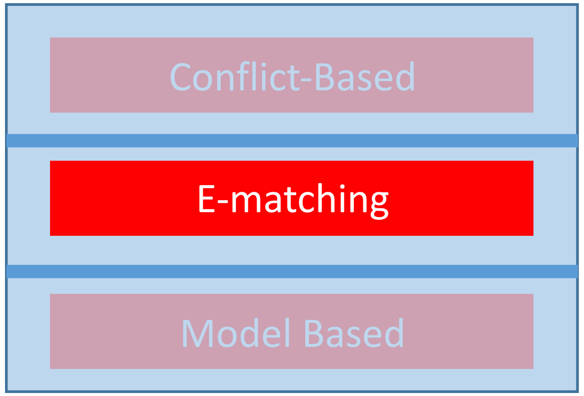
b

c

T=P(a,c)

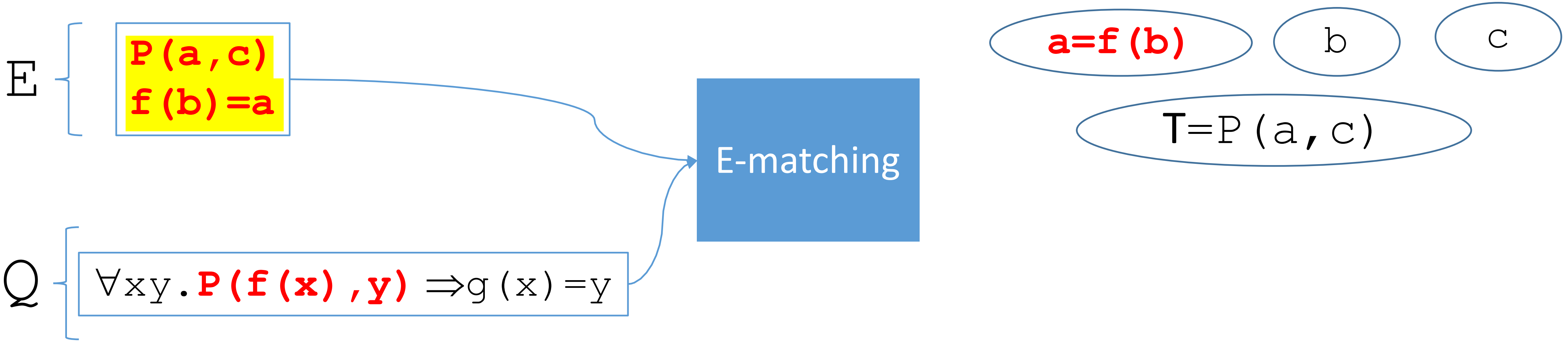
Congruence closure of E

E-matching: Functions, Equality


Conflict-Based

E-matching

Model Based


E

P(a,c)

f(b)=a

Q

∀xy.P(f(x),y) ⇒g(x)=y

E-matching

a=f(b)

b

c

T=P(a,c)

P(f(b),c)

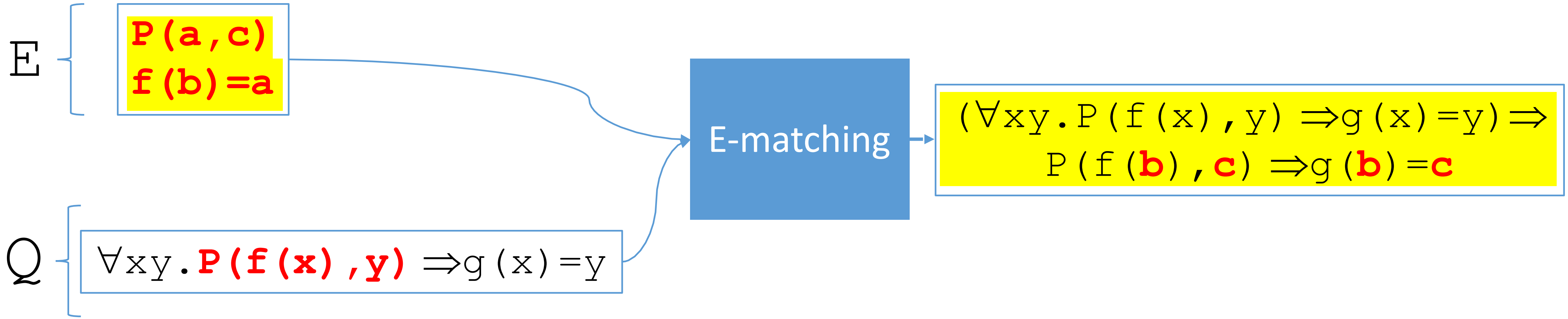
…E implies P(a,c)⇔P(f(b),c)

E-matching: Functions, Equality

Conflict-Based

E-matching

Model Based



E

P(a,c)

f(b)=a

Q

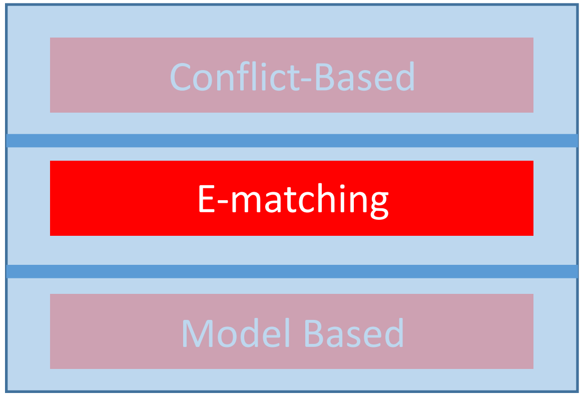
$\forall xy.P(f(x),y) \Rightarrow g(x)=y$

E-matching

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

P(f(b),c)

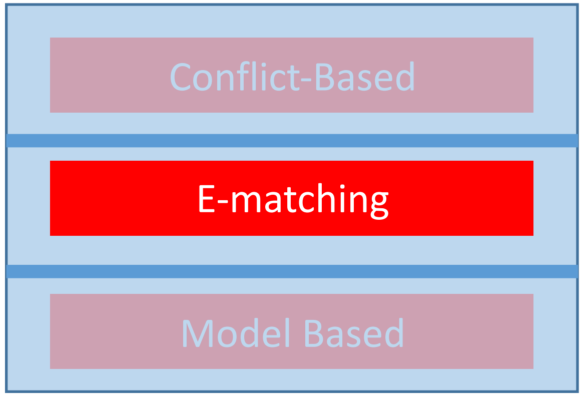
E-matching: Challenges



- E-matching has no standard way of selecting patterns
- E-matching generates too many instances
  - Many instances may overload the ground solver
- E-matching is incomplete
  - It may be non-terminating
  - When it terminates, we generally cannot answer “$E \cup Q$ is T-satisfiable”
    - Although for some fragments+variants, we may guarantee (termination ⇔ model)
      - Decision Procedures via Triggers **[Dross et al 13]**
      - Local Theory Extensions **[Bansal et al 15]**

      ⇒ Typically are established by a separate pencil-and-paper proof

E-matching: Pattern Selection

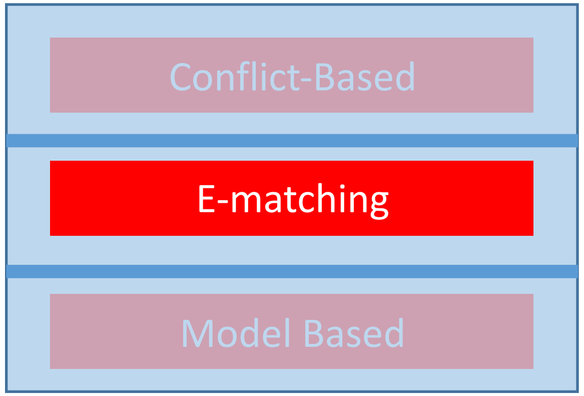


Conflict-Based

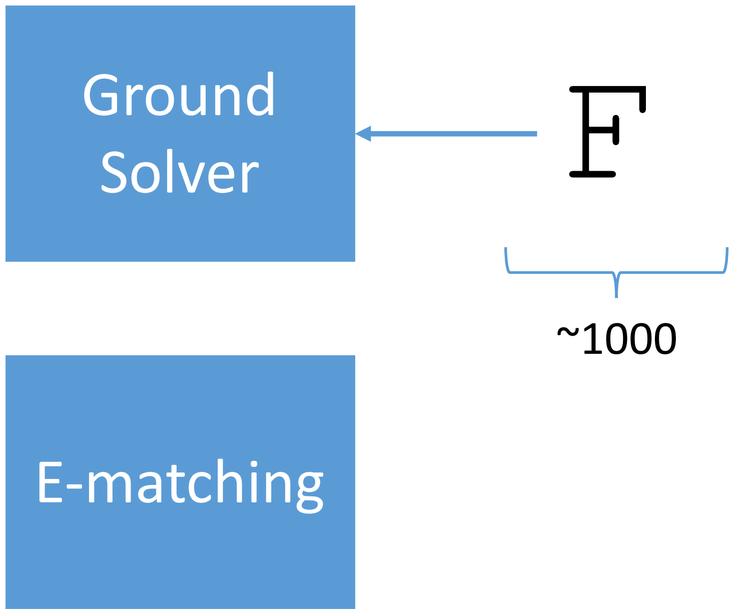
E-matching

Model Based

- In practice, pattern selection can is done either by:
  - The user, via annotations, e.g. `(! … :pattern ((P x)))`
  - The SMT solver itself
- Recurrent questions:
  - Which terms be we permit as patterns? Typically, applications of UF:
    - Use $f(x,y)$ but not $x+y$ for $\forall xy.f(x,y)=x+y$
  - What if multiple patterns exist? Typically use all available patterns:
    - Use both $P(x)$ and $R(x)$ for $\forall x.P(x)\vee R(x)$
  - What if no appropriate term contains all variables? May use “multi-patterns”:
    - $\{R(x,y),R(y,z)\}$ for $\forall xyz.(R(x,y)\wedge R(y,z))\Rightarrow R(x,z)$
- Pattern selections may impact performance significantly **[Leino et al 16]**

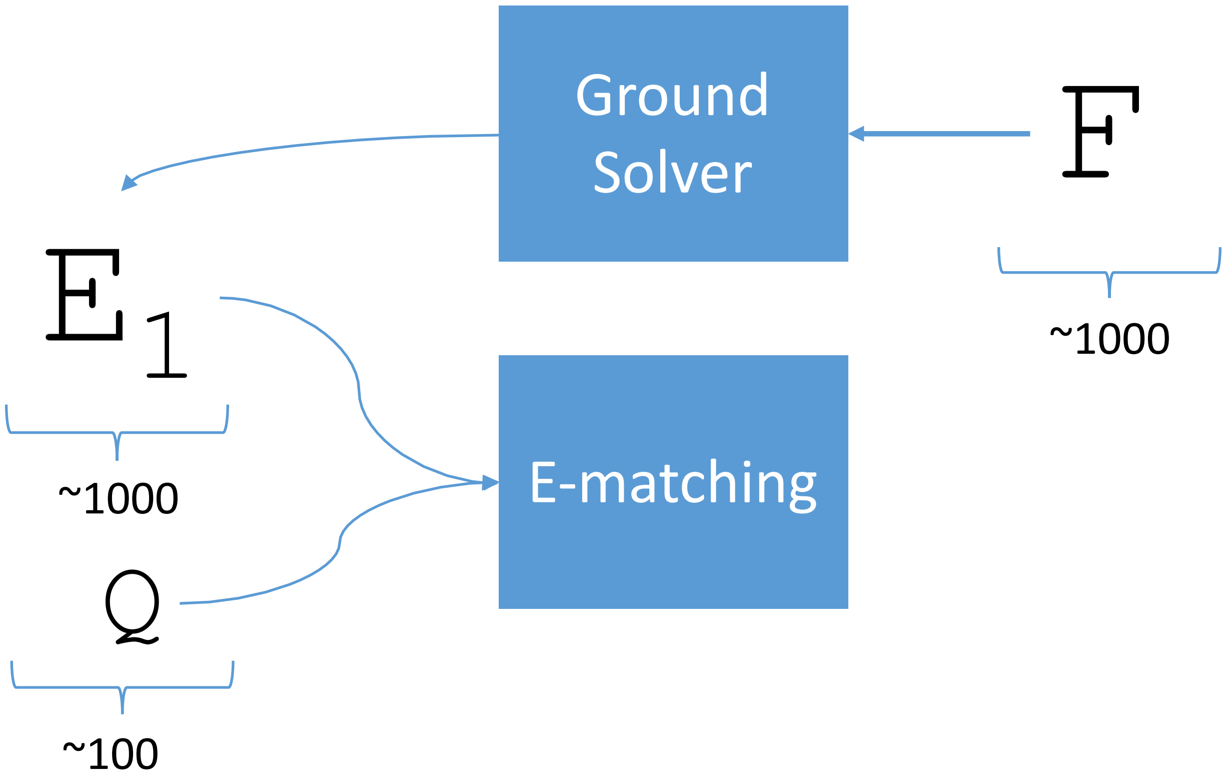


E-matching: Too Many Instances



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

E-matching: Too Many Instances



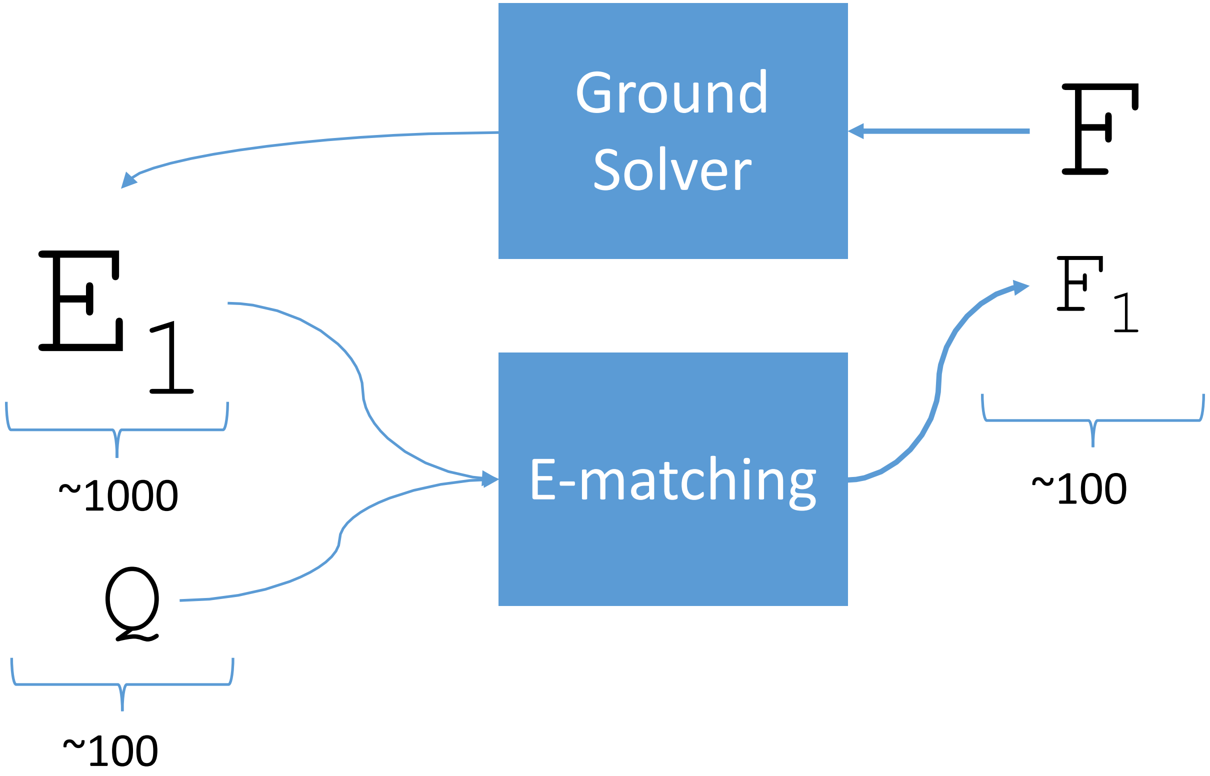
- Typical problems in applications:
  - $\mathtt{F}$ contains 1000s of clauses
  - Satisfying assignments contain 1000s of terms in $\mathtt{E}$, 100s of $\forall$ in $\mathtt{Q}$

Conflict-Based

E-matching

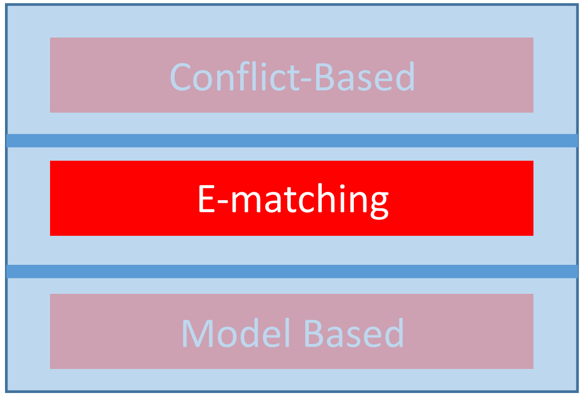
Model Based

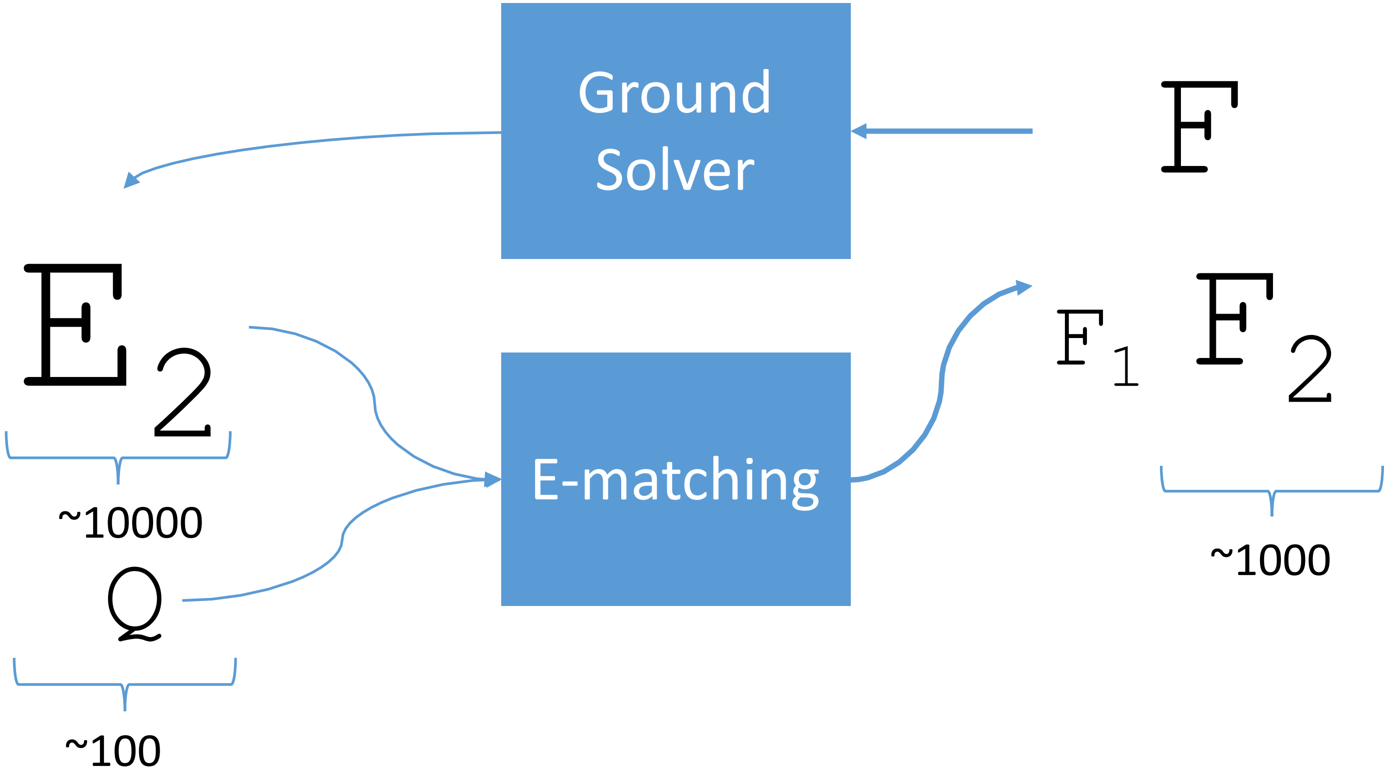
E-matching: Too Many Instances



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

E-matching: Too Many Instances





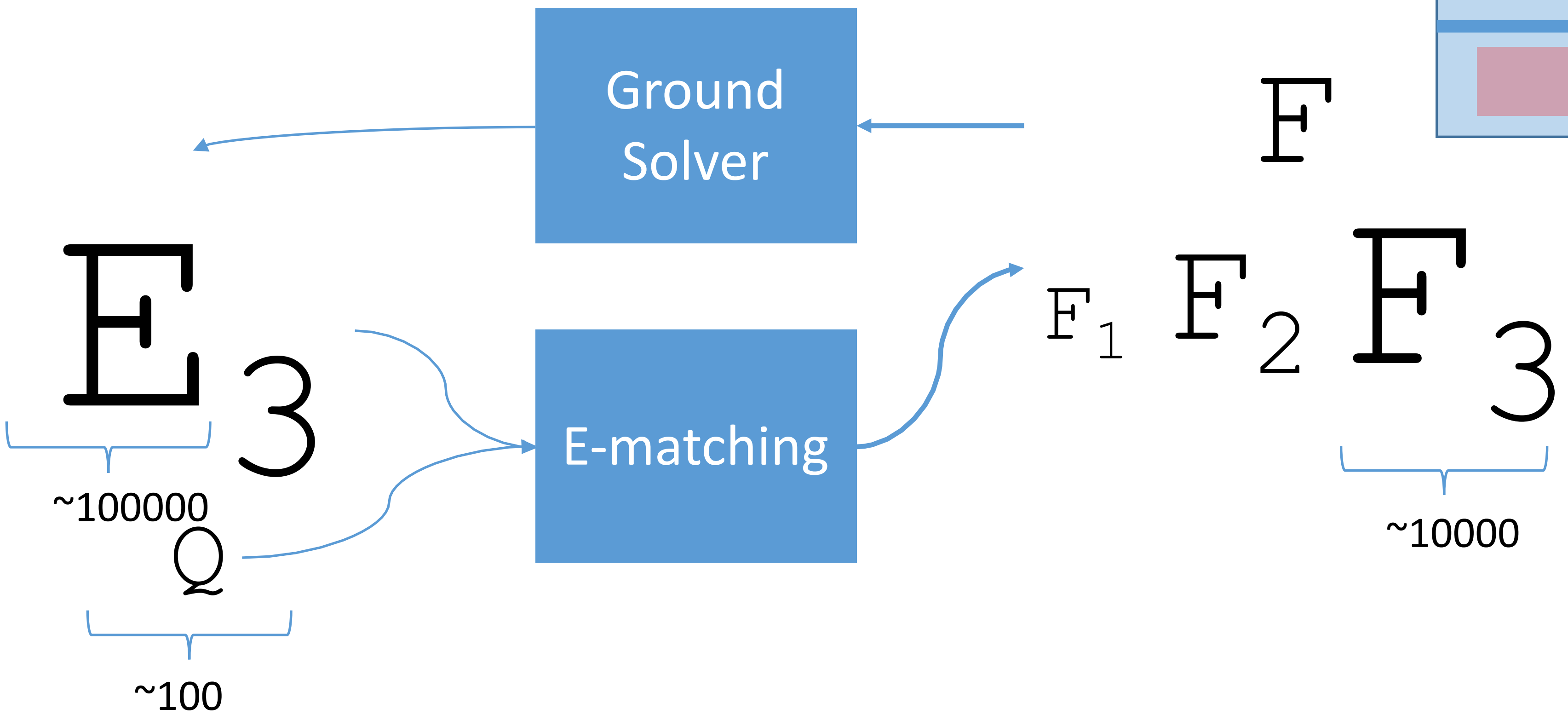
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Satisfying assignments contain 1000s of terms in $\mathbb{E}$, 100s of ∀ in Q
  - Leads to 100s, 1000s

Conflict-Based

E-matching

Model Based

E-matching: Too Many Instances



Ground Solver

E-matching

$\mathbb{E}_3$

~100000

$\mathbb{Q}$

~100

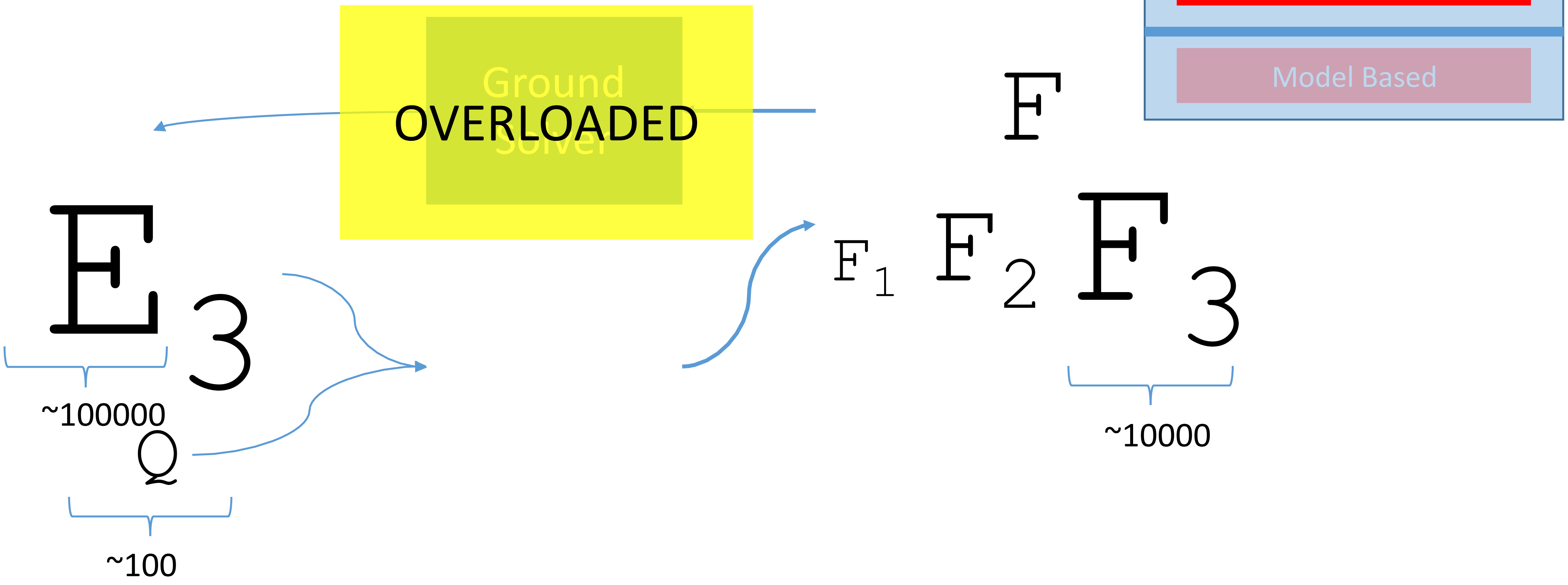
$\mathbb{F}$

$\mathbb{F}_1$ $\mathbb{F}_2$ $\mathbb{F}_3$

~10000

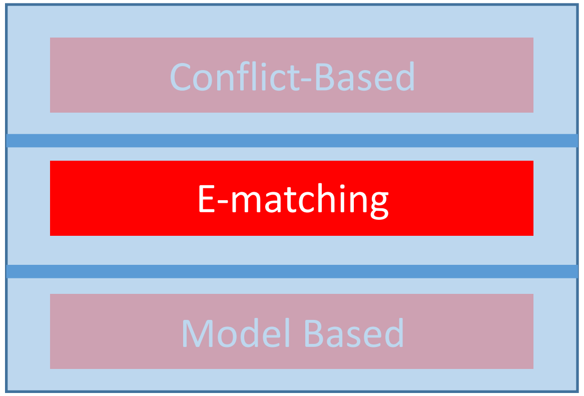
- Typical problems in applications:
  - $\mathbb{F}$ contains 1000s of clauses
  - Satisfying assignments contain 1000s of terms in $\mathbb{E}$, 100s of ∀ in Q
  - Leads to 100s, 1000s, 10000s of instances

E-matching: Too Many Instances



⇒ Ground solver is overloaded, loop becomes slow, …solver times out

E-matching: Too Many Instances



Conflict-Based

E-matching

Model Based

# 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**

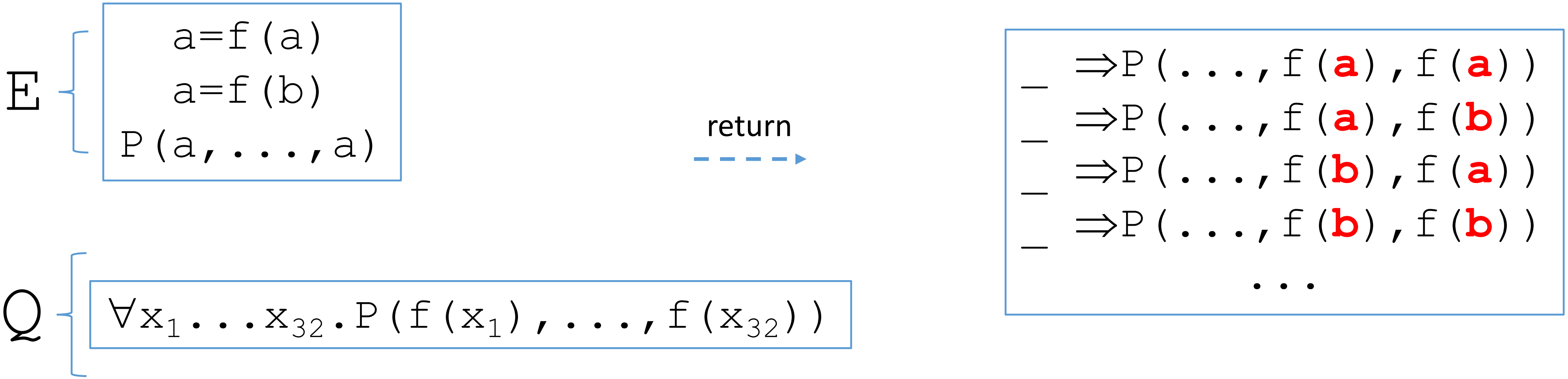
    (Above, 16.6% required >10k, max 19.5M by z3 on a software verification benchmark from TPTP)

Conflict-Based

E-matching

Model Based

E-matching: Too Many Instances



E

$a=f(a)$

$a=f(b)$

$P(a,\ldots,a)$

Q

$\forall x_1\ldots x_{32}.P(f(x_1),\ldots,f(x_{32}))$

return

$\_ \Rightarrow P(\ldots,f(\mathbf{a}),f(\mathbf{a}))$

$\_ \Rightarrow P(\ldots,f(\mathbf{a}),f(\mathbf{b}))$

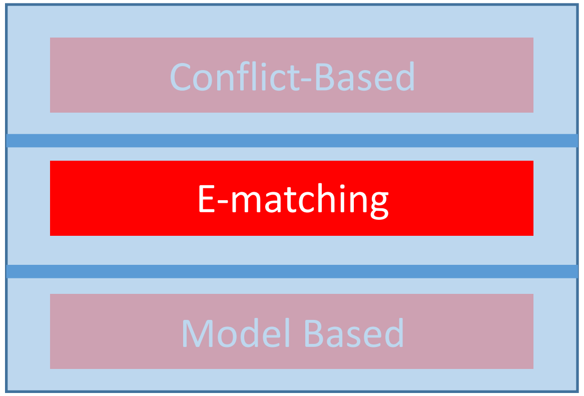
$\_ \Rightarrow P(\ldots,f(\mathbf{b}),f(\mathbf{a}))$

$\_ \Rightarrow P(\ldots,f(\mathbf{b}),f(\mathbf{b}))$

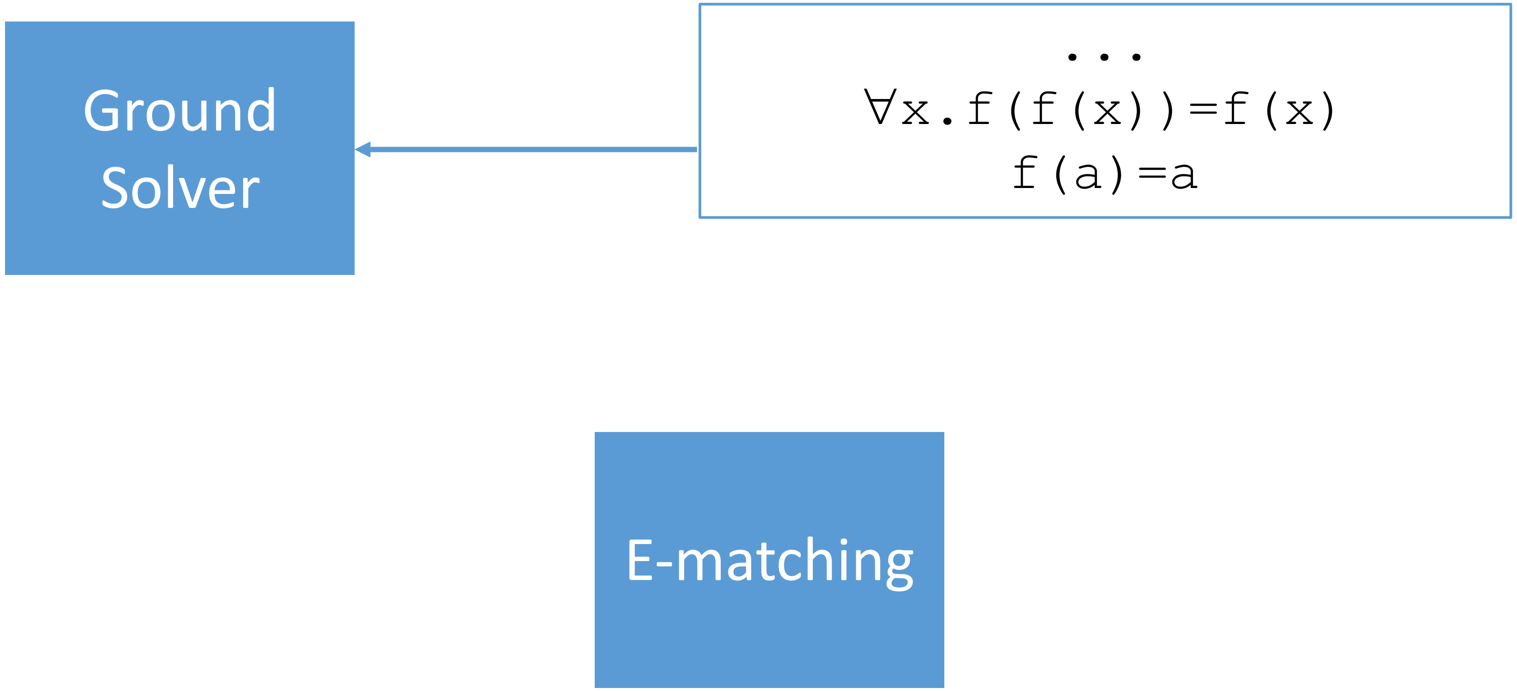
$\ldots$

$\Rightarrow$ In fact, E-matching may be *exponential*, above produces 2^{32} instances

- Thus, we limit # matches per ground term/pattern pair



E-matching: Non-termination



...

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

f(a)=a

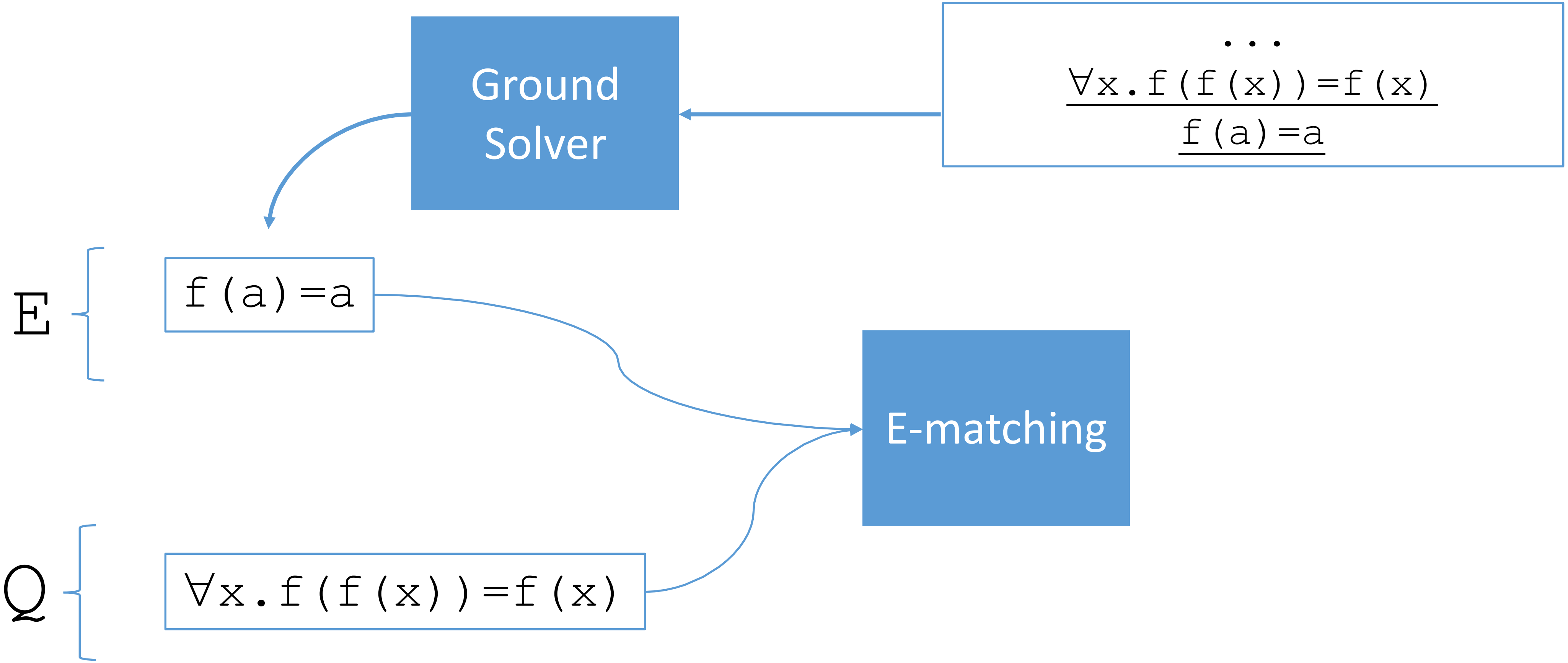
$\Rightarrow$ E-matching may be non-terminating

Conflict-Based

E-matching

Model Based

E-matching: Non-termination



...

$\underline{\forall x. f(f(x))=f(x)}$

$\underline{f(a)=a}$

Ground Solver

E

$f(a)=a$

E-matching

Q

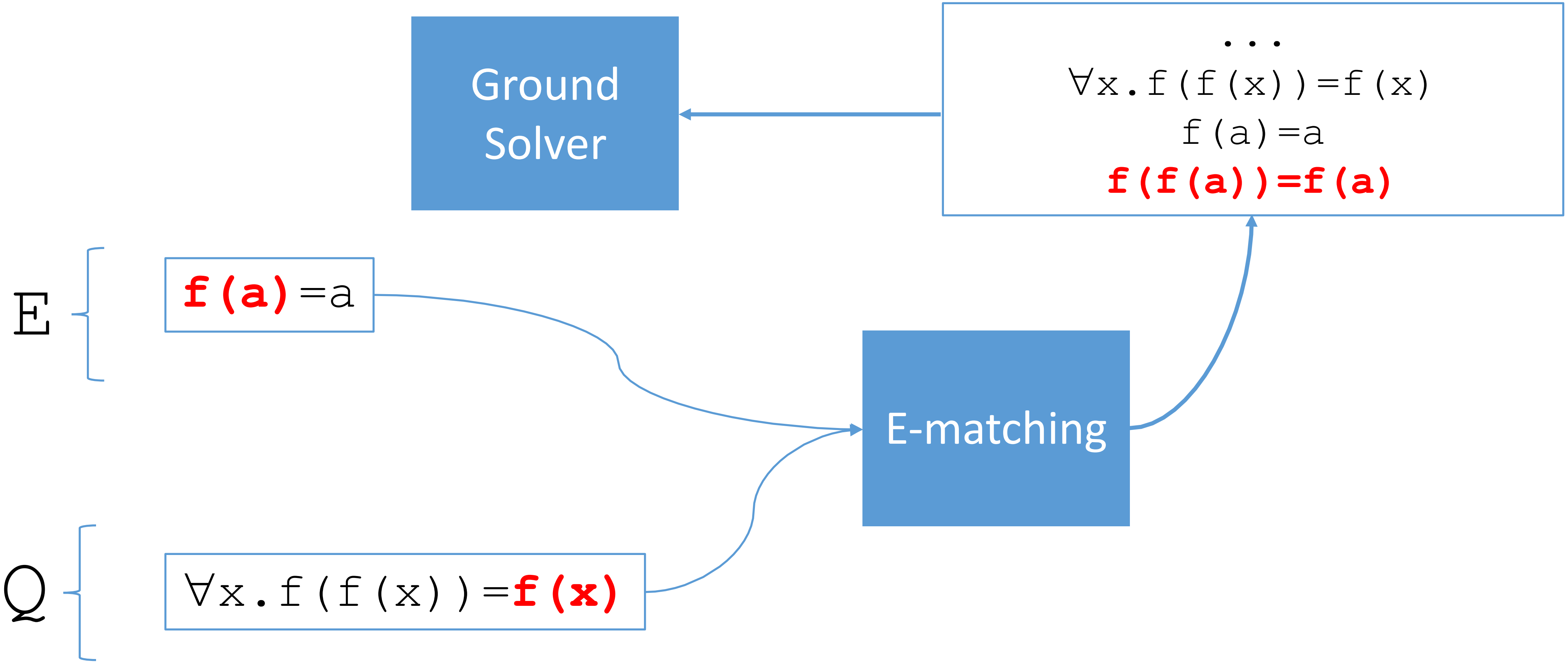
$\forall x. f(f(x))=f(x)$

Conflict-Based

E-matching

Model Based

E-matching: Non-termination



Ground Solver

...

∀x.f(f(x))=f(x)

f(a)=a

f(f(a))=f(a)

E

f(a)=a

E-matching

Q

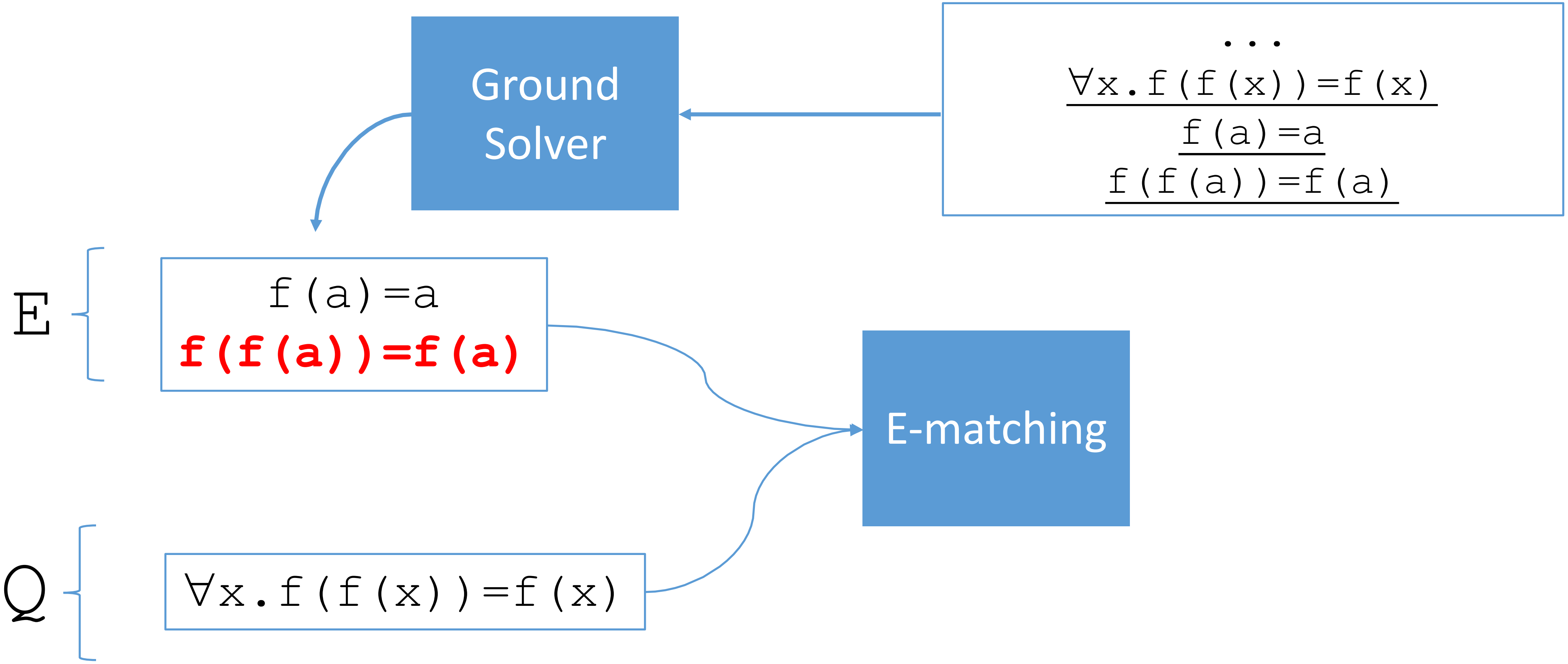
∀x.f(f(x))=**f(x)**

E-matching: Non-termination

Conflict-Based

E-matching

Model Based



...

$\underline{\forall x. f(f(x))=f(x)}$

$\underline{f(a)=a}$

$\underline{f(f(a))=f(a)}$

Ground Solver

E

f(a)=a

f(f(a))=f(a)

E-matching

Q

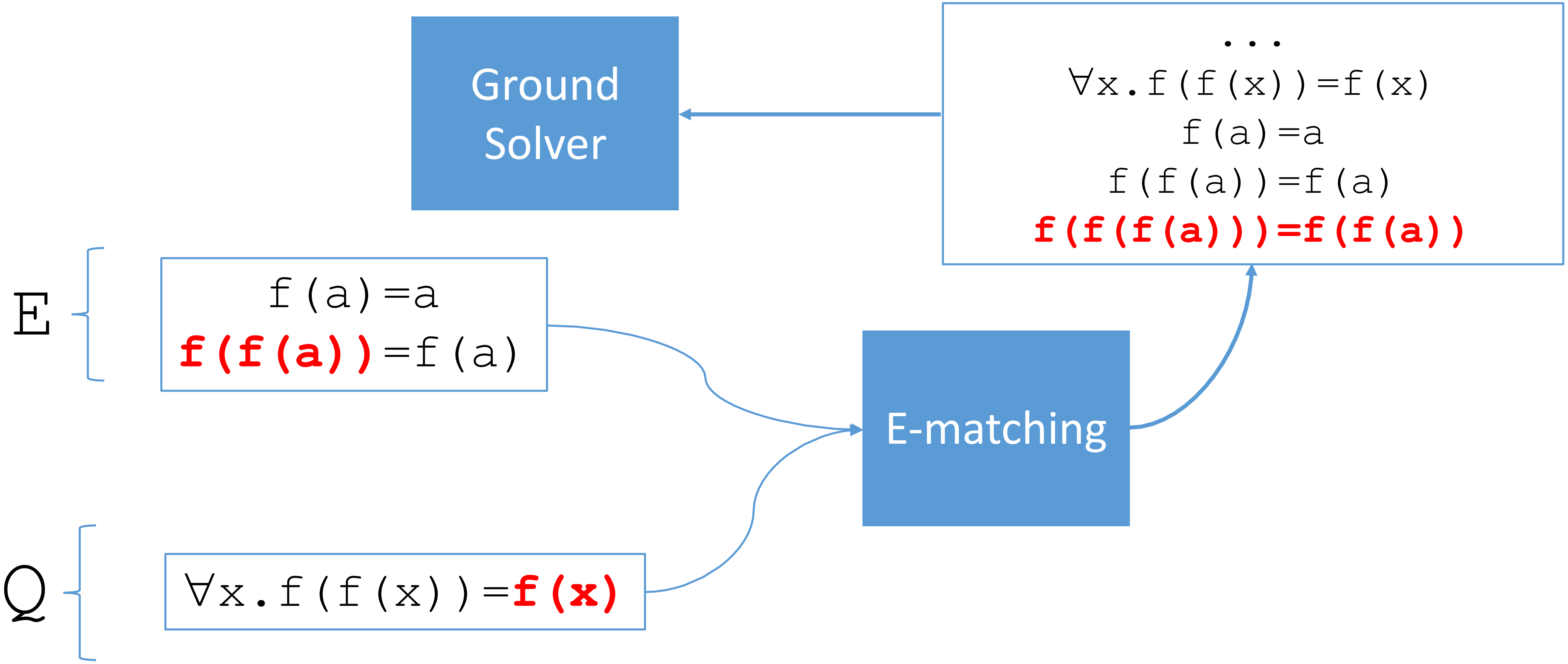
$\forall x. f(f(x))=f(x)$

Conflict-Based

E-matching

Model Based

E-matching: Non-termination



Ground Solver

...

∀x.f(f(x))=f(x)

f(a)=a

f(f(a))=f(a)

f(f(f(a)))=f(f(a))

E

f(a)=a

f(f(a))=f(a)

E-matching

Q

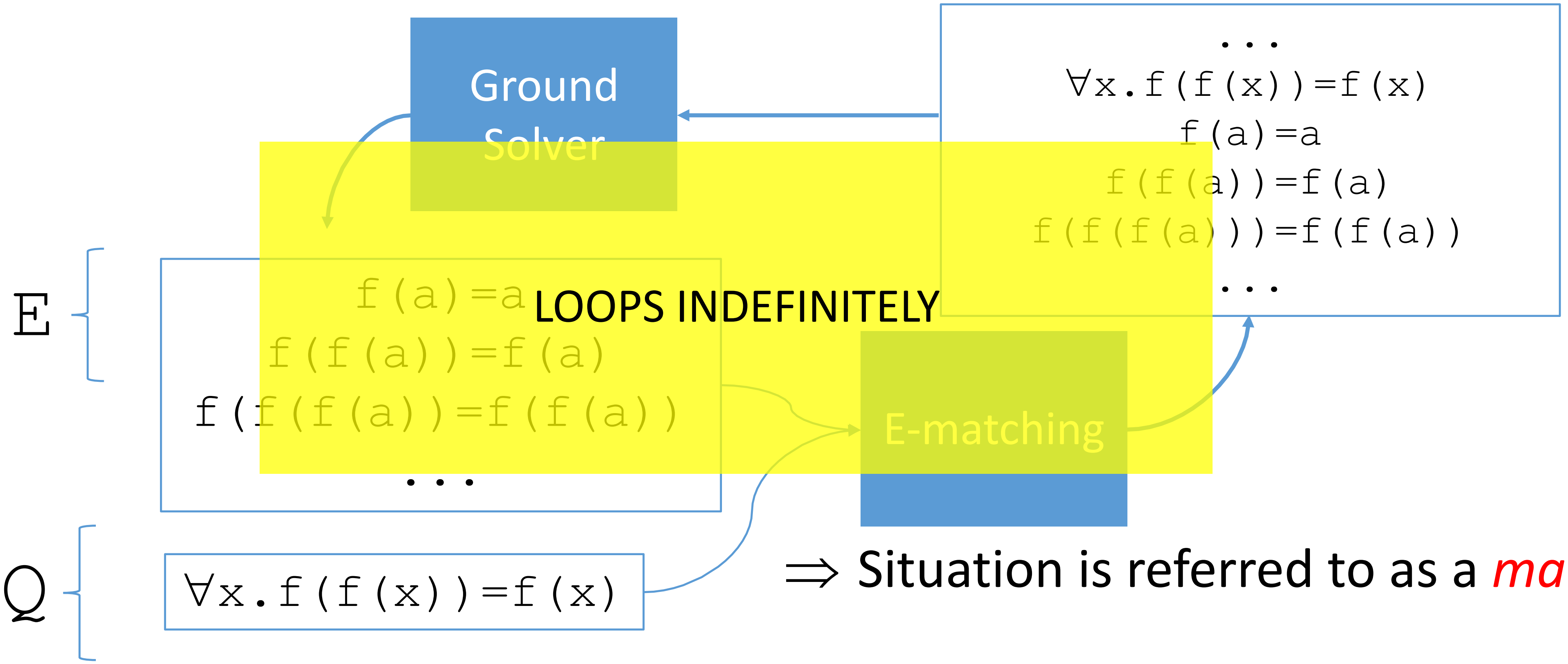
∀x.f(f(x))=**f(x)**

Conflict-Based

E-matching

Model Based

E-matching: Non-termination



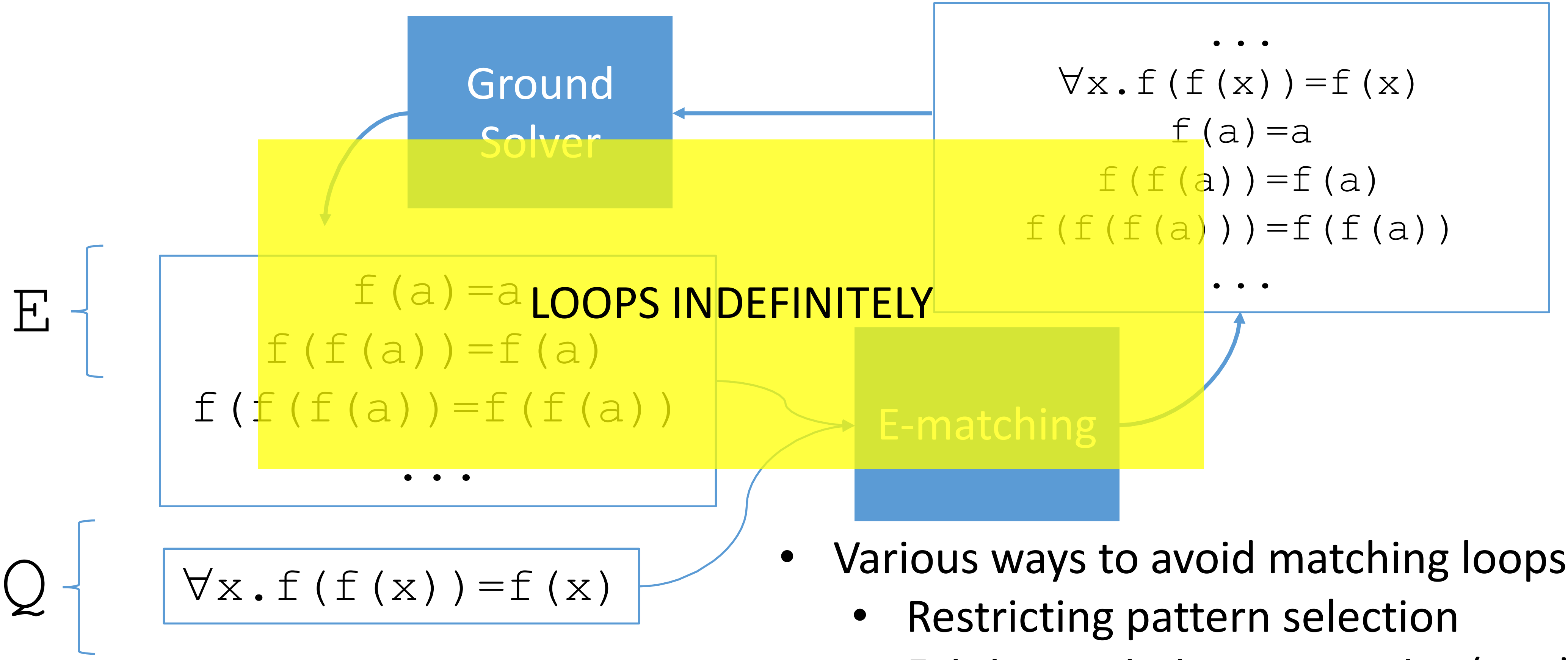
⇒ Situation is referred to as a *matching loop*

Conflict-Based

E-matching

Model Based

E-matching: Non-termination



Ground Solver

...

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

$f(a)=a$

$f(f(a))=f(a)$

$f(f(f(a)))=f(f(a))$

...

E

$f(a)=a$

$f(f(a))=f(a)$

$f(f(f(a)))=f(f(a))$

...

LOOPS INDEFINITELY

E-matching

Q

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

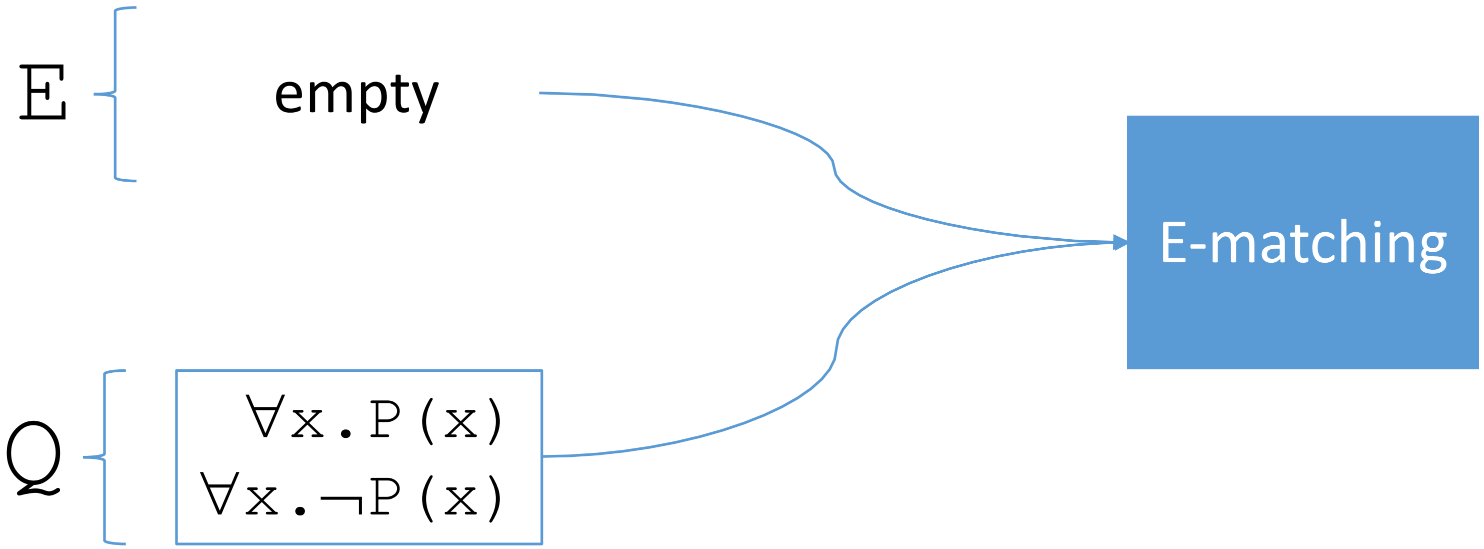
- Various ways to avoid matching loops, e.g. by:
  - Restricting pattern selection
  - Fair instantiations strategies (track “levels”)

Conflict-Based

E-matching

Model Based

E-matching: Incompleteness



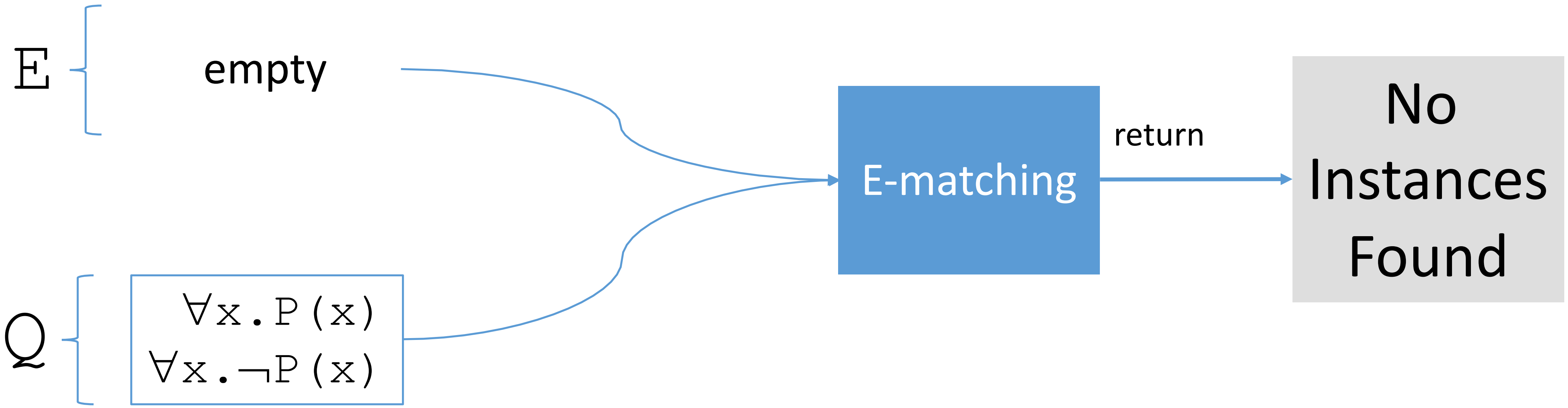
⇒ E-matching is an incomplete procedure

Conflict-Based

E-matching

Model Based

E-matching: Incompleteness



$\forall x. P(x)$

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

$\Rightarrow$ If E-matching produces no instances,

this *does not guarantee $E \cup Q$ is T-satisfiable*

E-matching: Summary

- Using matching ground terms from `E` against patterns in `Q`:
  - **From `Q`, learn constraints about ground terms `g` from `E`**

E-matching: Summary

- Using matching ground terms from `E` against patterns in `Q`:
  - From `Q`, learn constraints about ground terms `g` from `E`
- Challenges
  - What can we do when there **too many instances** to add?
  - What can we do when there are **no instances** to add, problem is “**sat**”?

E-matching: Summary

- Using matching ground terms from $\mathtt{E}$ against patterns in $\mathtt{Q}$:
  - From $\mathtt{Q}$, learn constraints about ground terms $\mathtt{g}$ from $\mathtt{E}$
- Challenges
  - What can we do when there **too many instances** to add?

    ⇒*Use conflict-based instantiation* **[Reynolds/Tinelli/deMoura FMCAD14]**
  - What can we do when there are **no instances** to add, problem is "**sat**"?

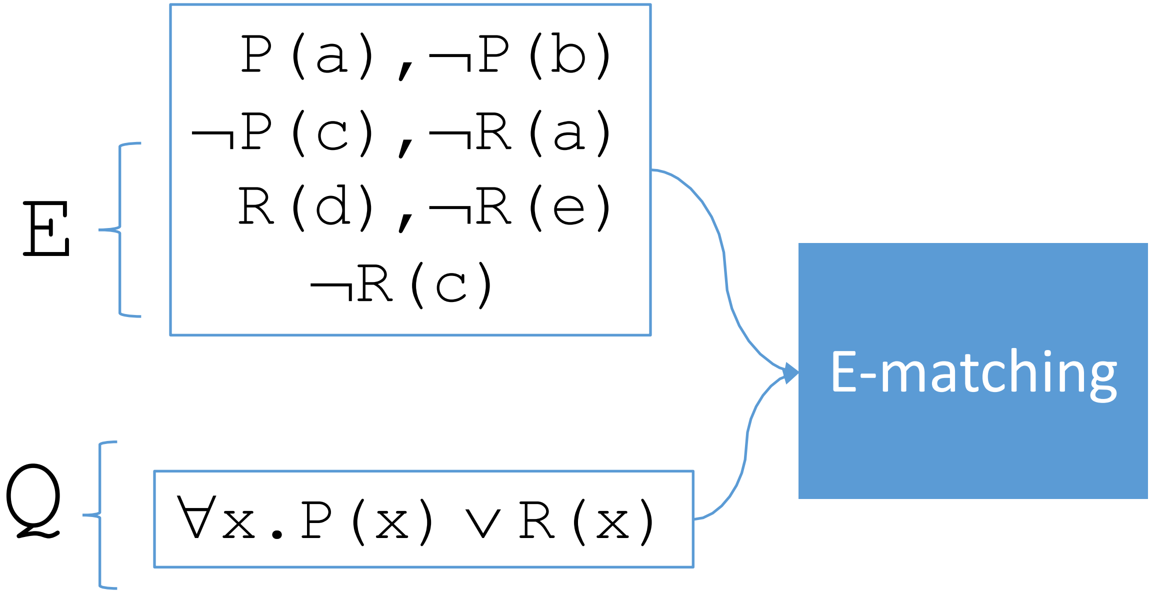
    ⇒*Use model-based instantiation* **[Ge/deMoura CAV09]**

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation



E

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

Q

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

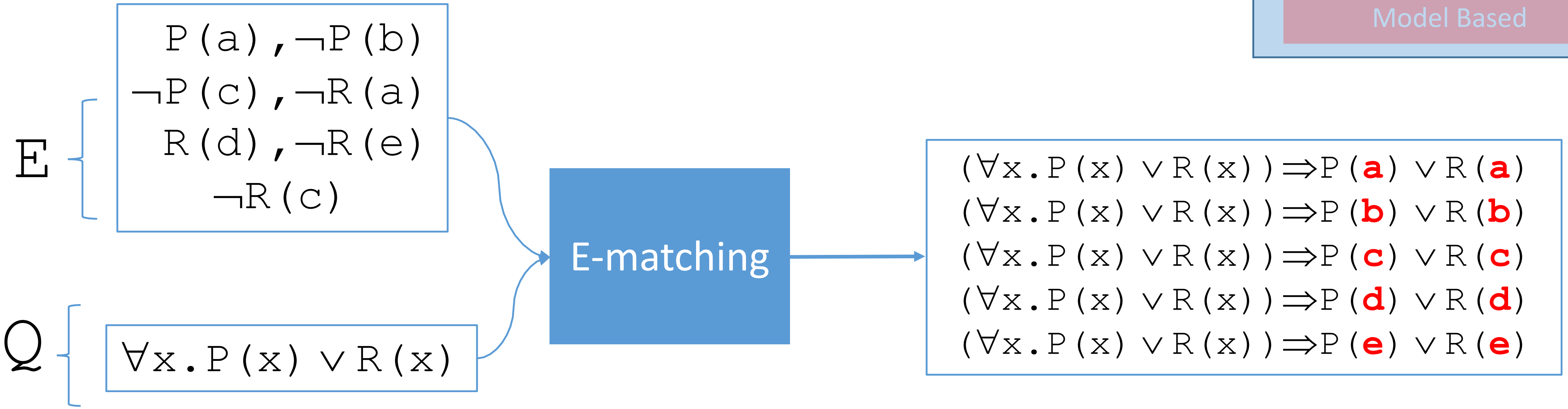
E-matching

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation



E

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

Q

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

E-matching

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

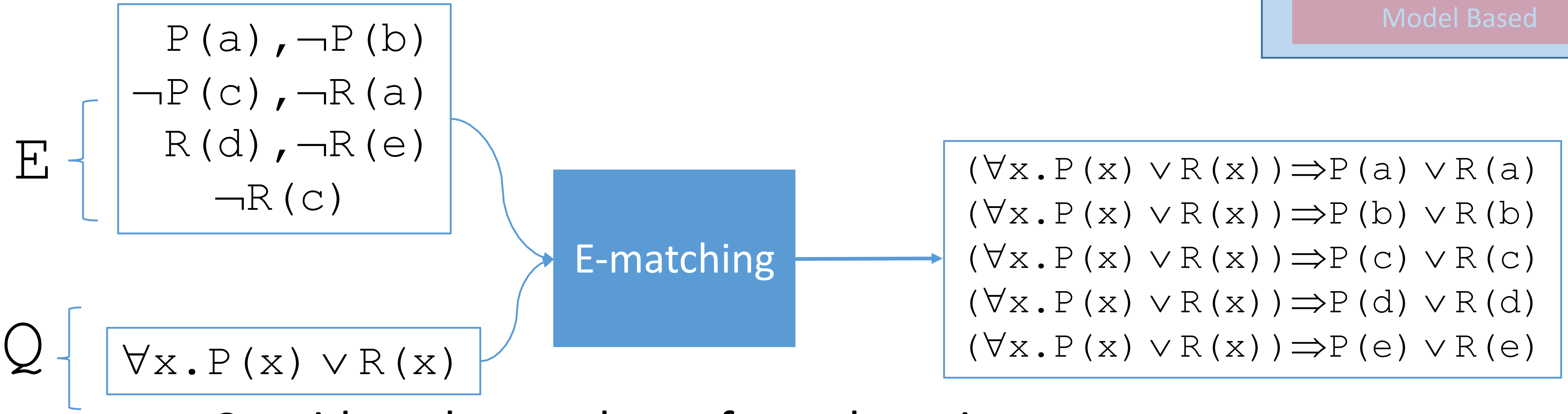
$\Rightarrow$ E-matching would produce $\{x \rightarrow \mathbf{a}\}$, $\{x \rightarrow \mathbf{b}\}$, $\{x \rightarrow \mathbf{c}\}$, $\{x \rightarrow \mathbf{d}\}$, $\{x \rightarrow \mathbf{e}\}$

Conflict-Based

E-matching

Model Based

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:

$$E, Q, P(a) \vee R(a) \models P(a) \vee R(a)$$

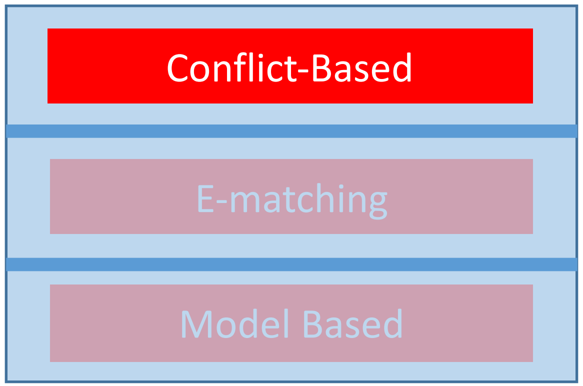
$$E, Q, P(b) \vee R(b) \models P(b) \vee R(b)$$

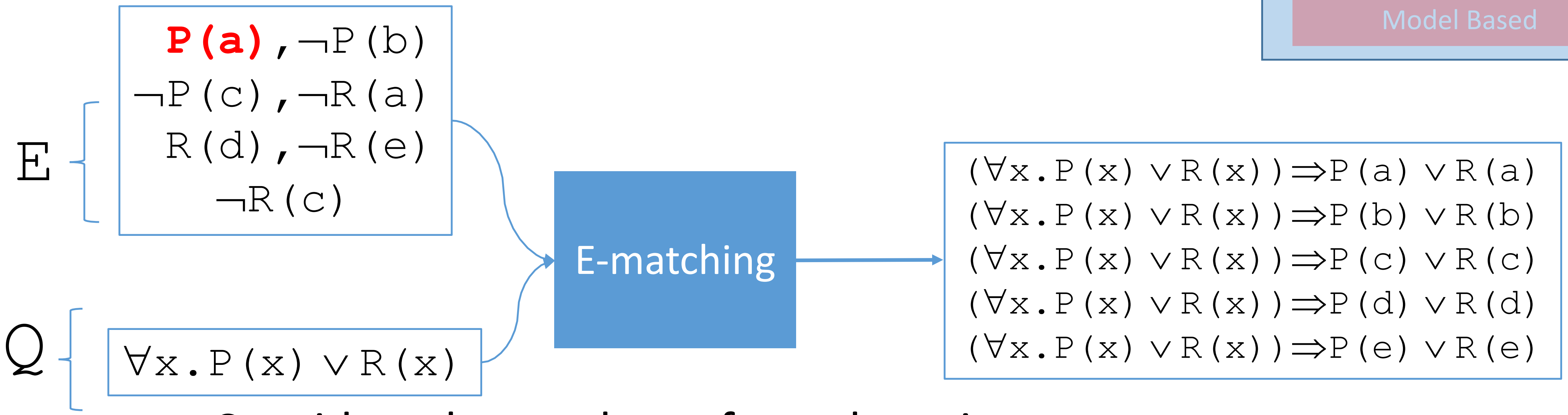
$$E, Q, P(c) \vee R(c) \models P(c) \vee R(c)$$

$$E, Q, P(d) \vee R(d) \models P(d) \vee R(d)$$

$$E, Q, P(e) \vee R(e) \models P(e) \vee R(e)$$

Conflict-Based Instantiation





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

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:

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

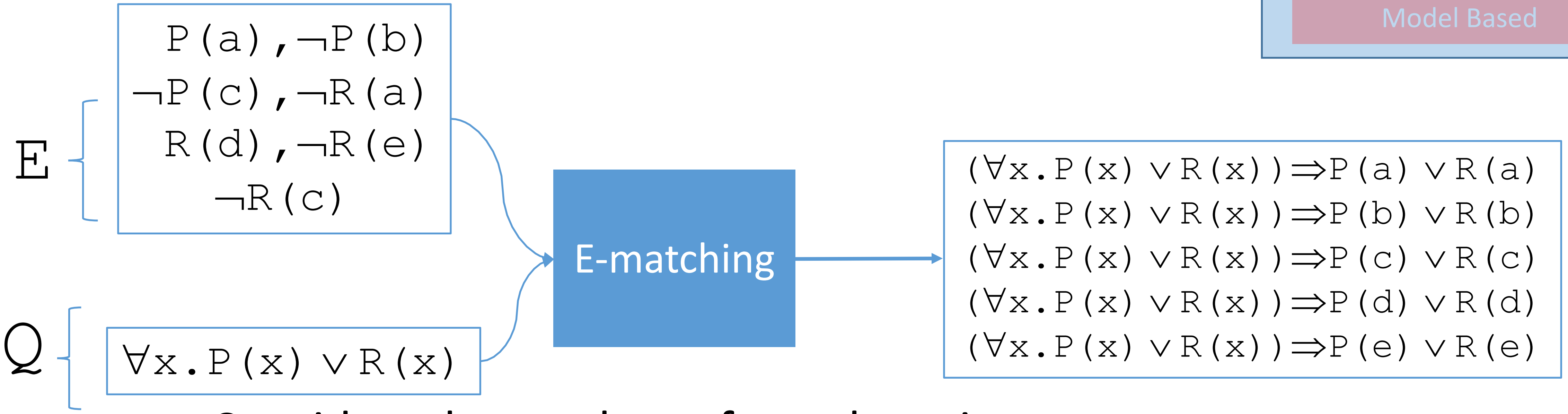
By E, we know $\mathbf{P(a)} \Leftrightarrow \mathbf{T}$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation



E

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

Q

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

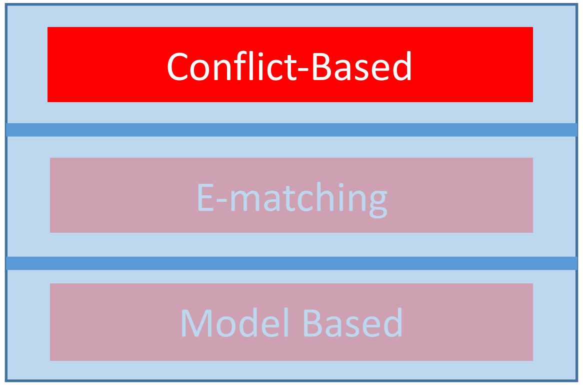
E-matching

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

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

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

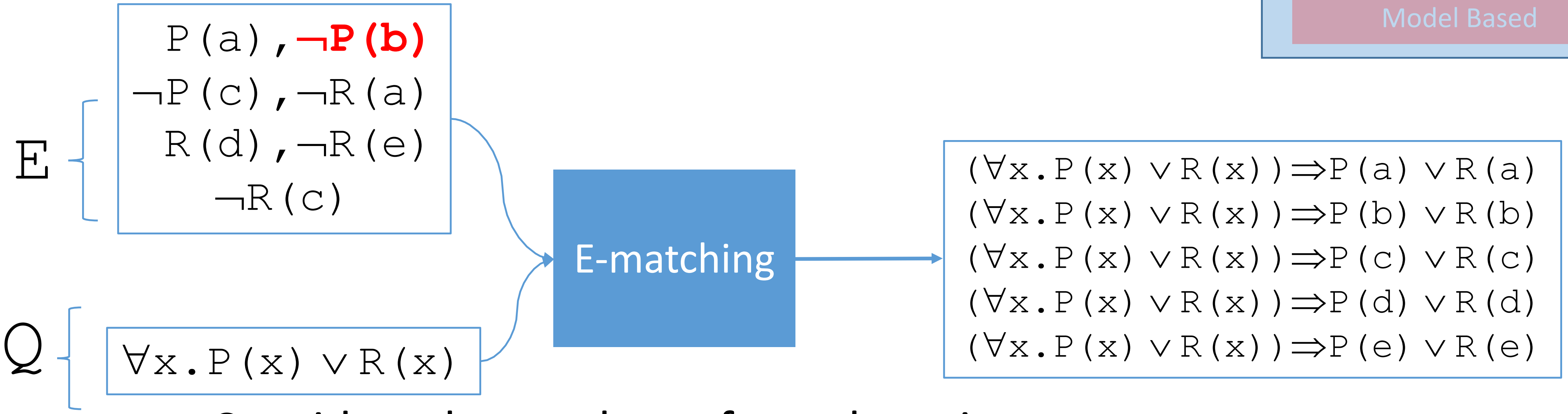
Conflict-Based Instantiation



Conflict-Based

E-matching

Model Based



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

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

⇒ Consider what we learn from these instances:

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

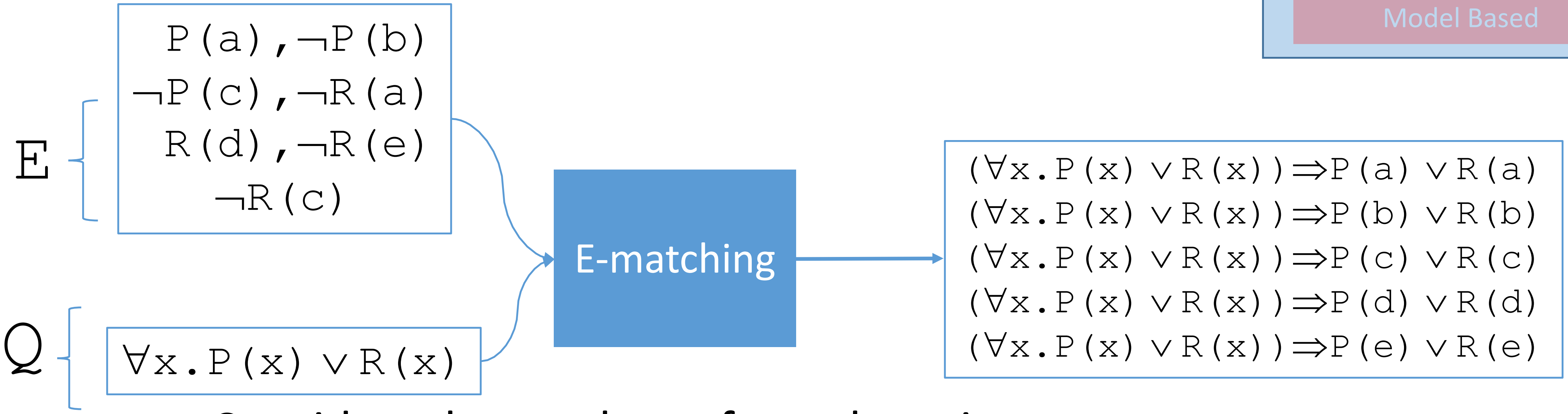
We know **P(b)** ⇔ ⊥

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation



$\mathbb{E}$

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

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

$R(d), \neg R(e)$

$\neg R(c)$

$\mathbb{Q}$

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

E-matching

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

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

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

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

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

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

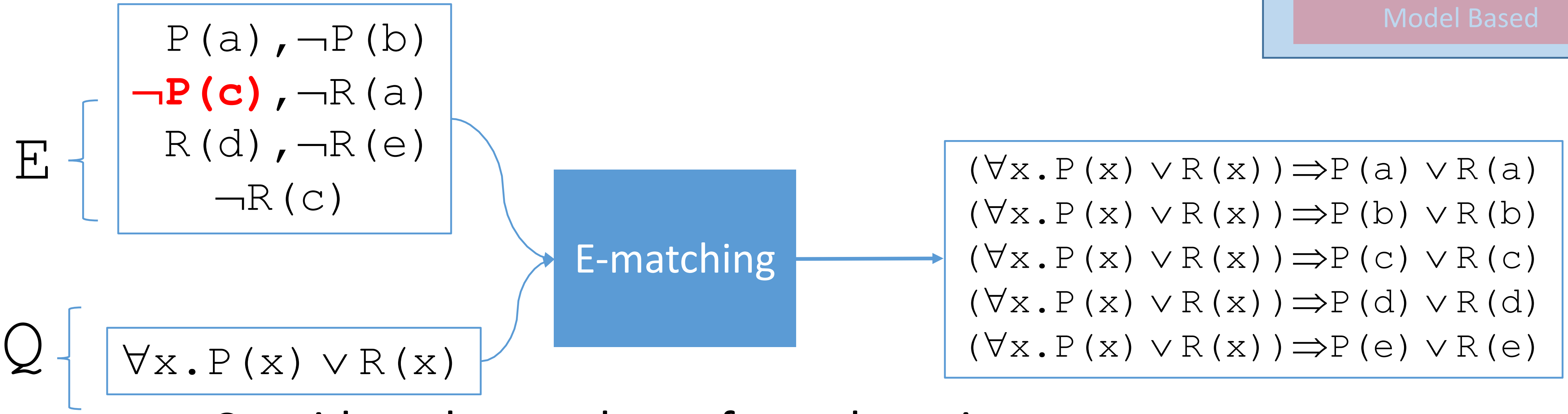
$$\begin{aligned}
\mathbb{E}, \mathbb{Q}, P(a) \vee R(a) &\models \top \\
\mathbb{E}, \mathbb{Q}, P(b) \vee R(b) &\models R(b) \\
\mathbb{E}, \mathbb{Q}, P(c) \vee R(c) &\models P(c) \vee R(c) \\
\mathbb{E}, \mathbb{Q}, P(d) \vee R(d) &\models P(d) \vee R(d) \\
\mathbb{E}, \mathbb{Q}, P(e) \vee R(e)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

Conflict-Based

E-matching

Model Based

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

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

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

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

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

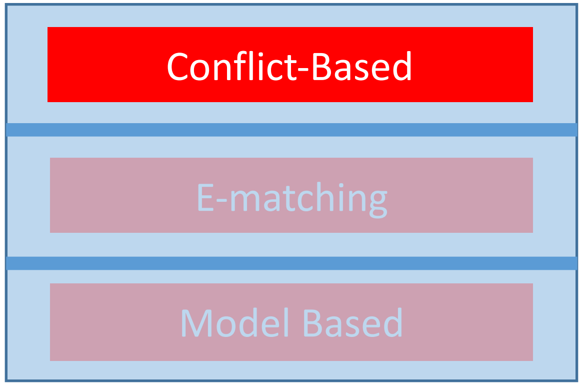
$(\forall x.P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

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

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

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

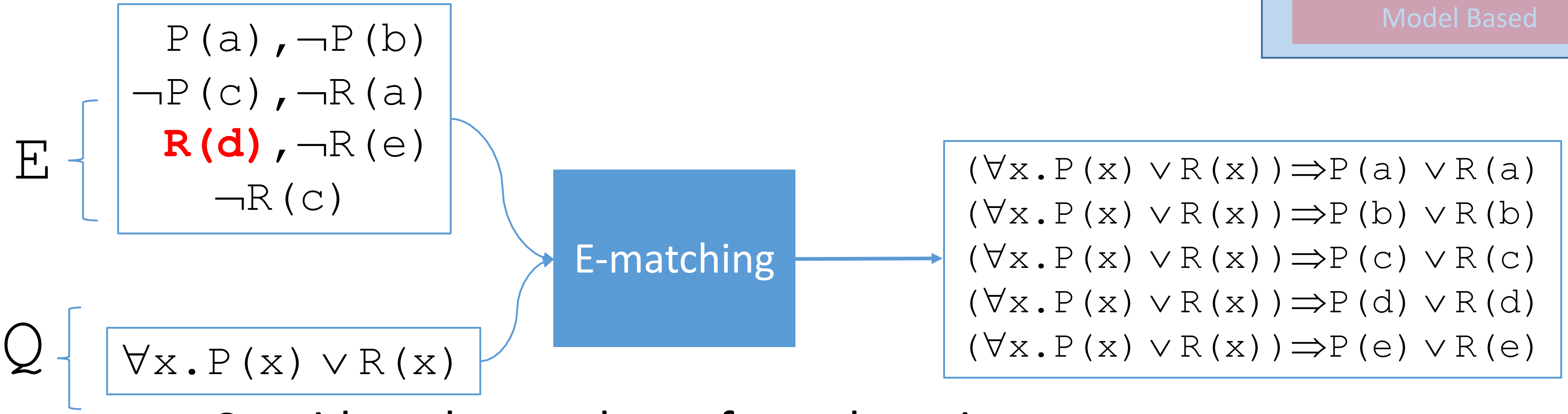
Conflict-Based Instantiation



Conflict-Based

E-matching

Model Based



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:

$$E, Q, P(a) \vee R(a) \models T$$

$$E, Q, P(b) \vee R(b) \models R(b)$$

$$E, Q, P(c) \vee R(c) \models R(c)$$

$$E, Q, P(d) \vee R(d) \models T$$

$$E, Q, P(e) \vee R(e) \models P(e) \vee R(e)$$

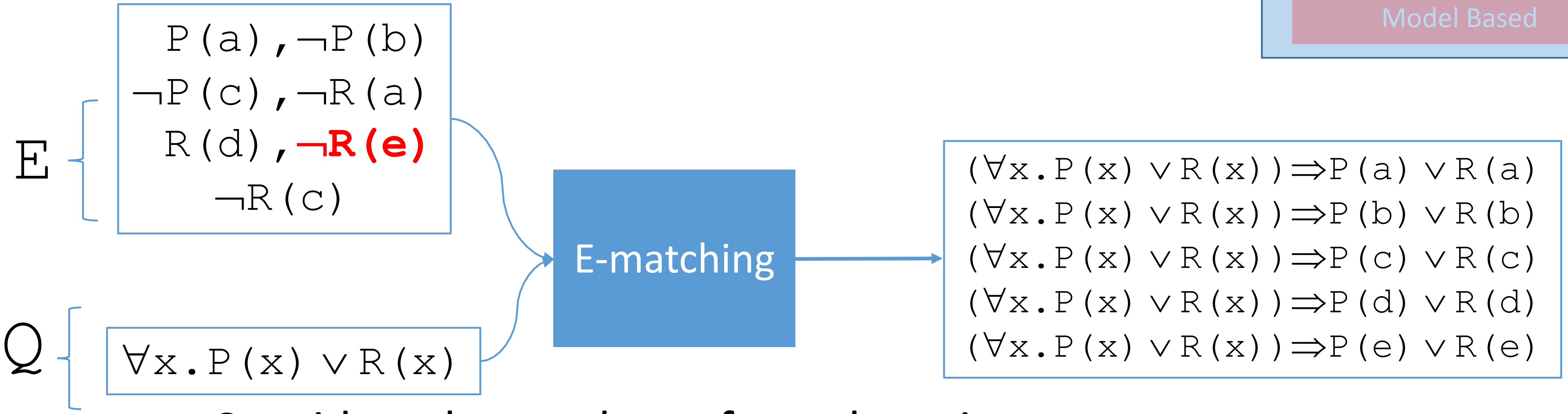
We know $\mathbf{R(d)} \Leftrightarrow \mathbf{T}$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation

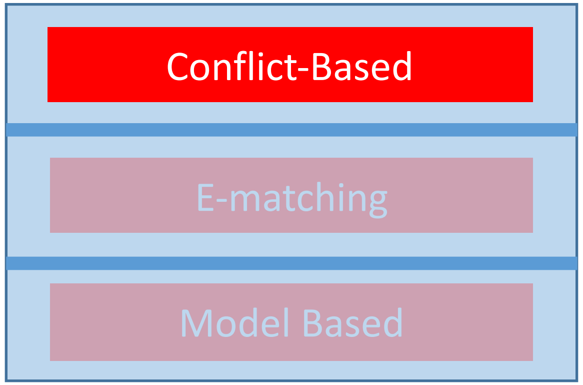

E
$P(a), \neg P(b)$
$\neg P(c), \neg R(a)$
$R(d), \neg R(e)$
$\neg R(c)$
Q
$\forall x. P(x) \vee R(x)$
E-matching
$(\forall x. P(x) \vee R(x)) \Rightarrow P(a) \vee R(a)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(b) \vee R(b)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(c) \vee R(c)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(d) \vee R(d)$
$(\forall x. P(x) \vee R(x)) \Rightarrow P(e) \vee R(e)$

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

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

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

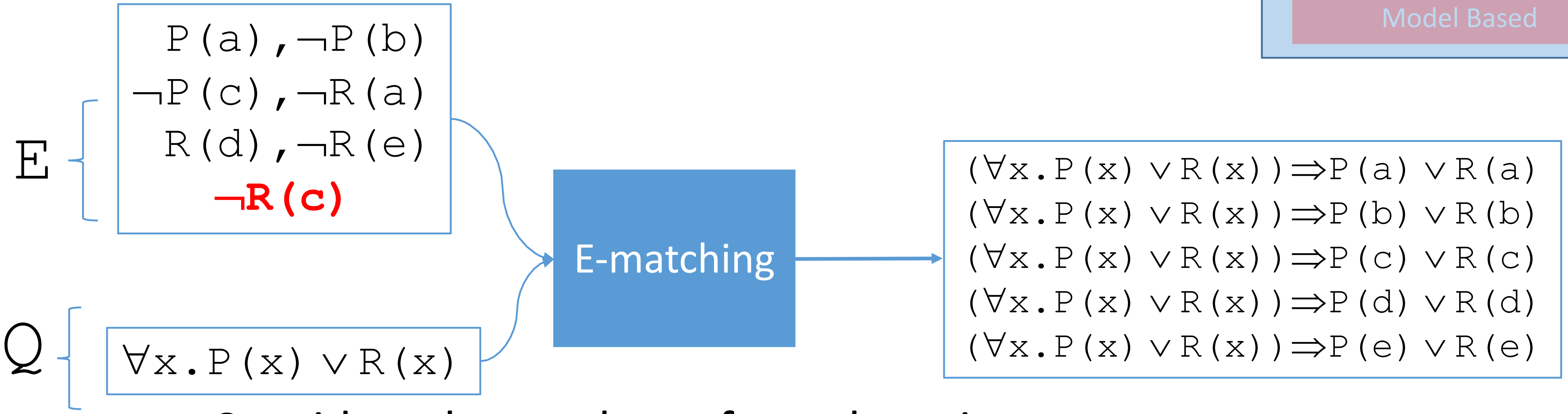
Conflict-Based Instantiation



Conflict-Based

E-matching

Model Based



E

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

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

$R(d), \neg R(e)$

$\mathbf{\neg R(c)}$

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:

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

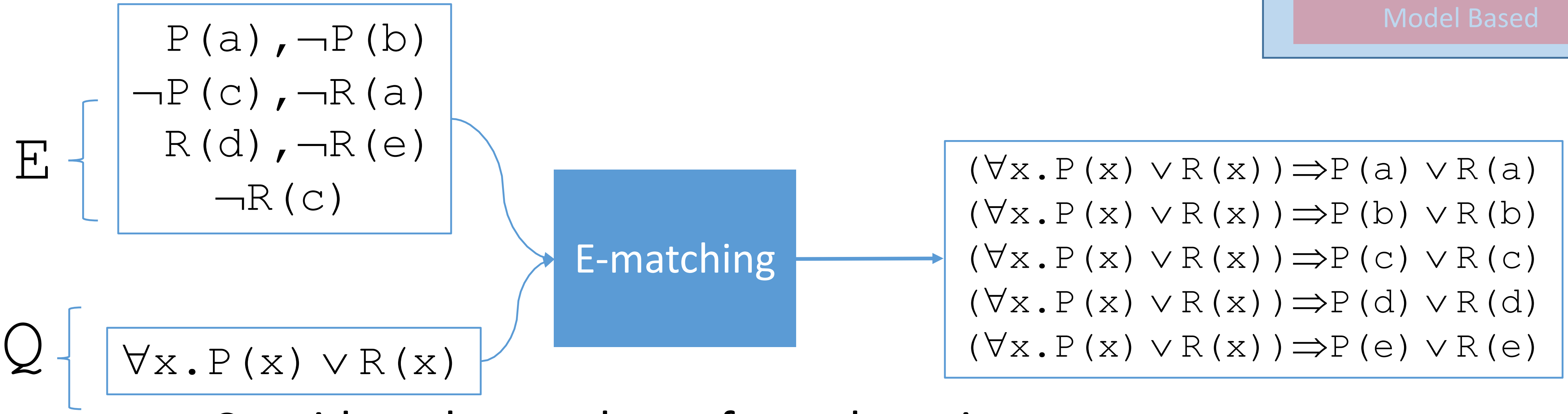
We know $\mathbf{R(c)} \Leftrightarrow \bot$

Conflict-Based Instantiation

Conflict-Based

E-matching

Model Based



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:

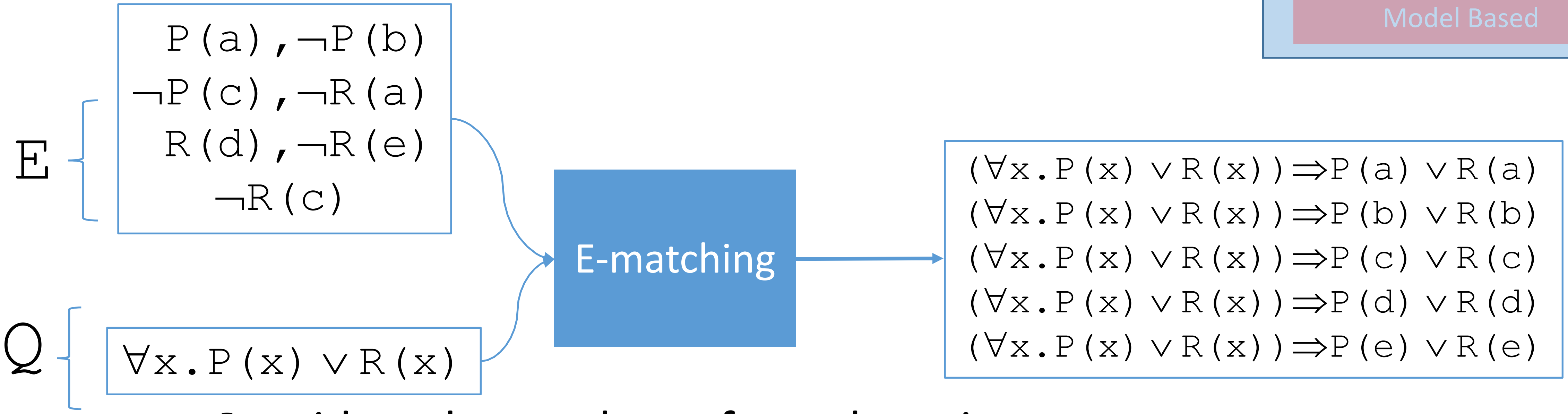
$$
\begin{array}{l|l}
E, Q, P(a) \vee R(a) & \models \top \\
E, Q, P(b) \vee R(b) & \models R(b) \\
E, Q, P(c) \vee R(c) & \models \bot \\
E, Q, P(d) \vee R(d) & \models \top \\
E, Q, P(e) \vee R(e) & \models P(e)
\end{array}
$$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation



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

$E,Q,P(a)\vee R(a) \models \top$

$E,Q,P(b)\vee R(b) \models R(b)$

$\mathbf{E,Q,P(c)\vee R(c)} \models \bot$

$E,Q,P(d)\vee R(d) \models \top$

$E,Q,P(e)\vee R(e) \models P(e)$

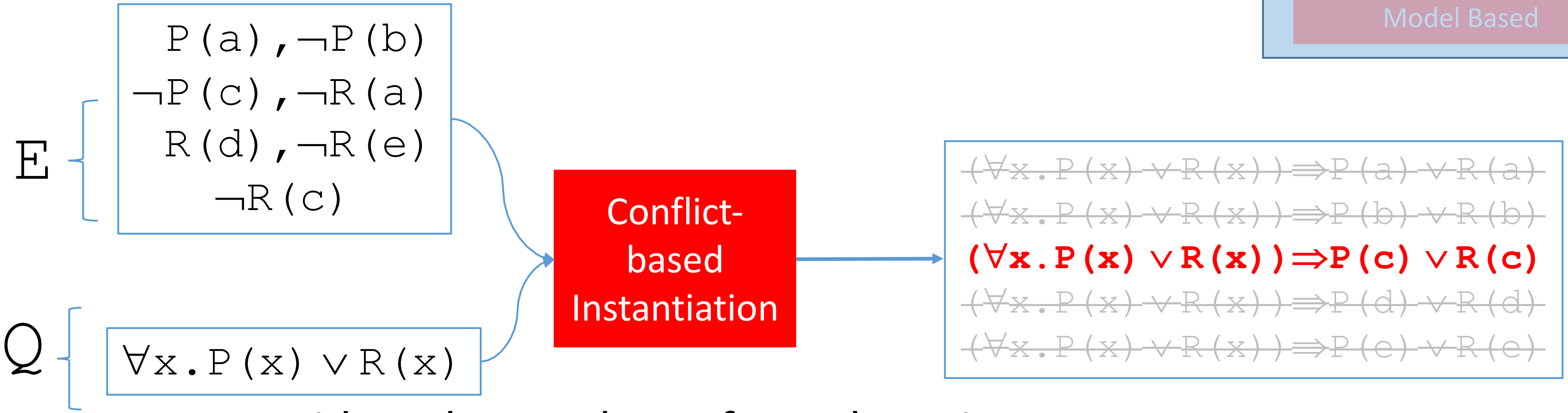
$P(c) \vee R(c)$ *is a* ***conflicting instance*** *for* (E,Q)*!*

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation



E

P(a), ¬P(b)
¬P(c), ¬R(a)
R(d), ¬R(e)
¬R(c)

Q

∀x.P(x) ∨ R(x)

Conflict-based Instantiation

~~(∀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:

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

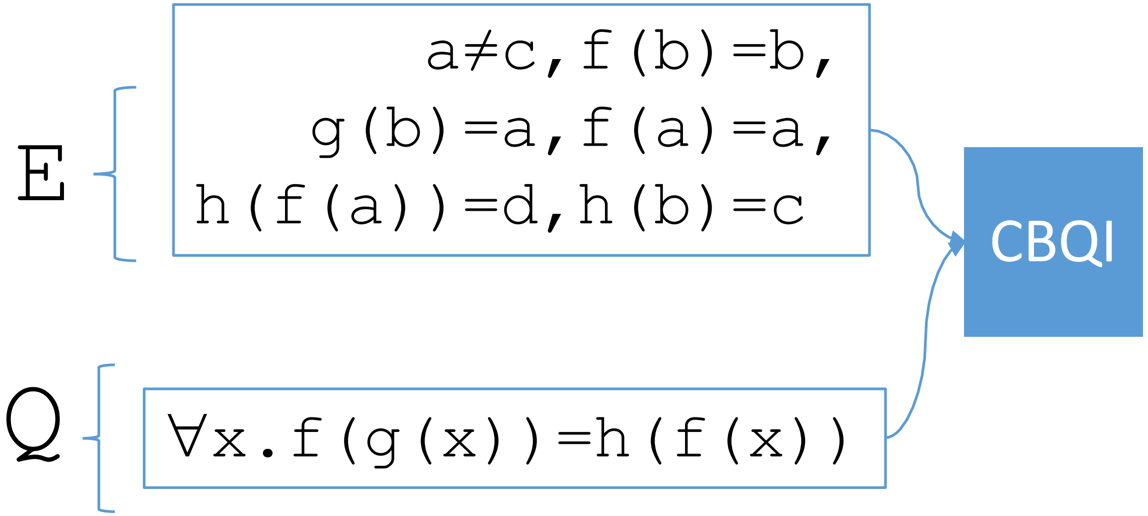
Since P(c) ∨ R(c) suffices to derive ⊥, return ***only*** this instance

Conflict-Based

E-matching

Model Based

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

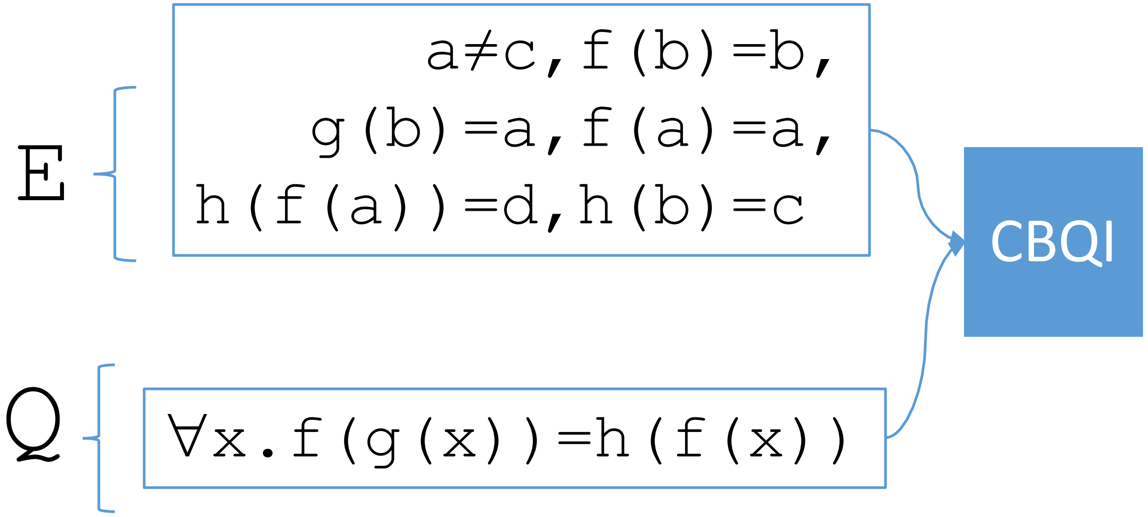
$\forall x. f(g(x))=h(f(x))$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: EUF



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

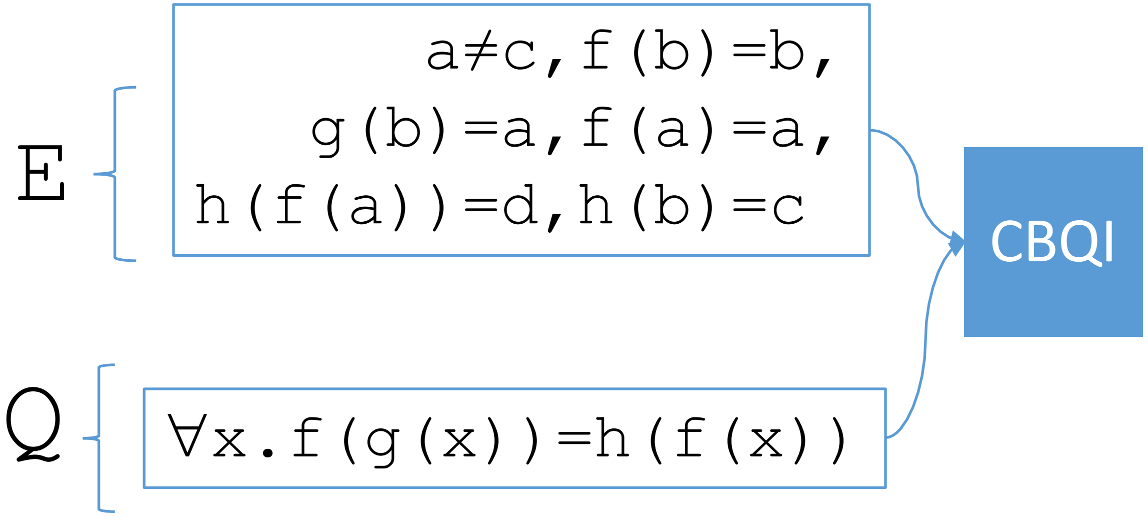
- Is this conflicting for (E, Q)?

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: EUF



E

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

Q

∀x.f(g(x))=h(f(x))

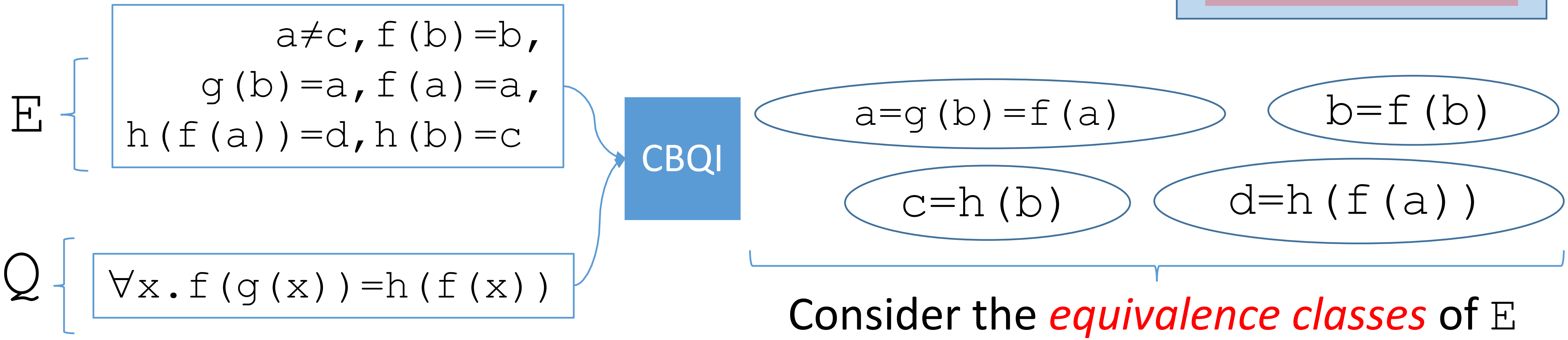
$$E, Q, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$$

Conflict-Based Instantiation: EUF

Conflict-Based

E-matching

Model Based



Consider the *equivalence classes* of E

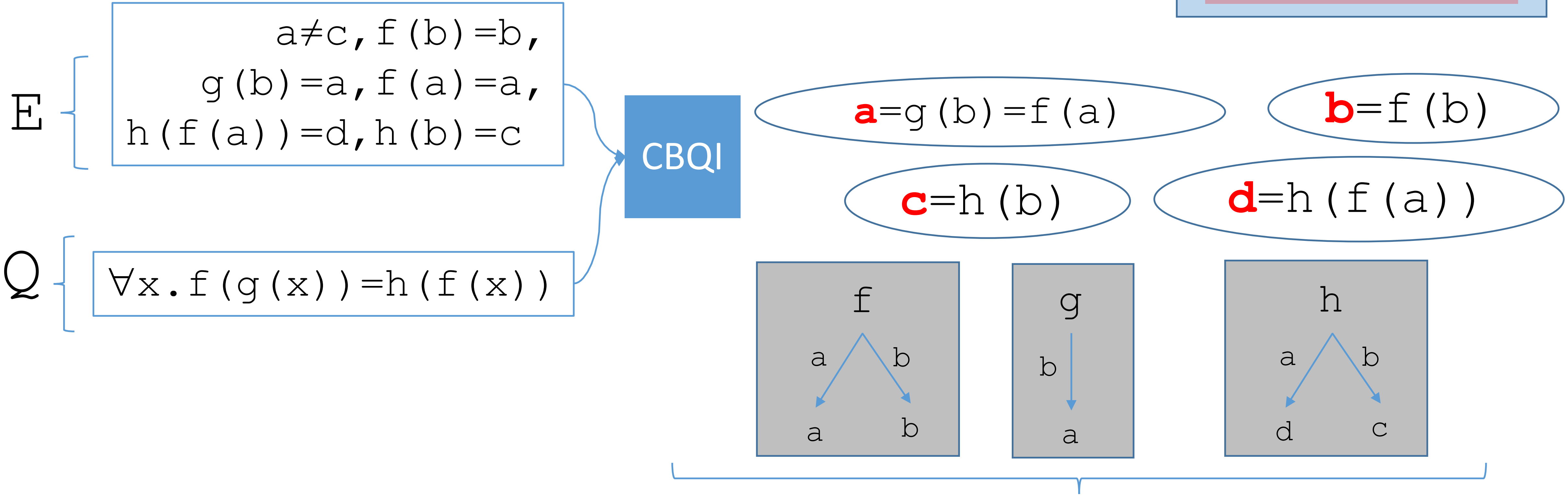
$$\mathtt{E,Q,f(g(b))=h(f(b))} \models_{\mathrm{E}} \mathtt{f(g(b))=h(f(b))}$$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: EUF



Build partial definitions for functions in terms of *representatives*

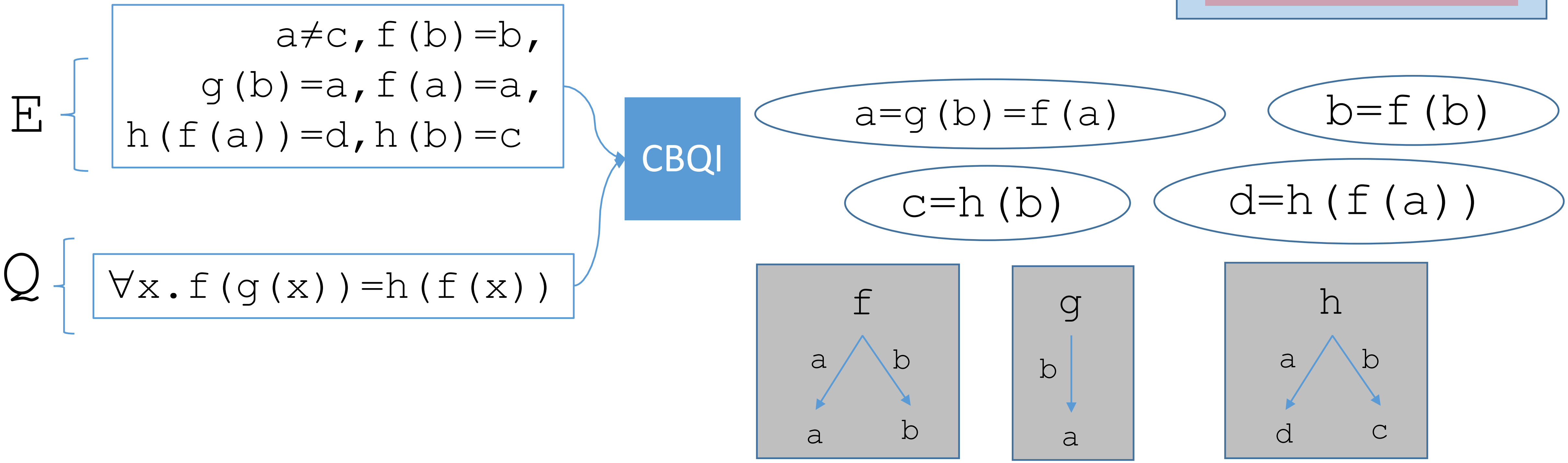
$$E, Q, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: EUF



E: a≠c, 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))

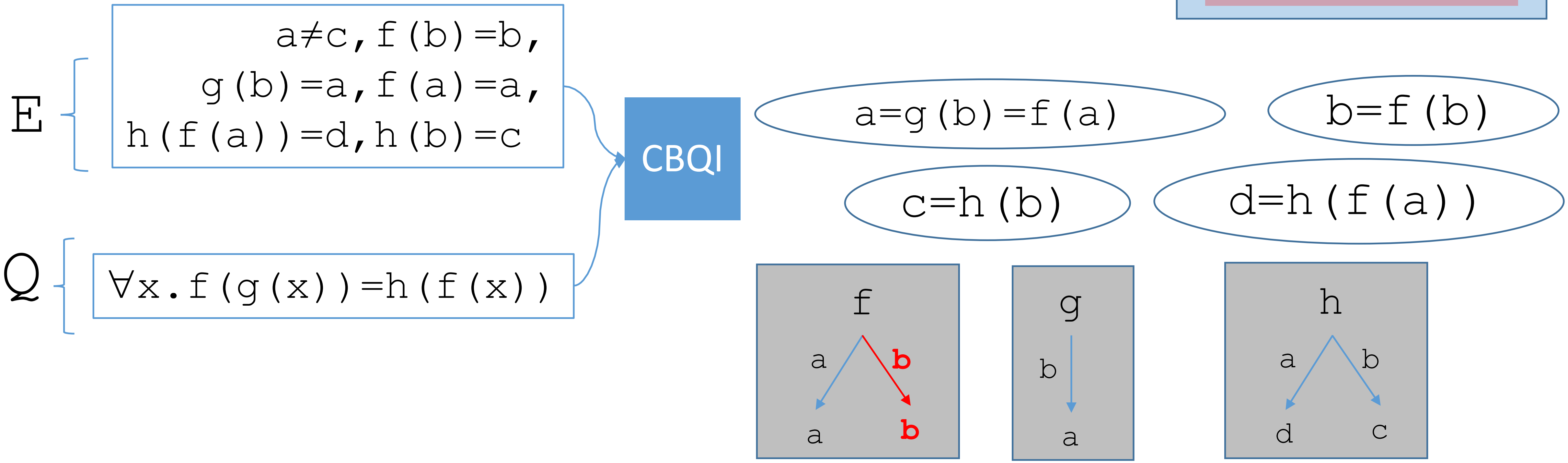
$$E, Q, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$$

Conflict-Based Instantiation: EUF

Conflict-Based

E-matching

Model Based



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

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

$b=f(b)$

$c=h(b)$

$d=h(f(a))$

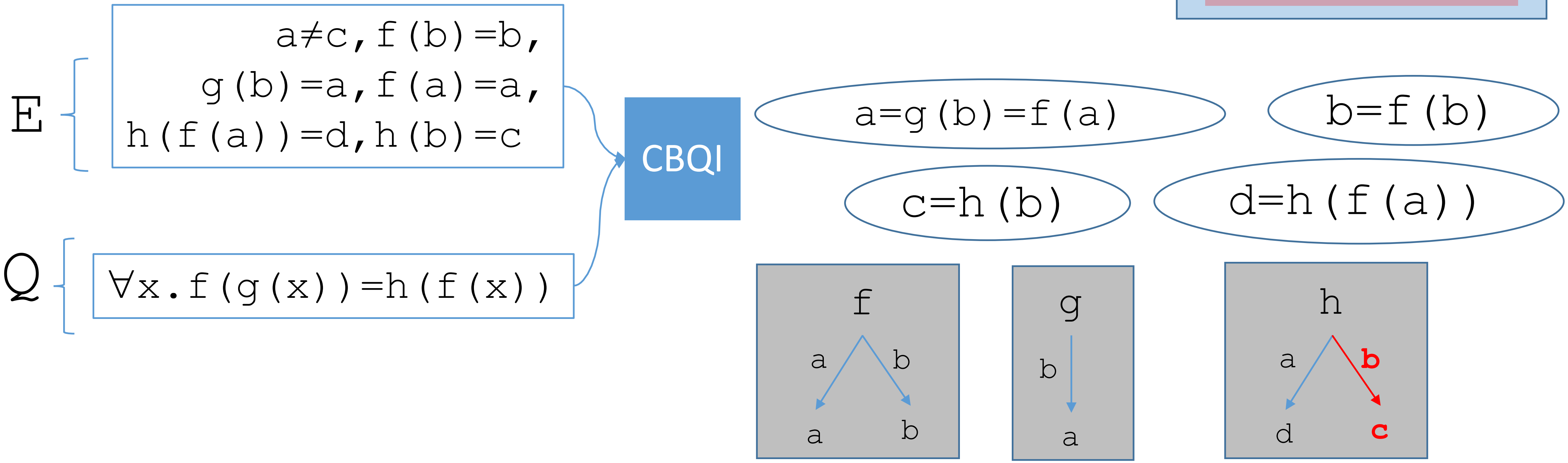
$$E, Q, f(g(b))=h(f(b)) \models_E f(g(b))=h(\ \mathbf{b}\ \ )$$

Conflict-Based

E-matching

Model Based

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

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

$b=f(b)$

$c=h(b)$

$d=h(f(a))$

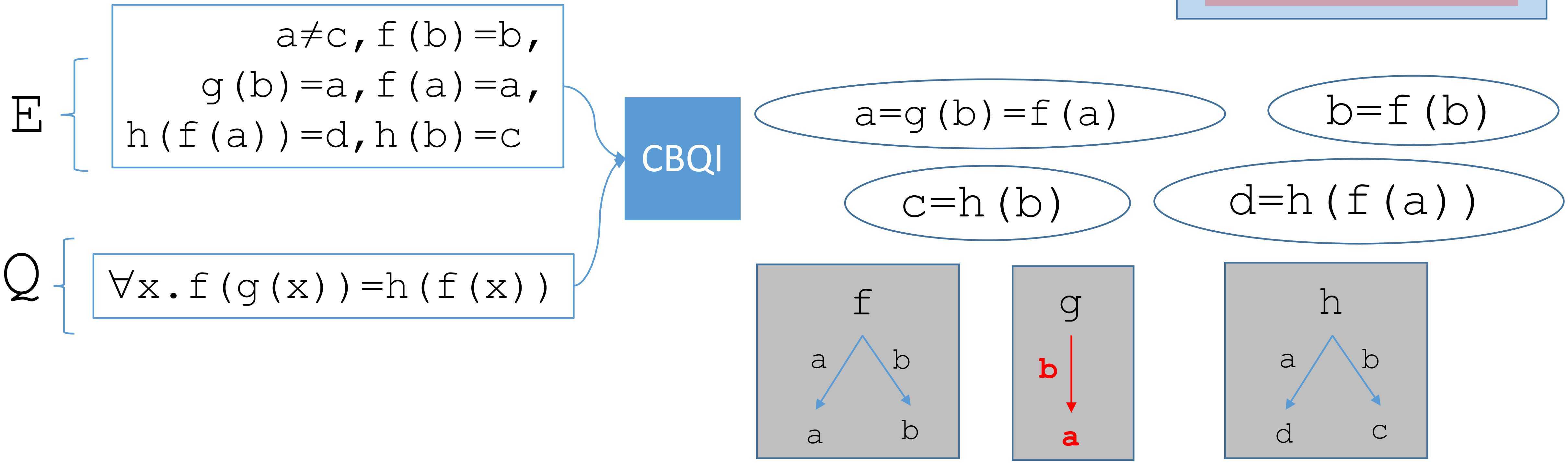
$$E, Q, f(g(b))=h(f(b)) \models_E f(g(b)) = \quad c$$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: EUF



E

a≠c, 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))

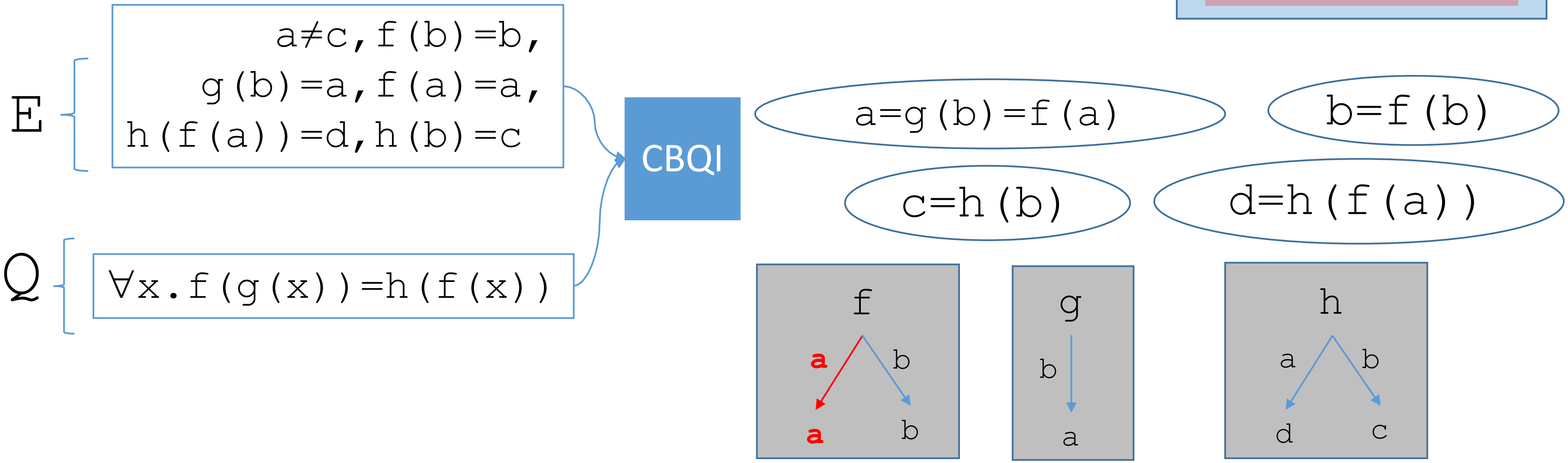
$$E, Q, f(g(b))=h(f(b)) \models_E f(\ \mathbf{a}\ ) = \quad c$$

Conflict-Based Instantiation: EUF

Conflict-Based

E-matching

Model Based



E

a≠c, 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)

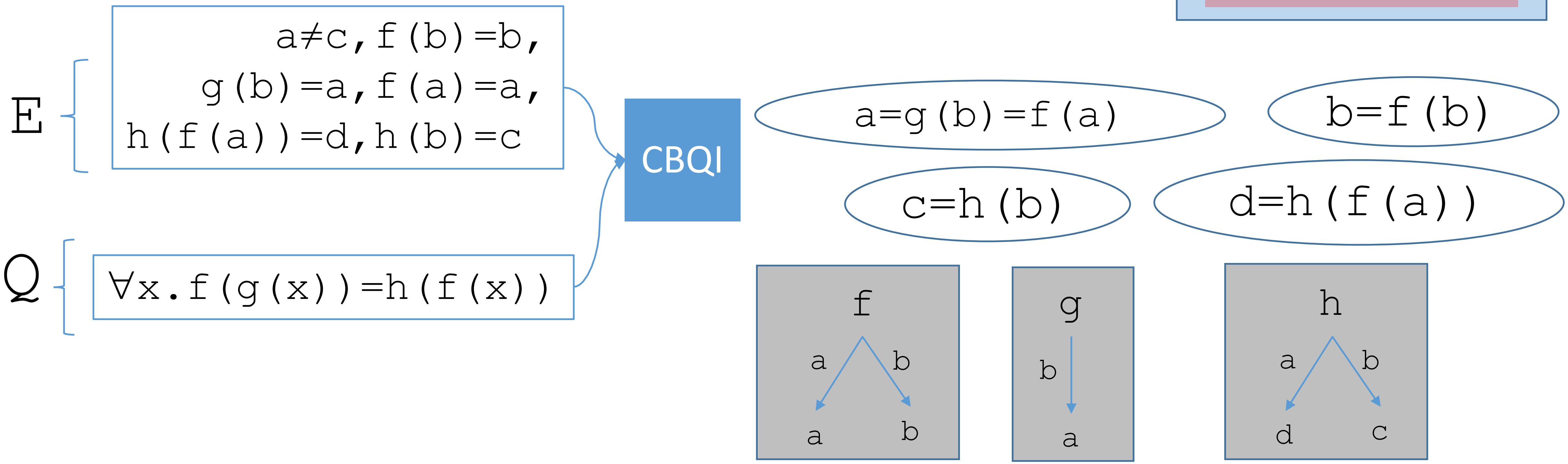
c=h(b)

d=h(f(a))

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

Conflict-Based Instantiation: EUF

- Conflict-Based
- E-matching
- Model Based



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

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

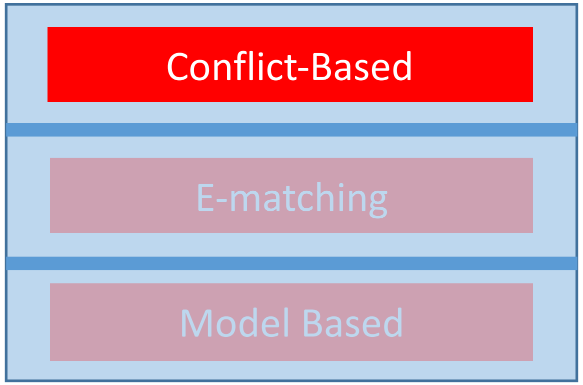
$b=f(b)$

$c=h(b)$

$d=h(f(a))$

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

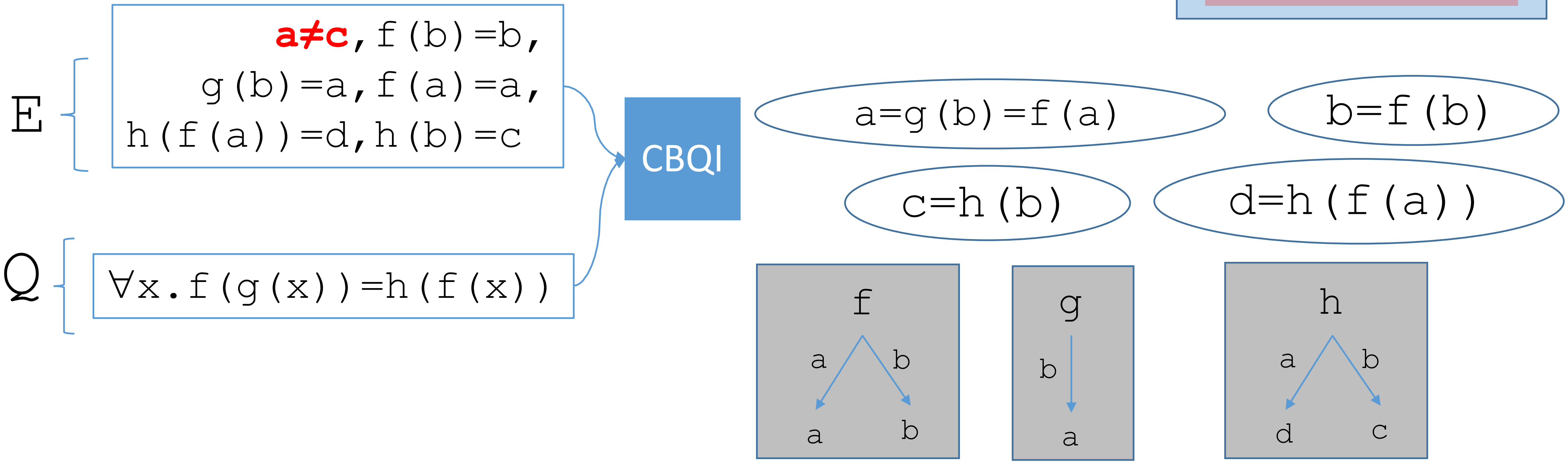
Conflict-Based Instantiation: EUF



Conflict-Based

E-matching

Model Based



E

a≠c, 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)

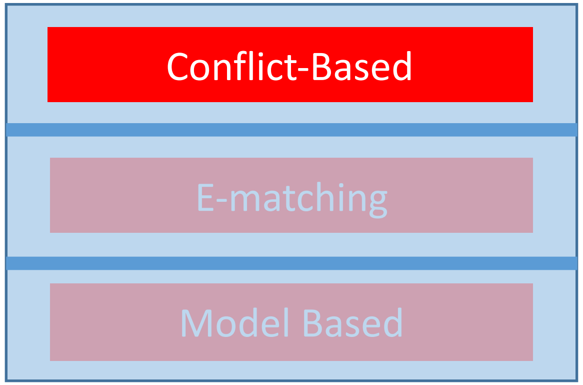
c=h(b)

d=h(f(a))

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

From E, we know **a≠c**

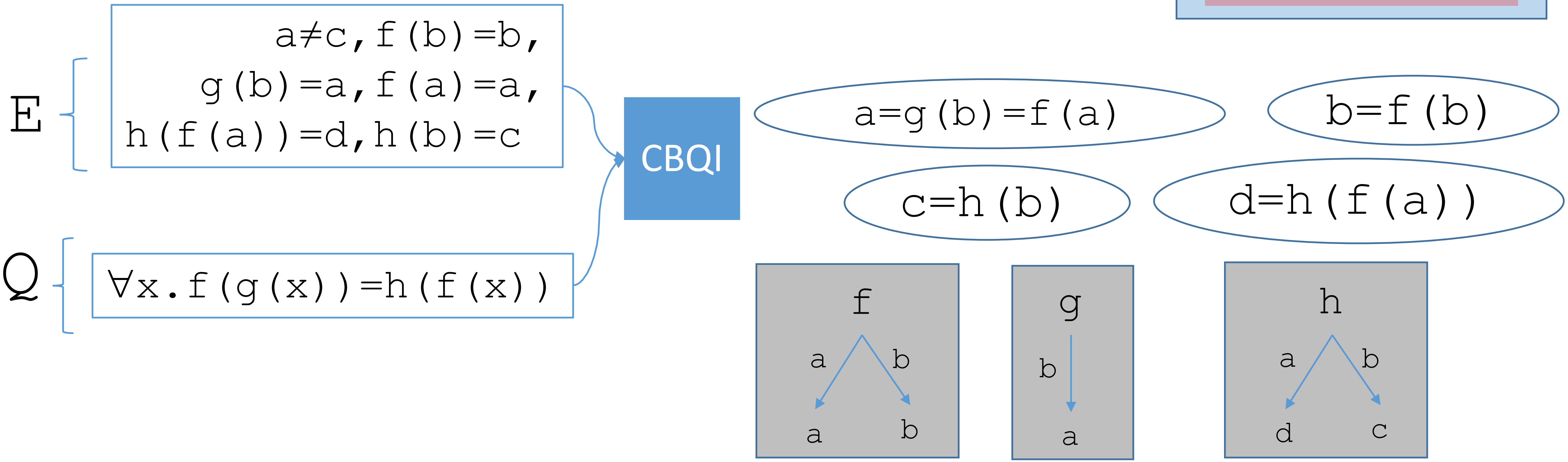
Conflict-Based Instantiation: EUF



Conflict-Based

E-matching

Model Based



E: a≠c, 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))

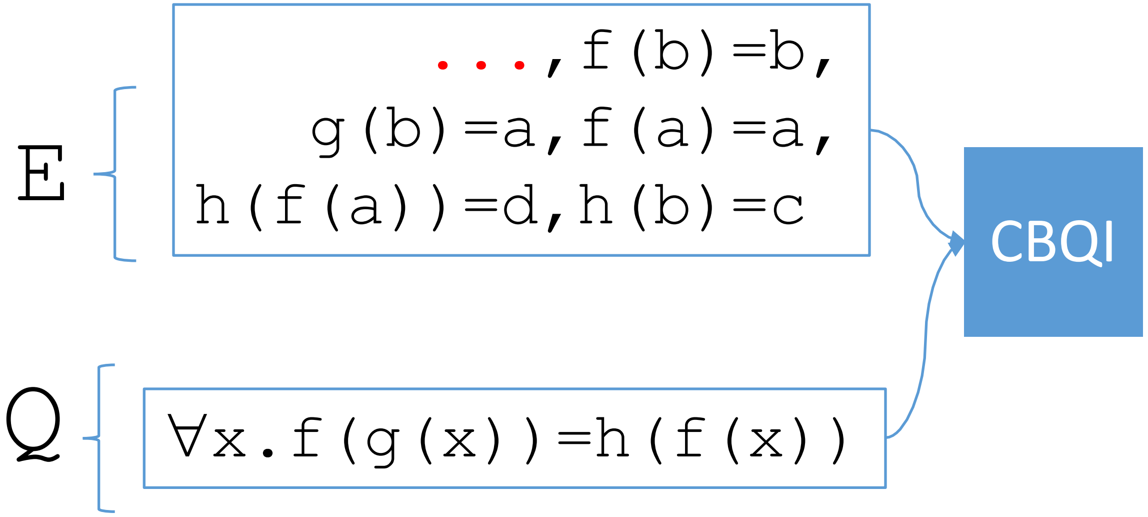
$E, Q, f(g(b))=h(f(b)) \models_E \quad \perp$ } f(g(b))=h(f(b)) is a conflicting instance for (E,Q)!

Conflict-Based

E-matching

Model Based

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

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

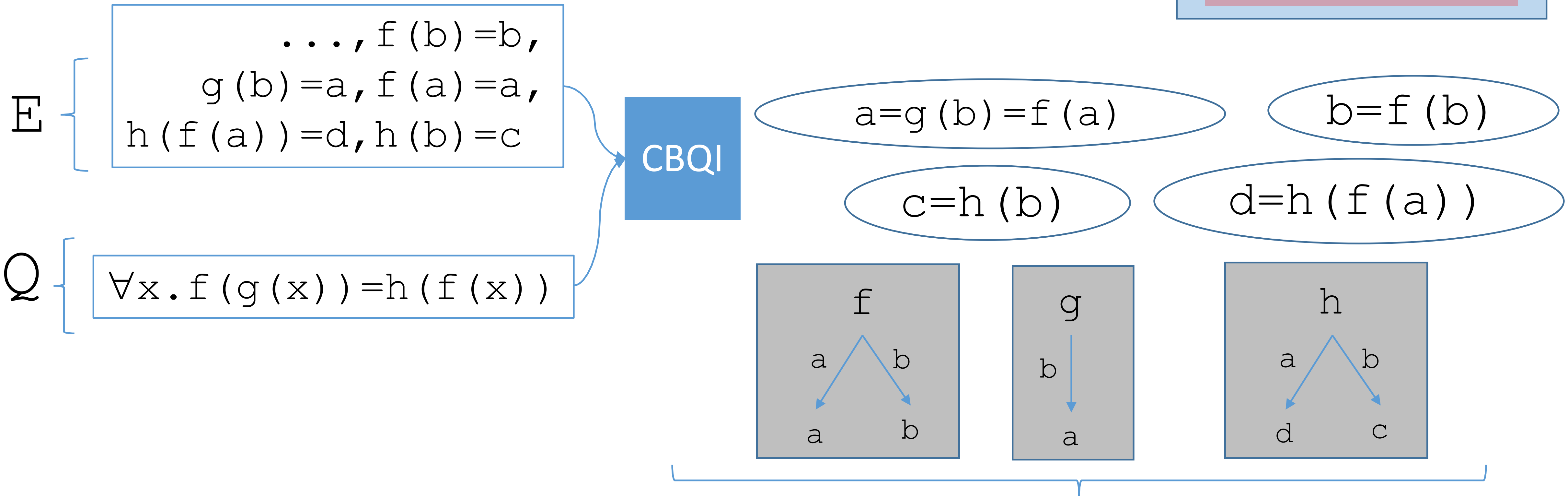
- Is the instance f(g(b))=h(f(b)) still useful?

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: EUF



E

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

Q

`∀x.f(g(x))=h(f(x))`

CBQI

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

`b=f(b)`

`c=h(b)`

`d=h(f(a))`

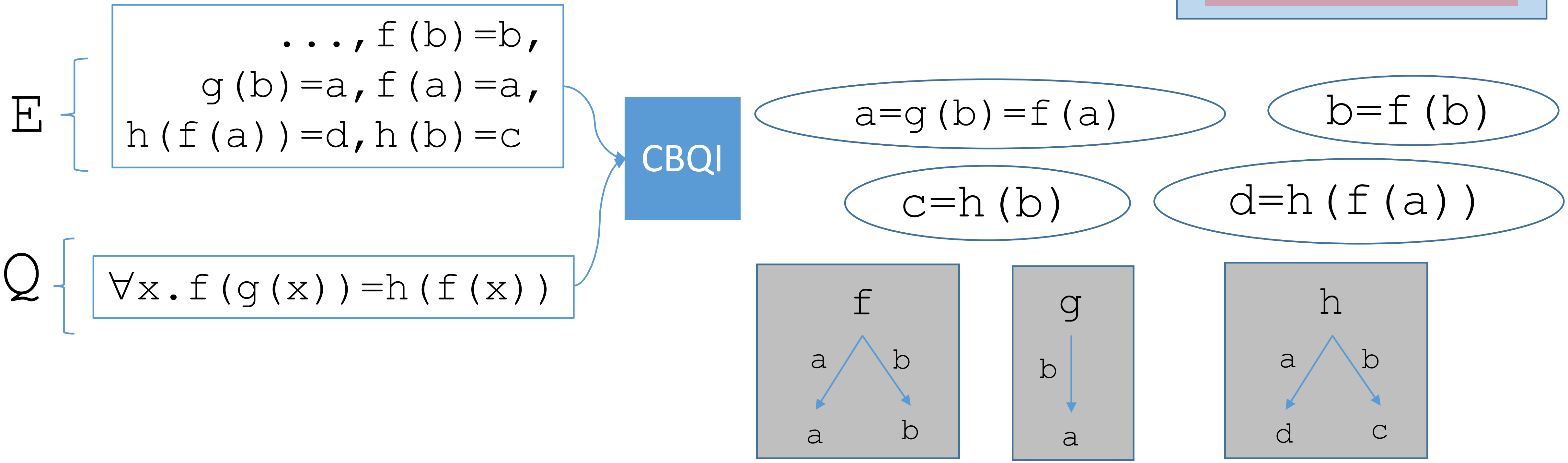
Build partial definitions

Conflict-Based Instantiation: EUF

Conflict-Based

E-matching

Model Based



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))

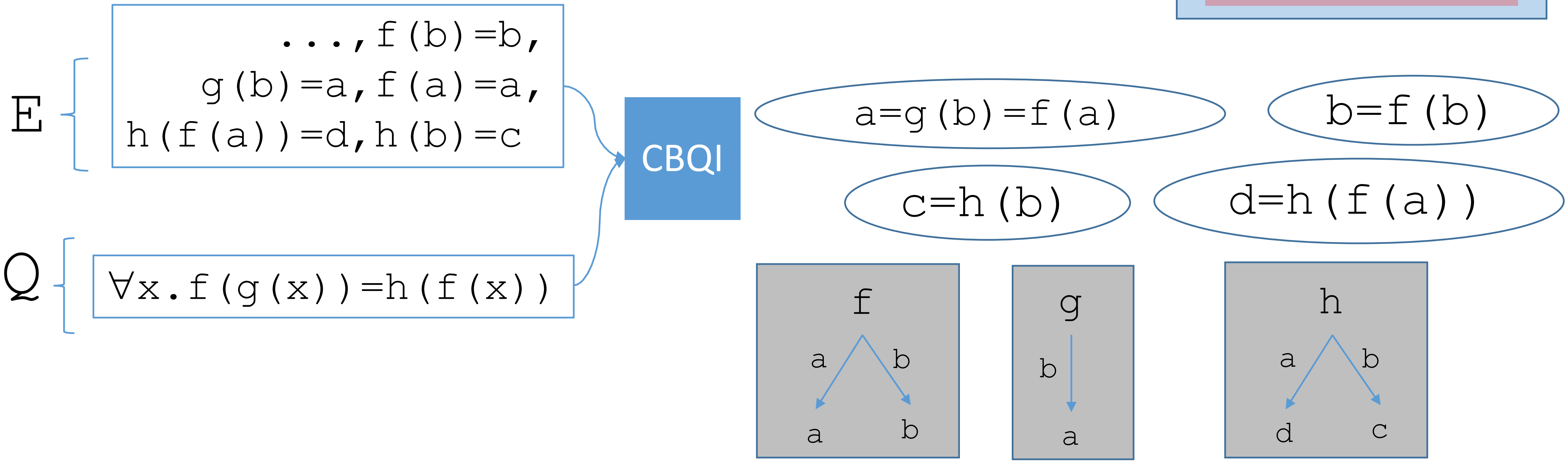
$E, Q, f(g(b))=h(f(b)) \models_E f(g(b))=h(f(b))$ **Check entailment**

Conflict-Based

E-matching

Model Based

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))

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

Conflict-Based Instantiation: EUF

Conflict-Based

E-matching

Model Based

$\mathtt{E}$

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

$\mathtt{Q}$

```
∀x.f(g(x))=h(f(x))
```

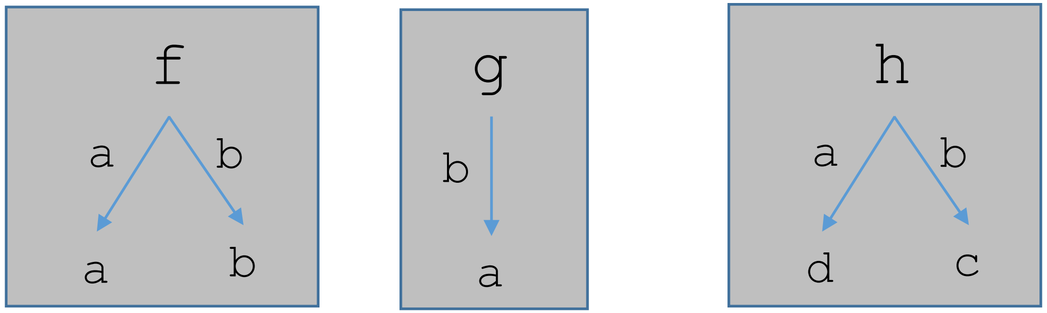
CBQI

a=g(b)=f(a)

b=f(b)

c=h(b)

d=h(f(a))



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

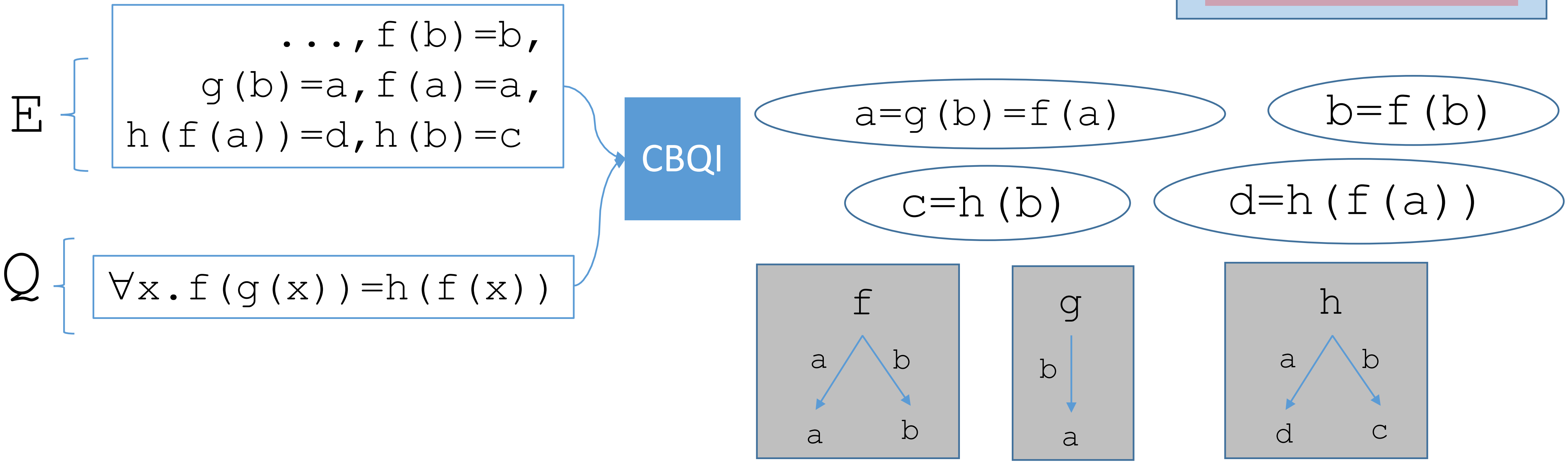
Instance is *not conflicting*,
but *propagates* an equality
between two existing terms in $\mathtt{E}$

Conflict-Based

E-matching

Model Based

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)

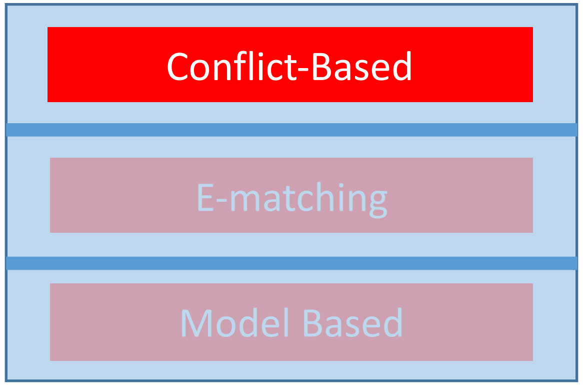
d=h(f(a))

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

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

⇒ *These are also useful*

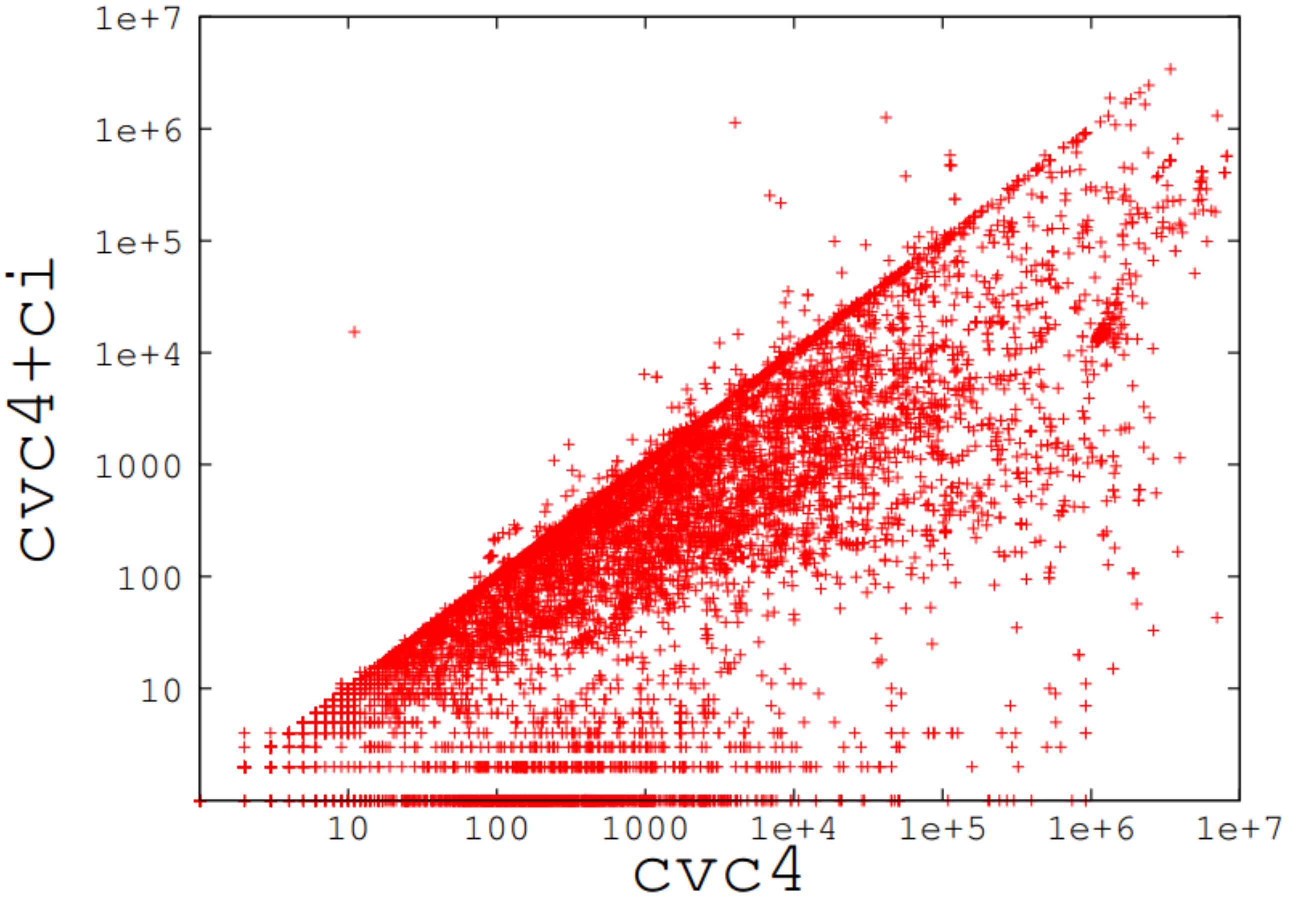
Conflict-Based Instantiation: Impact



Conflict-Based

E-matching

Model Based

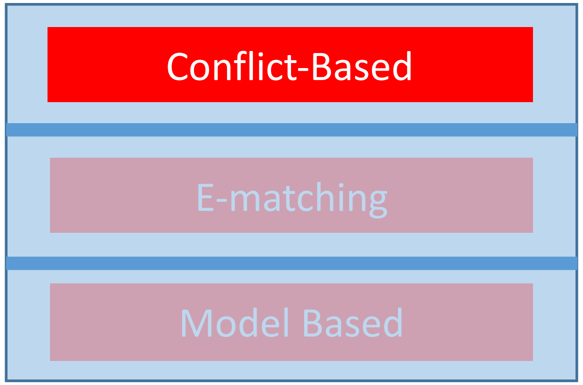


Reported number of instances.

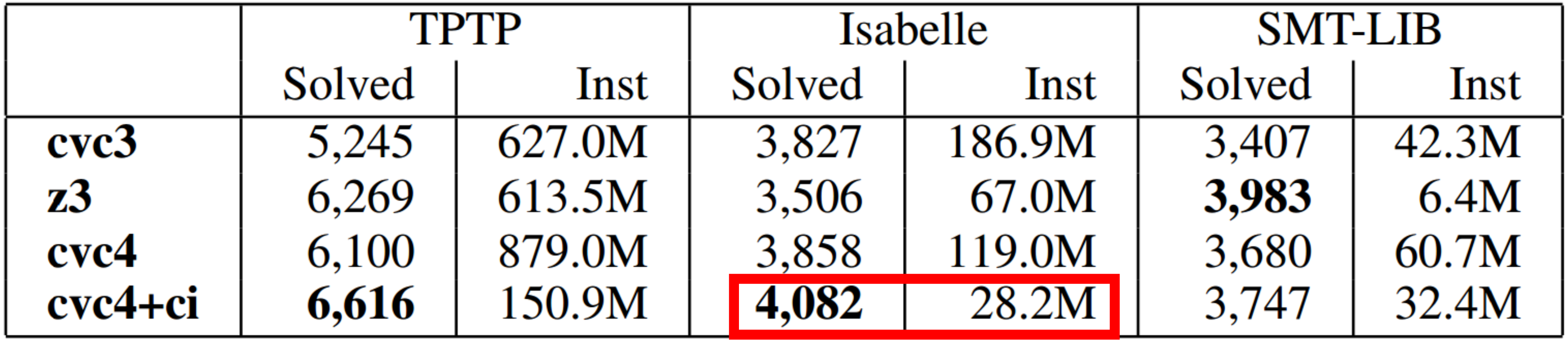
- Using conflict-based instantiation (**`cvc4+ci`**), require an order of magnitude fewer instances for showing “UNSAT” wrt E-matching alone

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

Conflict-Based Instantiation: Impact



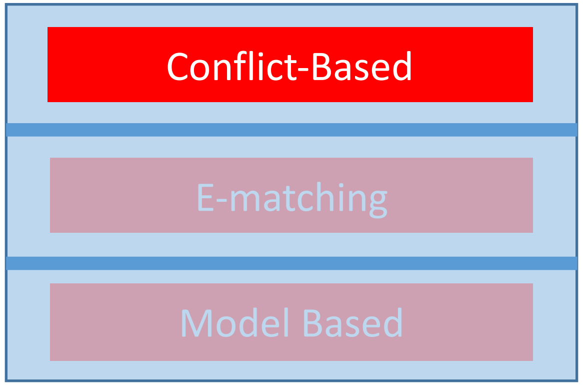
- CVC4 with conflicting instances **`cvc4+ci`**
  - Solves the most benchmarks for TPTP and Isabelle
  - Requires almost an order of magnitude fewer instantiations



	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

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

Conflict-Based Instantiation: Challenges

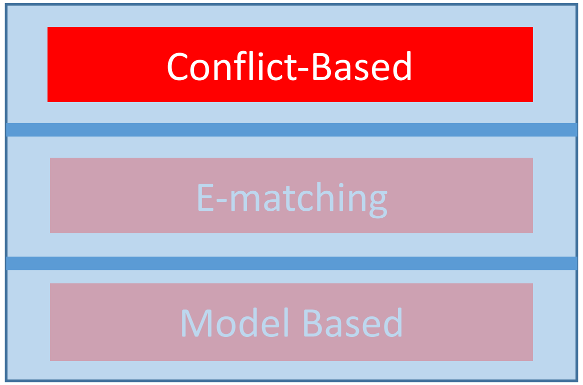


Conflict-Based

E-matching

Model Based

- How do we *find* conflicting instances?
- What about conflicts involving *multiple quantified formulas*?
- What if our quantified formulas that contain *theory symbols*?



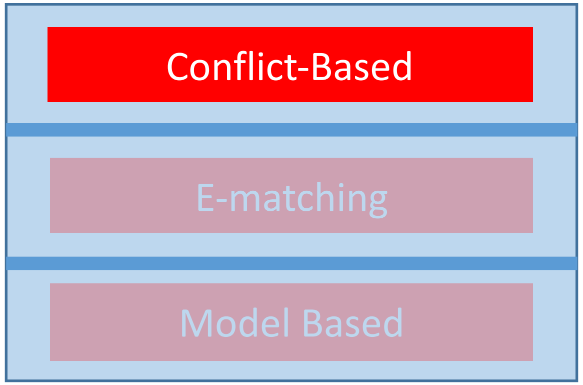
Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: Challenges

- How do we *find* conflicting instances?



Conflict-Based

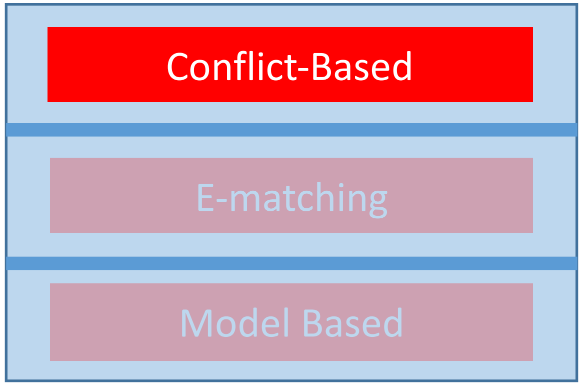
E-matching

Model Based

Conflict-Based Instantiation: Challenges

- How do we *find* conflicting instances?
  - Naively:
    1. Produce all instances Ψ_1, …, Ψ_n via E-matching for ($\mathtt{E}$,$\mathtt{Q}$)
    2. For $\mathtt{i=1}$, …, $\mathtt{n}$, check if $\Psi_{\mathtt{i}}$ is a conflicting instance for ($\mathtt{E}$,$\mathtt{Q}$)

Conflict-Based Instantiation: Challenges



Conflict-Based

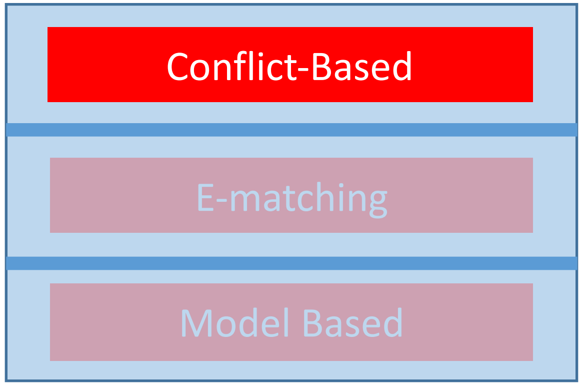
E-matching

Model Based

- How do we *find* conflicting instances?
  - Naively:
    1. Produce all instances Ψ_1, …, Ψ_n via E-matching for ($\mathtt{E}$,$\mathtt{Q}$)
    2. For $\mathtt{i=1,\ldots,n}$, check if $\Psi_{\mathtt{i}}$ is a conflicting instance for ($\mathtt{E}$,$\mathtt{Q}$)

    ⇒ *but n may be very large!*

Conflict-Based Instantiation: Challenges



Conflict-Based

E-matching

Model Based

- How do we *find* conflicting instances?
  - Naively:
    1. Produce all instances Ψ_1, …, Ψ_n via E-matching for ($\mathtt{E}$,$\mathtt{Q}$)
    2. For $\mathtt{i=1}$, …, $\mathtt{n}$, check if $\Psi_{\mathtt{i}}$ is a conflicting instance for ($\mathtt{E}$,$\mathtt{Q}$)
  - In practice: it can be done more efficiently:
    - Basic idea: construct instances via a stronger version of matching
      - Intuition: for $\forall \mathtt{x}.\mathtt{P(x)} \vee \mathtt{Q(x)}$, will *only* match $\mathtt{P(x)}$ with $\mathtt{P(t)} \Leftrightarrow \bot$

        (For technical details, see **[Reynolds et al FMCAD2014]**)

Conflict-Based Instantiation: Challenges

Conflict-Based

E-matching

Model Based

- What about conflicts involving *multiple quantified formulas*?

$$E \left\{ \boxed{\begin{array}{c} P_0(a) \\ \neg P_{100}(a) \end{array}} \right. \qquad Q \left\{ \boxed{\begin{array}{c} \forall x. P_0(x) \Rightarrow P_1(x) \\ \forall x. P_1(x) \Rightarrow P_2(x) \\ \ldots \\ \forall x. P_{99}(x) \Rightarrow P_{100}(x) \end{array}} \right.$$

Conflict-Based

E-matching

Model Based

Conflict-Based Instantiation: Challenges

- What about quantified formulas that contain *theory symbols*?

E `f(1)=5`

Q `∀xy.f(x+y)>x+2*y`

Conflict-Based Instantiation: Challenges

Conflict-Based

E-matching

Model Based

- What about quantified formulas that contain *theory symbols*?

E { `f(1)=5` } Q { `∀xy.f(x+y)>x+2*y` }

- Want to find, e.g.:
  - $E, f(\mathbf{-3}+\mathbf{4})>\mathbf{-3}+2*\mathbf{4} \models_{UFLIA} f(\mathbf{-3}+\mathbf{4})>\mathbf{-3}+2*\mathbf{4}$

Conflict-Based Instantiation: Challenges

Conflict-Based

E-matching

Model Based

- What about quantified formulas that contain *theory symbols*?

$$E \left\{ \boxed{\texttt{f(1)=5}} \right. \qquad Q \left\{ \boxed{\forall \texttt{xy.f(x+y)>x+2*y}} \right.$$

- Want to find, e.g.:
  - $\texttt{E,f(-3+4)>-3+2*4} \models_{UFLIA} \texttt{f(1)>5}$

Conflict-Based Instantiation: Challenges

Conflict-Based

E-matching

Model Based

- What about quantified formulas that contain *theory symbols*?

$$\mathbb{E} \left\{ \boxed{\texttt{f(1)=5}} \right. \qquad \mathbb{Q} \left\{ \boxed{\forall \texttt{xy.f(x+y)>x+2*y}} \right.$$

- Want to find, e.g.:
  - $\mathbb{E}, \texttt{f(-3+4)>-3+2*4} \models_{\text{UFLIA}} \mathbf{5}\texttt{>5}$

By $\mathbb{E}$, we know **f(1)** = **5**

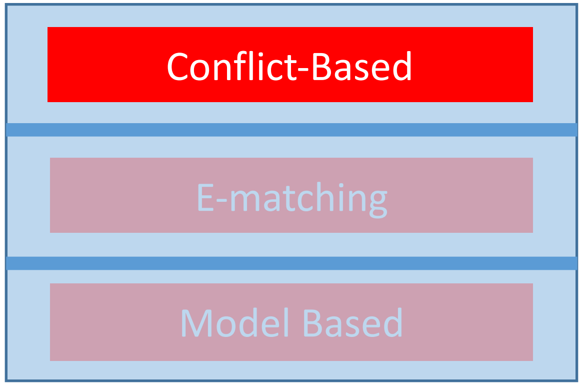
Conflict-Based Instantiation: Challenges

- What about quantified formulas that contain *theory symbols*?

$E \left\{ \boxed{\texttt{f(1)=5}} \right.$ $\qquad$ $Q \left\{ \boxed{\forall \texttt{xy.f(x+y)>x+2*y}} \right.$

- Want to find, e.g.:
  - $\texttt{E,f(-3+4)>-3+2*4} \models_{\text{UFLIA}} \bot$

Conflict-Based Instantiation: Challenges


Conflict-Based

E-matching

Model Based

- What about quantified formulas that contain *theory symbols*?

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

$\mathtt{Q}$ { `∀xy.f(x+y)>x+2*y`

- Want to find, e.g.:
  - `E,f(-3+4)>-3+2*4` $\models_{\textbf{UFLIA}} \bot$

⇒ *In practice, finding such instances cannot be done efficiently*

Conflict-Based Instantiation: Summary

- Instantiation technique for (`E`, `Q`), where:

  ⇒ ***From Q, derive conflicts ⊥, and***

  equalities $g_1=g_2$ between ground terms g_1, g_2 from E

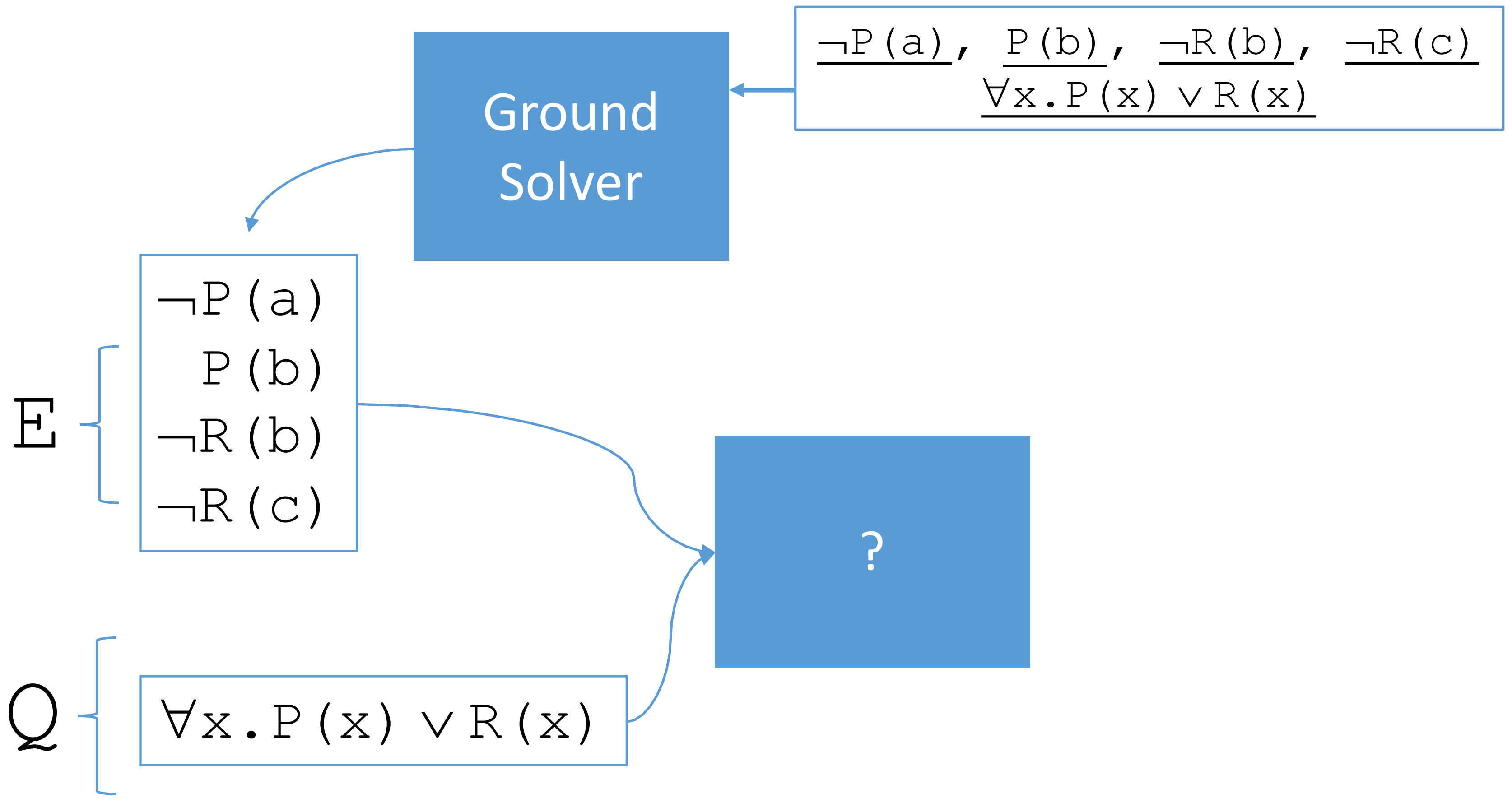
- Run with higher priority to E-matching
  - Resort to E-matching only if no conflicting or propagating instances can be found
- Leads to fewer instances, greater ability to answer “unsat”

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



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

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

Ground Solver

E

$\neg P(a)$

$P(b)$

$\neg R(b)$

$\neg R(c)$

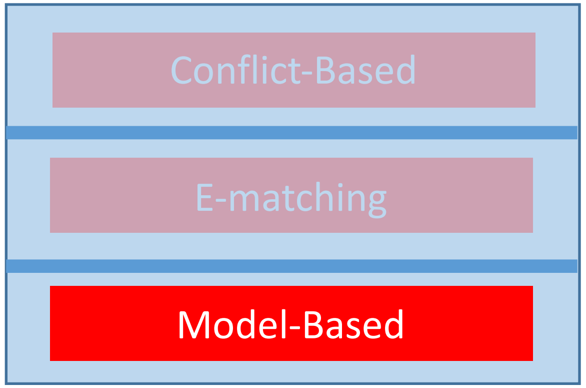
Q

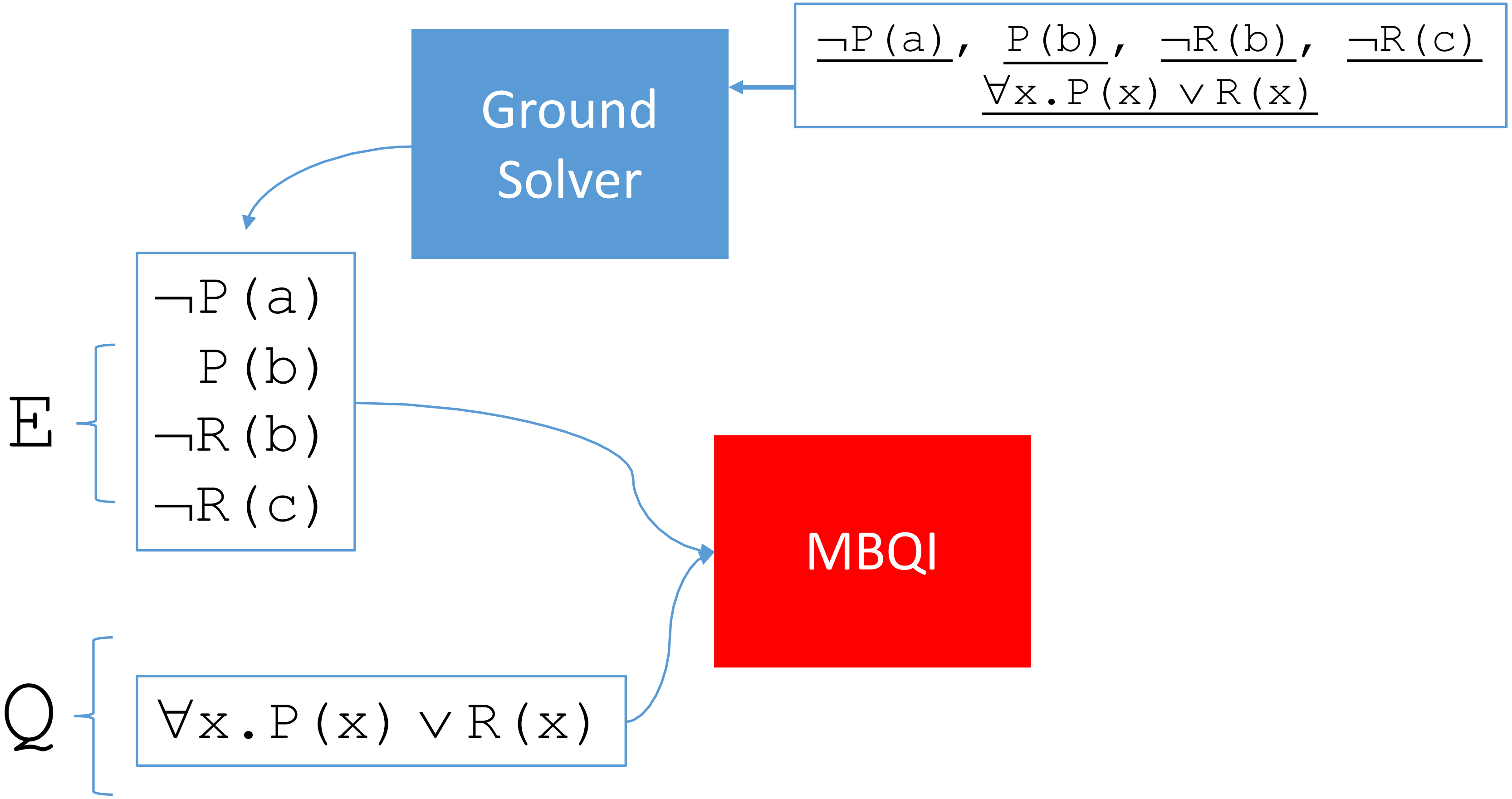
$\forall x.P(x) \vee R(x)$

?

$\Rightarrow$ What if $E \cup Q$ is satisfiable?

Model-based Instantiation




$\underline{\neg P(a)}$, $\underline{P(b)}$, $\underline{\neg R(b)}$, $\underline{\neg R(c)}$
$\underline{\forall x. P(x) \vee R(x)}$
Ground Solver
$\neg P(a)$
$P(b)$
$\neg R(b)$
$\neg R(c)$
E
MBQI
Q
$\forall x. P(x) \vee R(x)$

⇒ What if $E \cup Q$ is satisfiable?

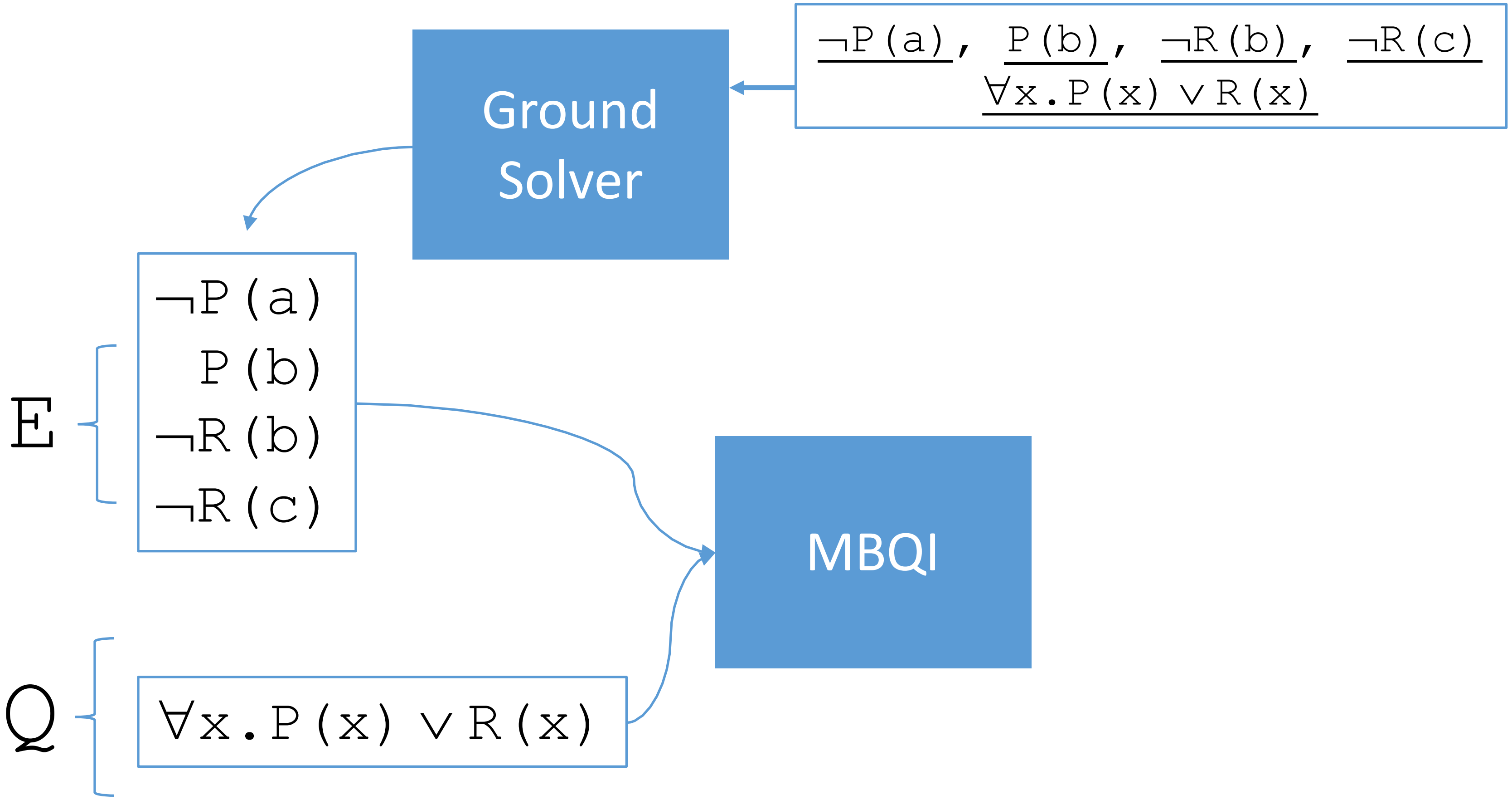
- Use model-based quantifier instantiation (MBQI)

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



$\underline{\neg P(a)}$, $\underline{P(b)}$, $\underline{\neg R(b)}$, $\underline{\neg R(c)}$
$\underline{\forall x. P(x) \lor R(x)}$

Ground Solver

E

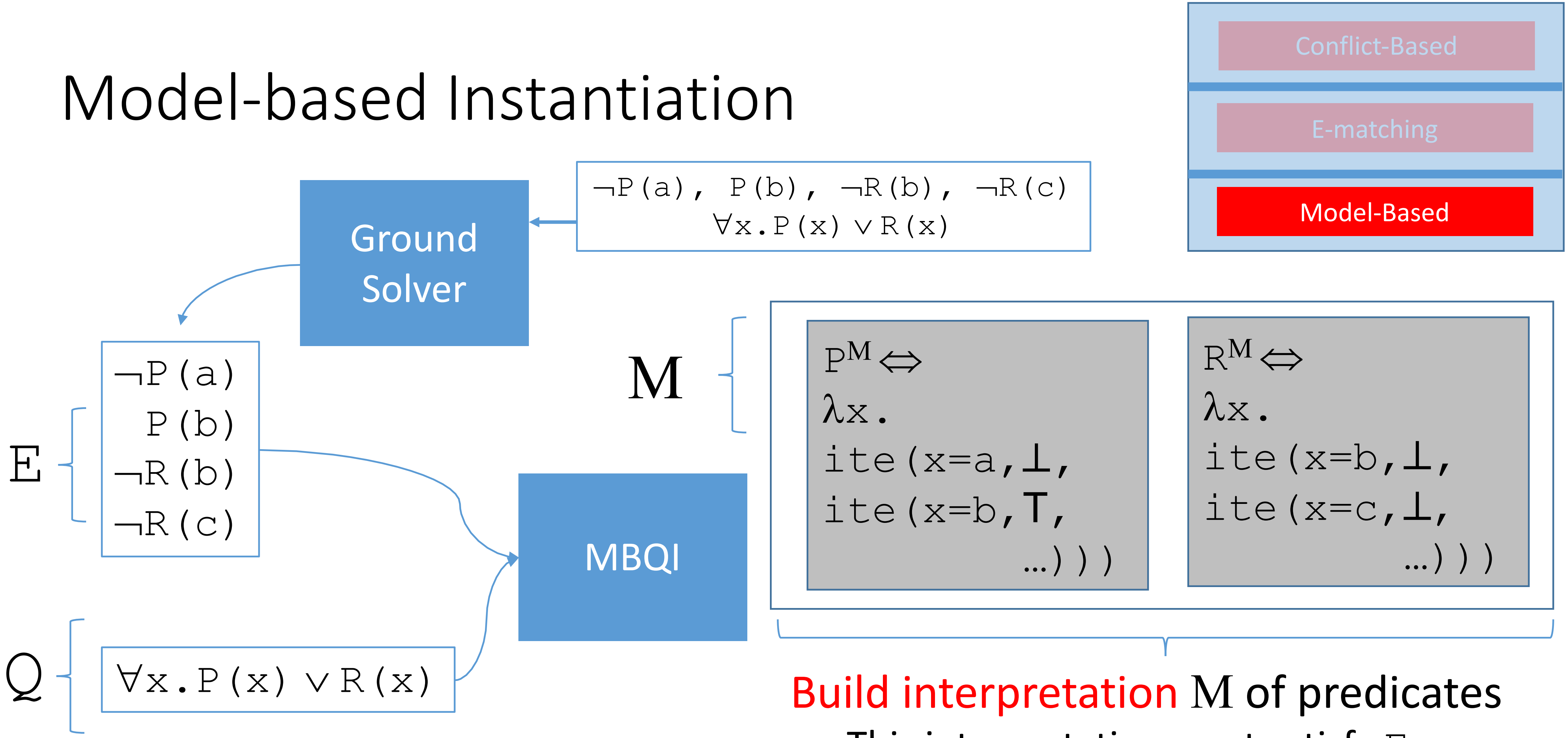
$\neg P(a)$
$P(b)$
$\neg R(b)$
$\neg R(c)$

MBQI

Q

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

Model-based Instantiation



Conflict-Based

E-matching

Model-Based

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

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

Ground Solver

E

$\neg P(a)$

$P(b)$

$\neg R(b)$

$\neg R(c)$

Q

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

MBQI

M

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

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

Build interpretation M of predicates

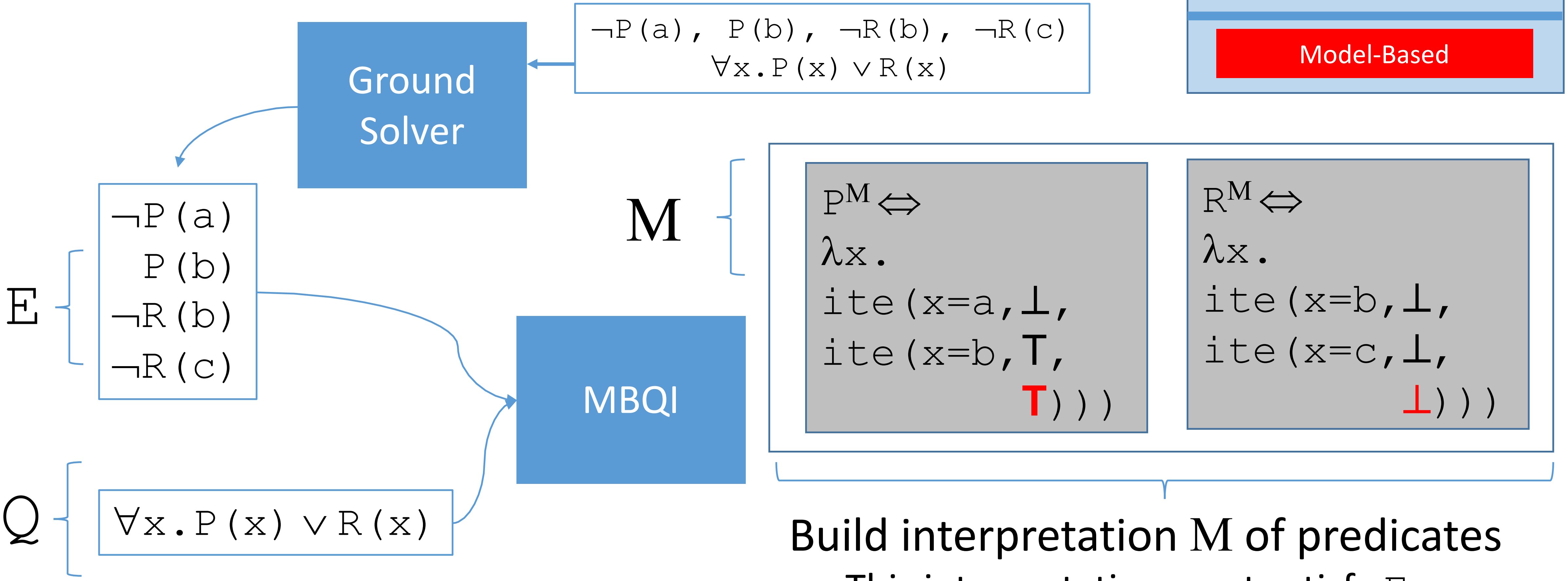
- This interpretation must satisfy E

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



$\neg P(a),\ P(b),\ \neg R(b),\ \neg R(c)$
$\forall x. P(x) \vee R(x)$

Ground Solver

E

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

Q

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

MBQI

M

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

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

Build interpretation M of predicates

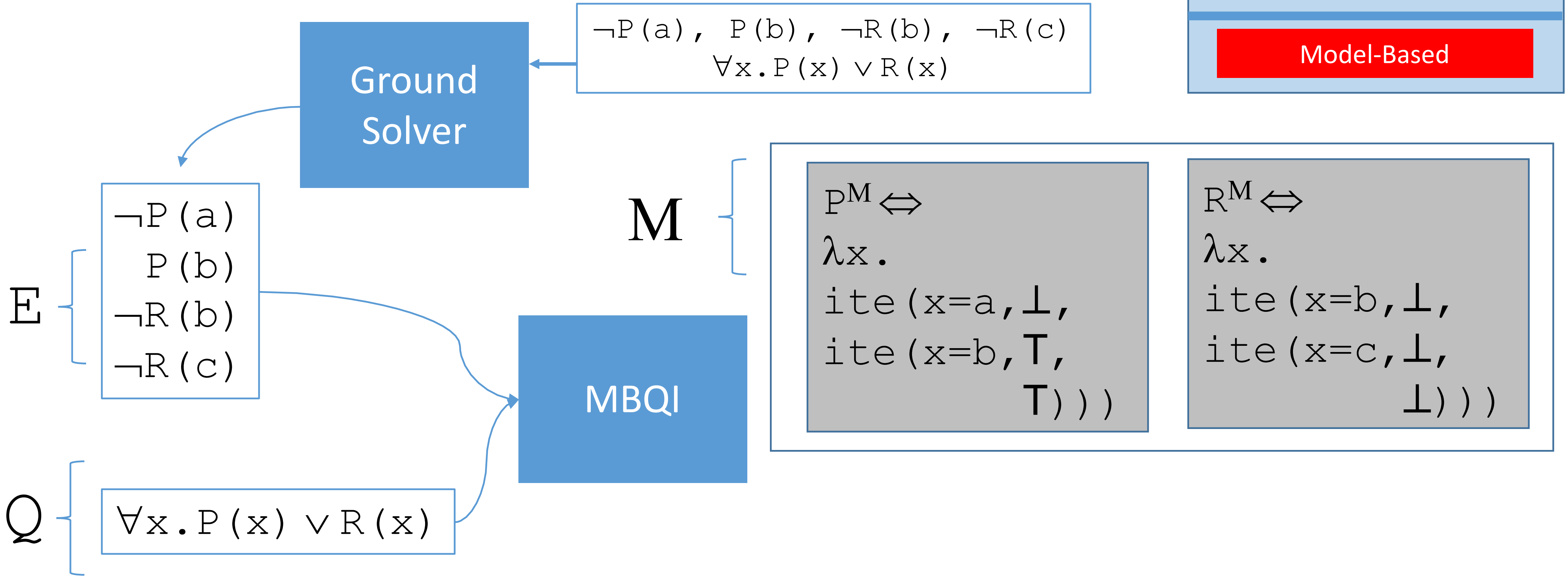
- This interpretation must satisfy E
- Missing values may be filled in arbitrarily

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



$\neg P(a),\ P(b),\ \neg R(b),\ \neg R(c)$
$\forall x.P(x) \vee R(x)$

Ground Solver

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

MBQI

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

M:

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

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

$\Rightarrow$ Does M satisfy Q?

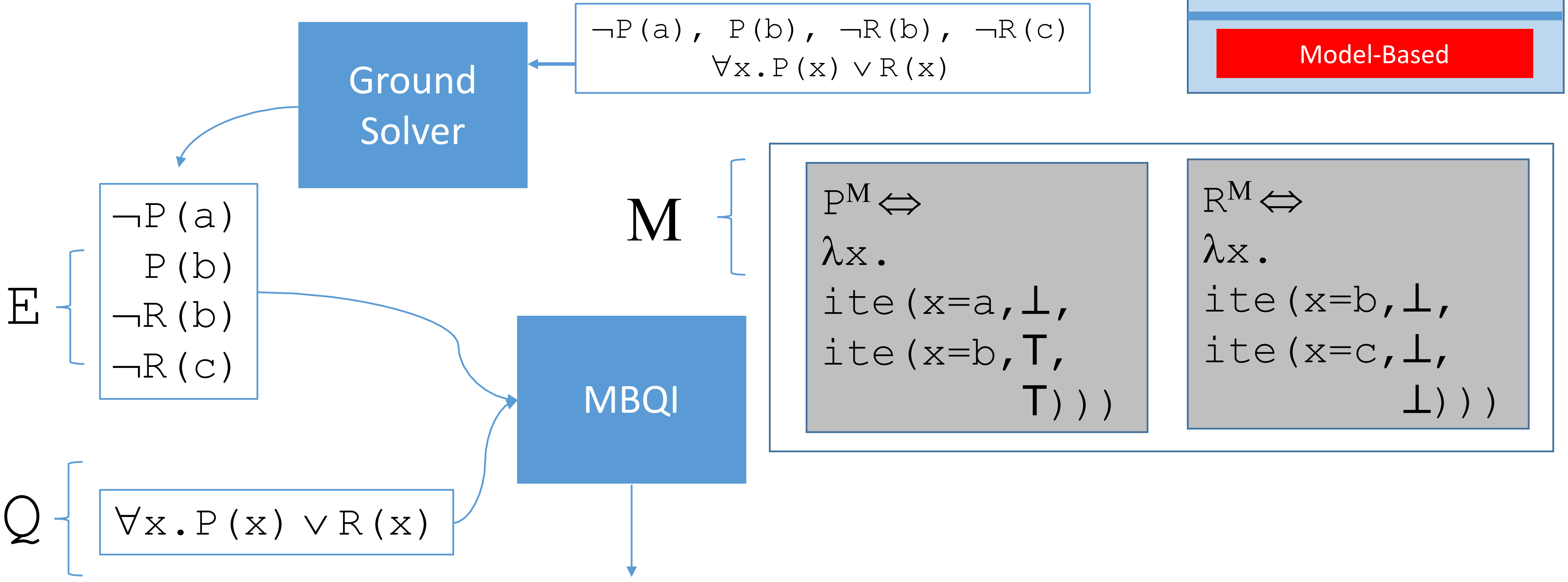
- Check (un)satisfiability of: $\exists x.\neg(P^M(x) \vee R^M(x))$

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



$\neg P(a),\ P(b),\ \neg R(b),\ \neg R(c)$
$\forall x. P(x) \vee R(x)$

Ground Solver

E

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

Q

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

MBQI

M

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

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

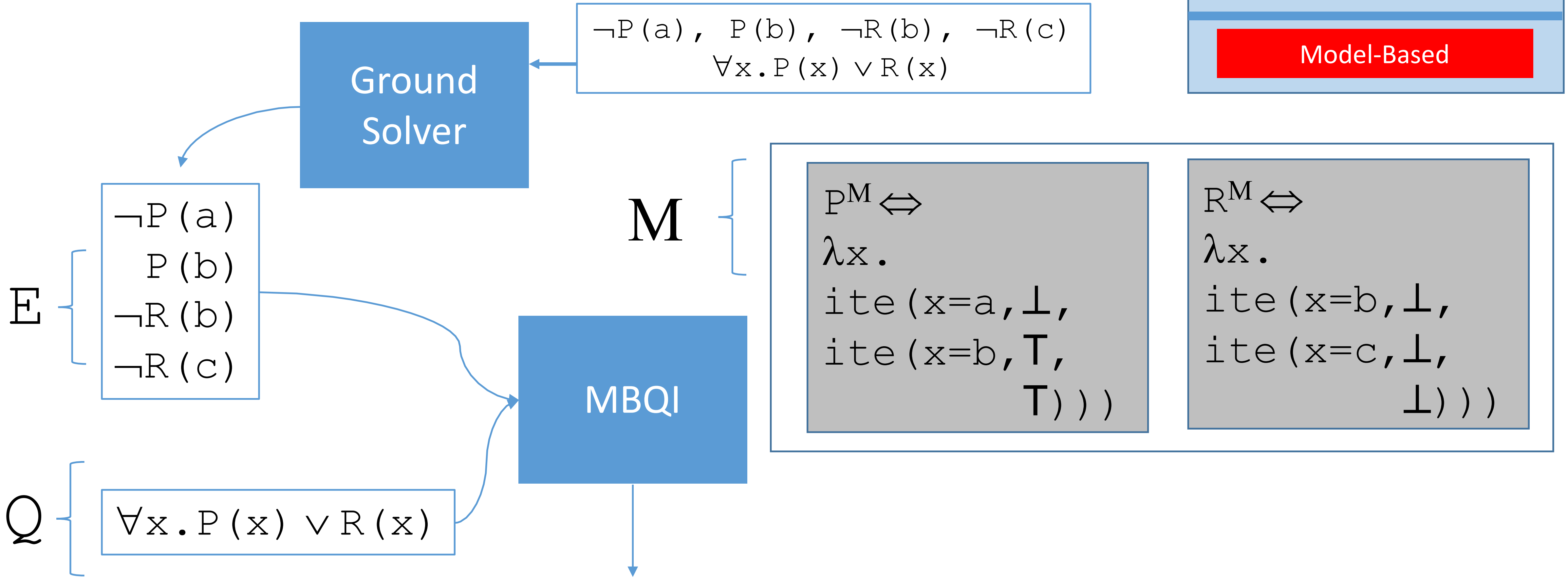
Check: $\exists x. \neg (P^M (x) \vee R^M (x))$

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



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

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

Ground Solver

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

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

MBQI

M

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

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

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

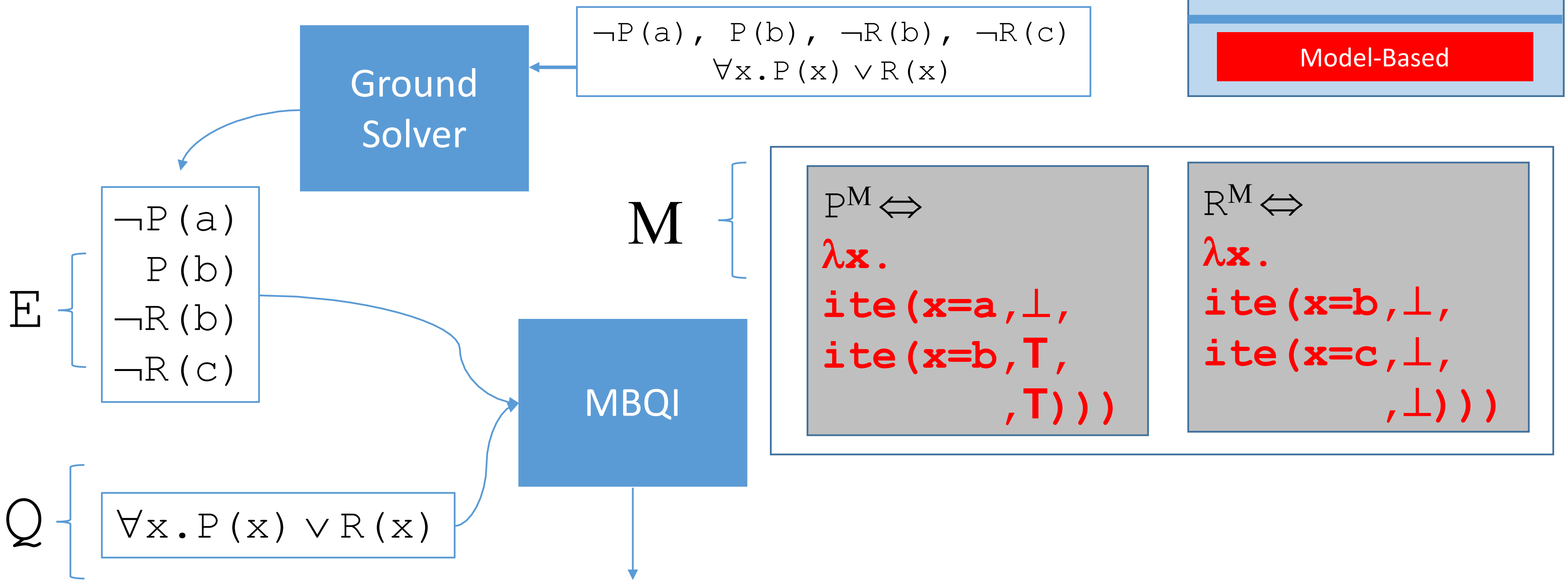
$\Rightarrow$ Skolemize

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



Ground Solver

$\neg P(a),\ P(b),\ \neg R(b),\ \neg R(c)$
$\forall x.P(x) \vee R(x)$

E

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

Q

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

MBQI

M

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

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

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

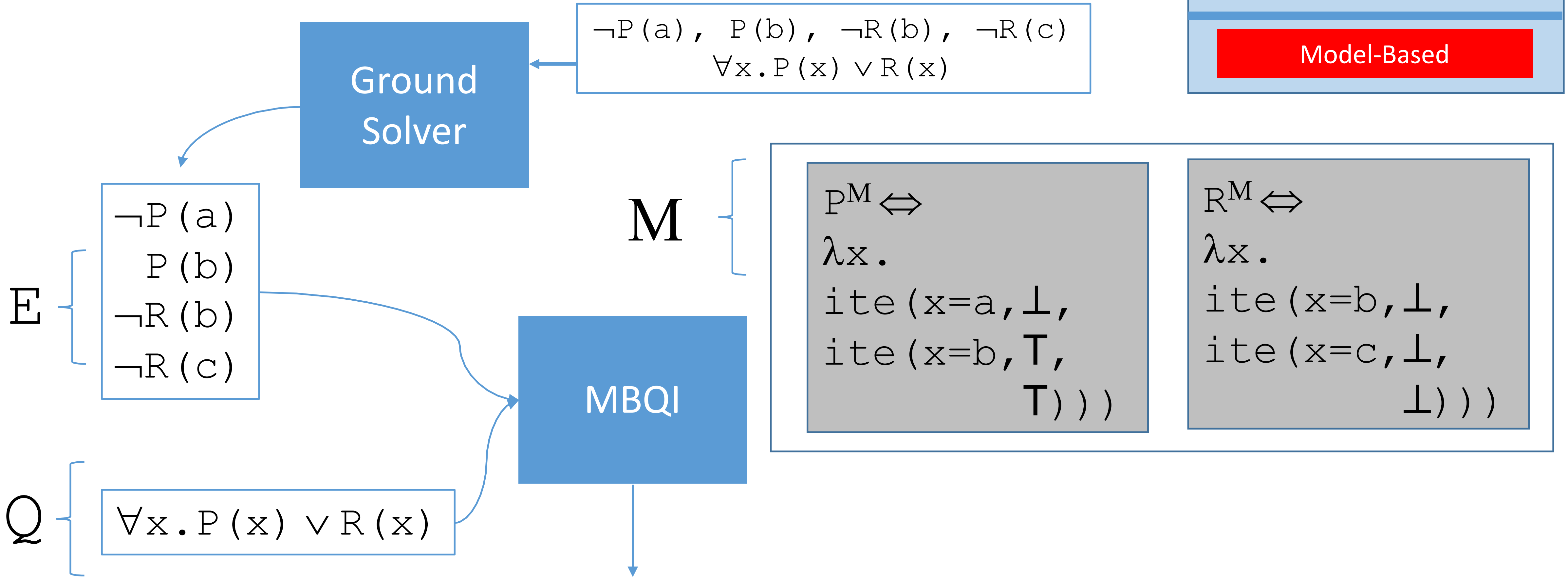
$\Rightarrow$ Substitute

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



¬P(a), P(b), ¬R(b), ¬R(c)
∀x.P(x) ∨ R(x)

Ground Solver

E
¬P(a)
P(b)
¬R(b)
¬R(c)

MBQI

Q
∀x.P(x) ∨ R(x)

M

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

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

Check: ¬(k≠a ∨ ⊥)

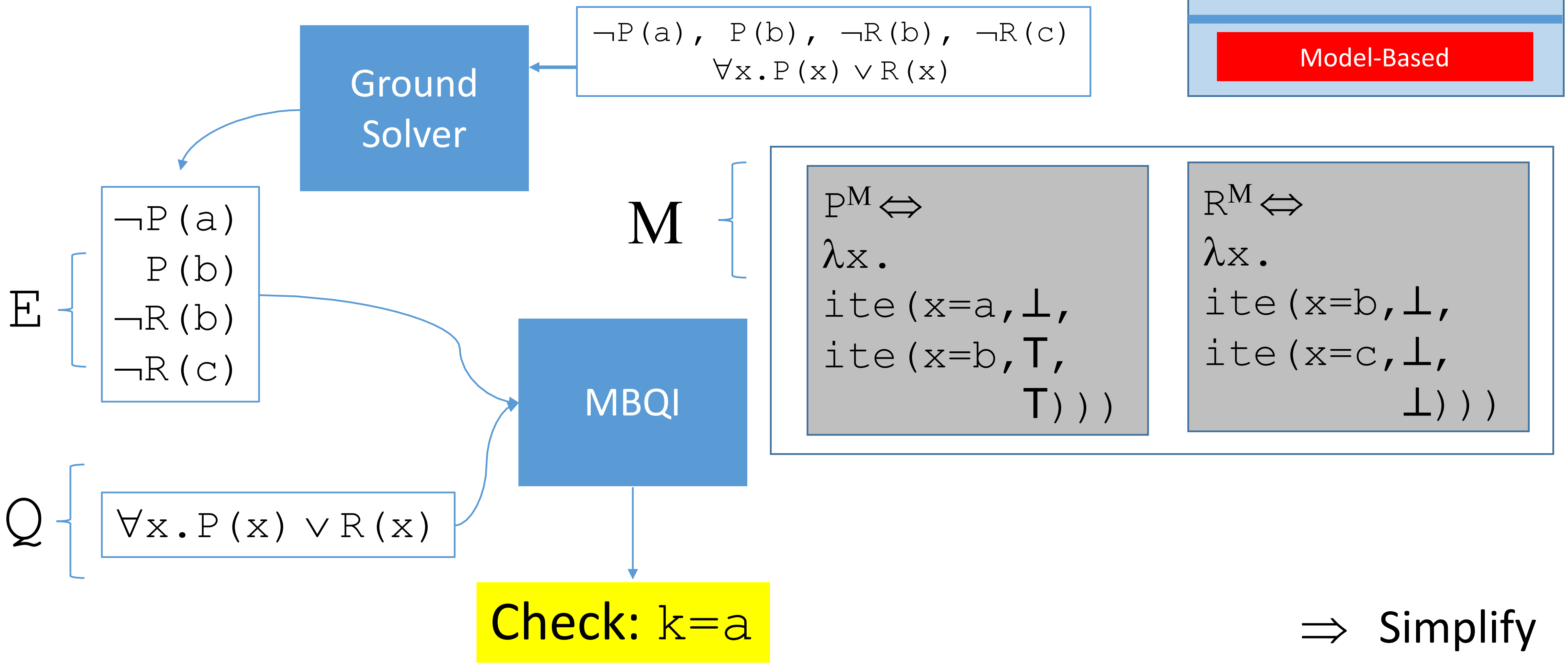
⇒ Simplify

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



$\neg P(a),\ P(b),\ \neg R(b),\ \neg R(c)$
$\forall x. P(x) \vee R(x)$

Ground Solver

E

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

Q

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

MBQI

M

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

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

Check: $k=a$

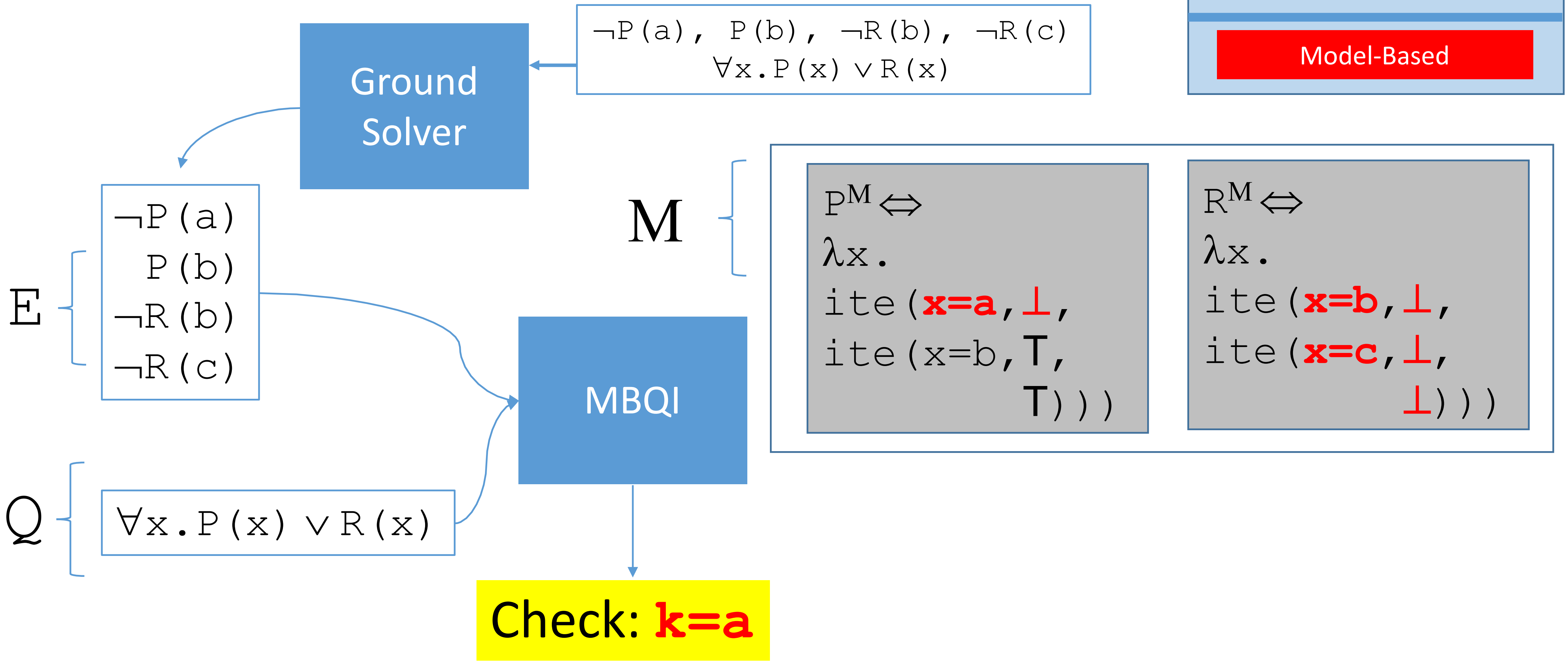
⇒ Simplify

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



¬P(a), P(b), ¬R(b), ¬R(c)
∀x.P(x) ∨ R(x)

Ground Solver

E

¬P(a)
P(b)
¬R(b)
¬R(c)

Q

∀x.P(x) ∨ R(x)

MBQI

M

```
P^M ⇔
λx.
ite(x=a,⊥,
ite(x=b,T,
          T)))
```

```
R^M ⇔
λx.
ite(x=b,⊥,
ite(x=c,⊥,
          ⊥)))
```

Check: **k=a**

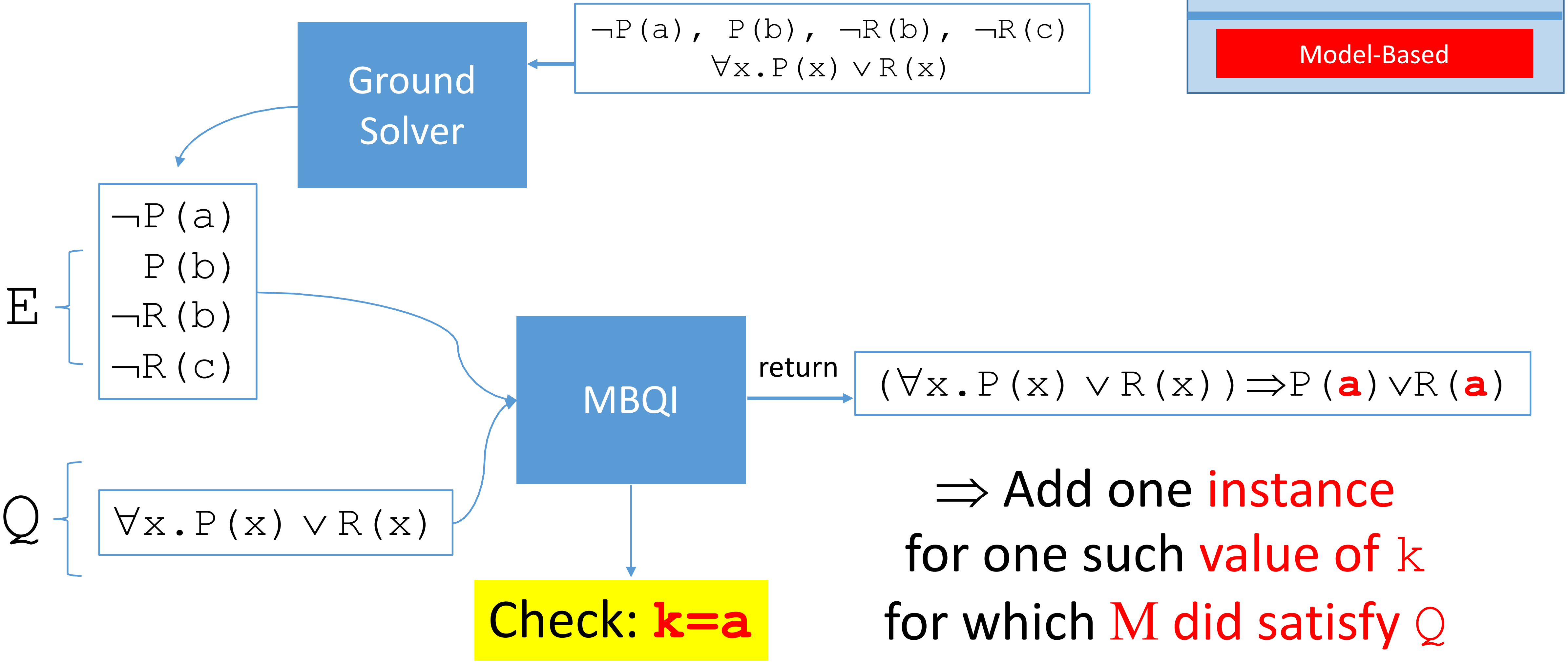
⇒ *Satisfiable! There are values k for which M does not satisfy Q*

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



Ground Solver
$\neg P(a),\ P(b),\ \neg R(b),\ \neg R(c)$
$\forall x.P(x) \lor R(x)$
E
$\neg P(a)$
$P(b)$
$\neg R(b)$
$\neg R(c)$
Q
$\forall x.P(x) \lor R(x)$
MBQI
return
$(\forall x.P(x) \lor R(x)) \Rightarrow P(\mathbf{a}) \lor R(\mathbf{a})$
Check: **k=a**

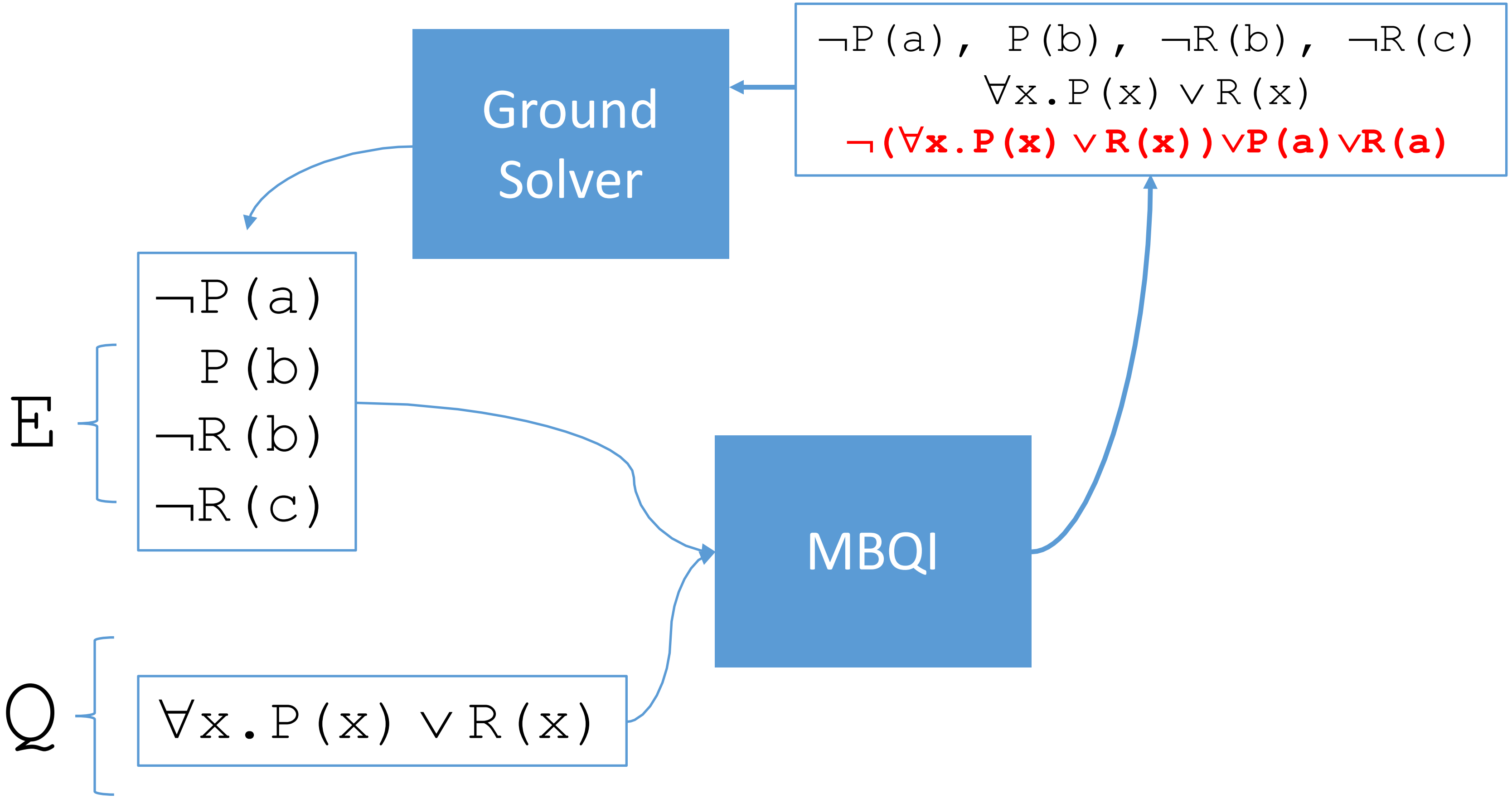
$\Rightarrow$ Add one instance
for one such value of k
for which M did satisfy Q

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



Ground Solver

¬P(a), P(b), ¬R(b), ¬R(c)

$\forall$x.P(x) ∨ R(x)

¬($\forall$x.P(x) ∨ R(x))∨P(a)∨R(a)

E

¬P(a)

P(b)

¬R(b)

¬R(c)

MBQI

Q

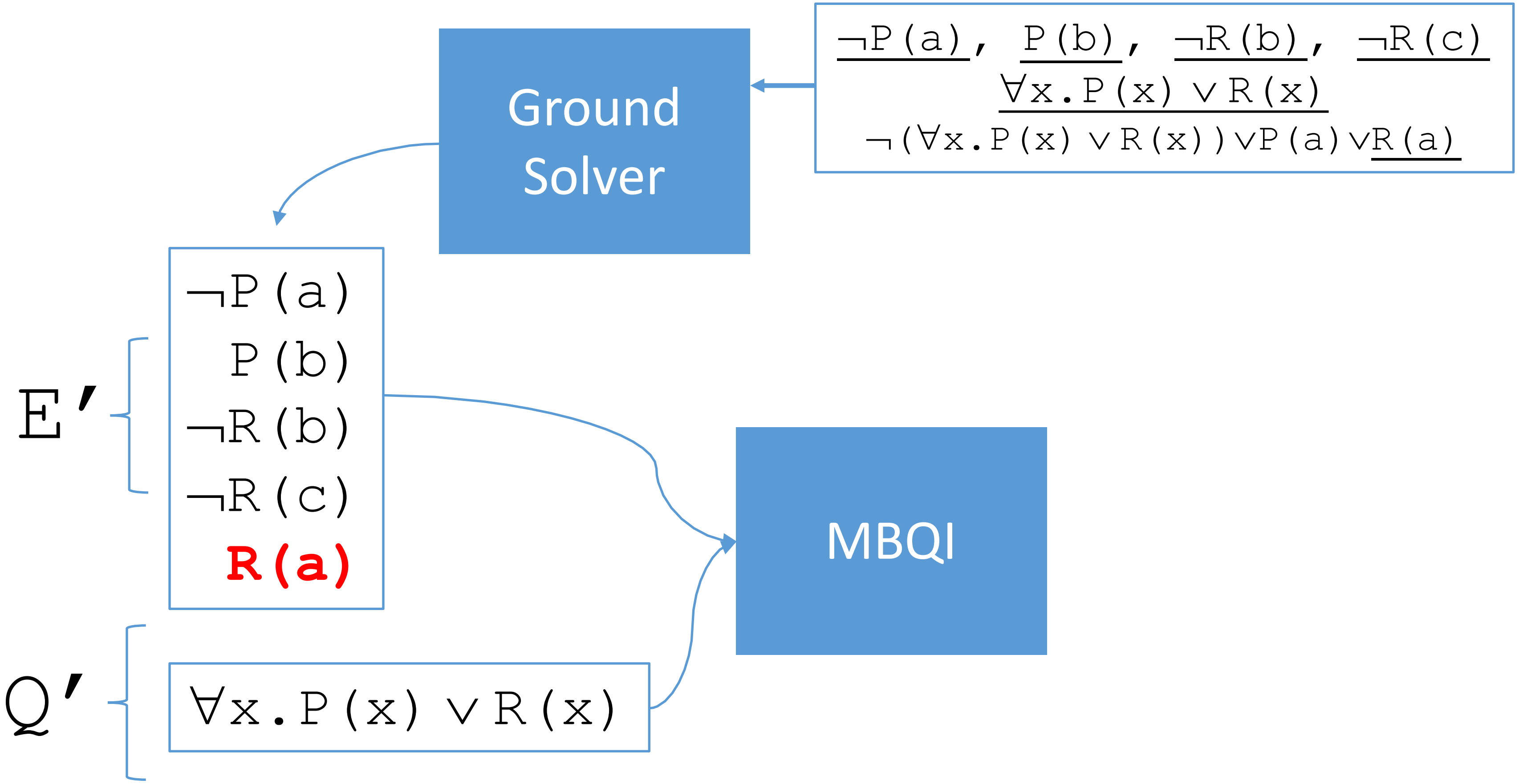
$\forall$x.P(x) ∨ R(x)

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



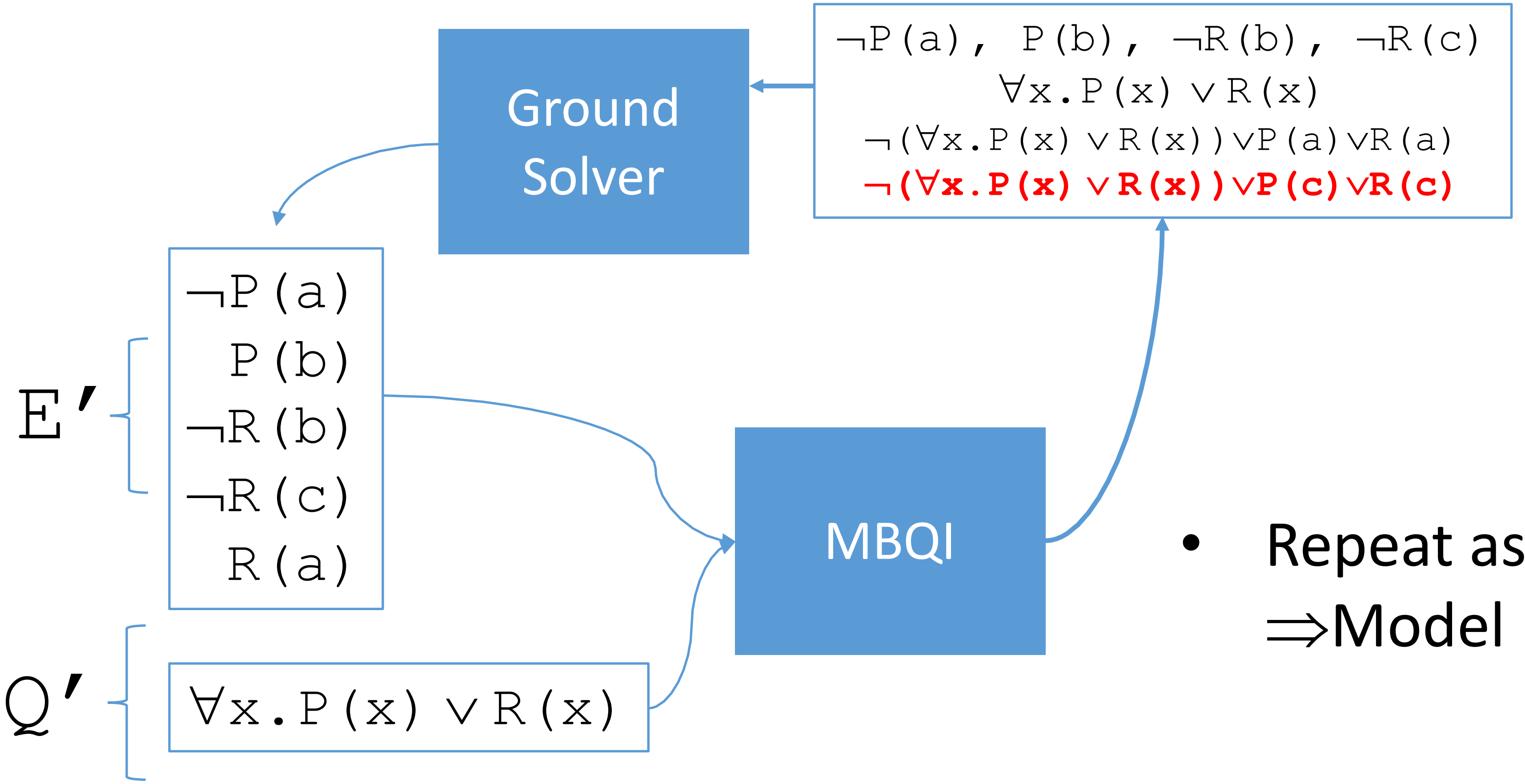
⇒ Subsequent models must satisfy P(x) ∨ R(x) for x→a

Conflict-Based

E-matching

Model-Based

Model-based Instantiation



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

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

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

$\boldsymbol{\neg(\forall x.P(x) \vee R(x)) \vee P(c) \vee R(c)}$

Ground Solver

E'

$\neg P(a)$

$P(b)$

$\neg R(b)$

$\neg R(c)$

$R(a)$

MBQI

Q'

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

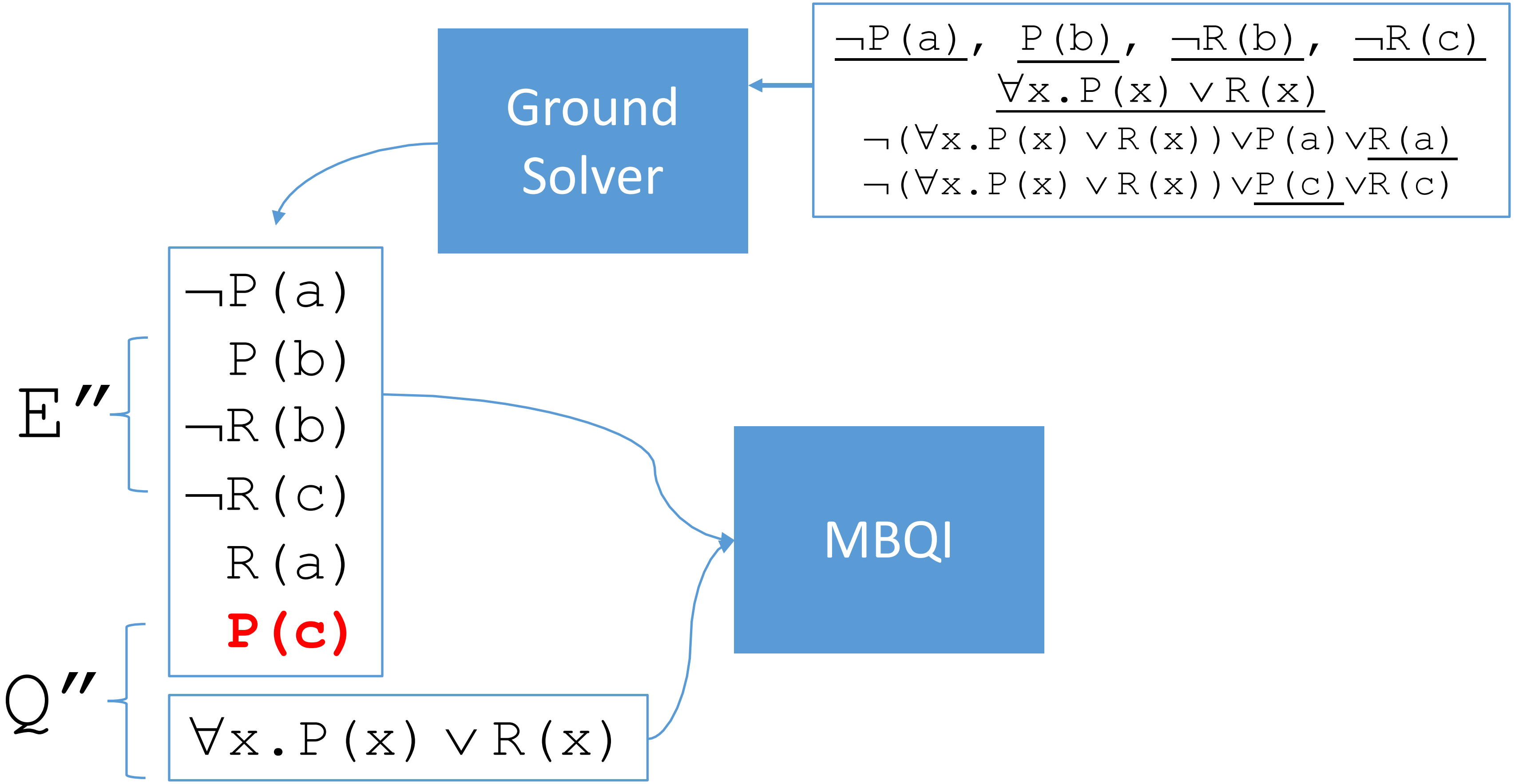
- Repeat as necessary
  ⇒Model refinement loop

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



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

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

$\neg(\forall x. P(x) \vee R(x)) \vee P(a) \vee \underline{R(a)}$

$\neg(\forall x. P(x) \vee R(x)) \vee \underline{P(c)} \vee R(c)$

Ground Solver

$\neg P(a)$

$P(b)$

$\neg R(b)$

$\neg R(c)$

$R(a)$

$\mathbf{P(c)}$

E''

Q''

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

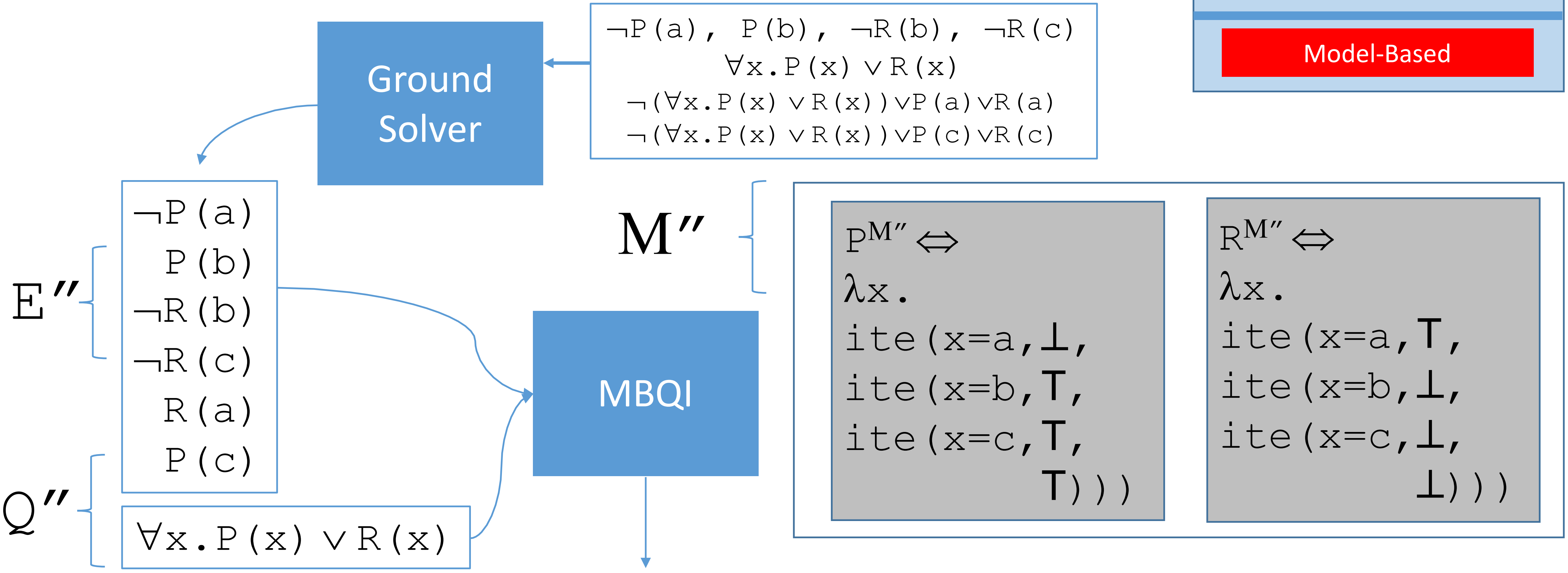
MBQI

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



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

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

$\neg(\forall x. P(x) \lor R(x)) \lor P(a) \lor R(a)$

$\neg(\forall x. P(x) \lor R(x)) \lor P(c) \lor R(c)$

$\neg P(a)$

$P(b)$

$\neg R(b)$

$\neg R(c)$

$R(a)$

$P(c)$

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

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

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

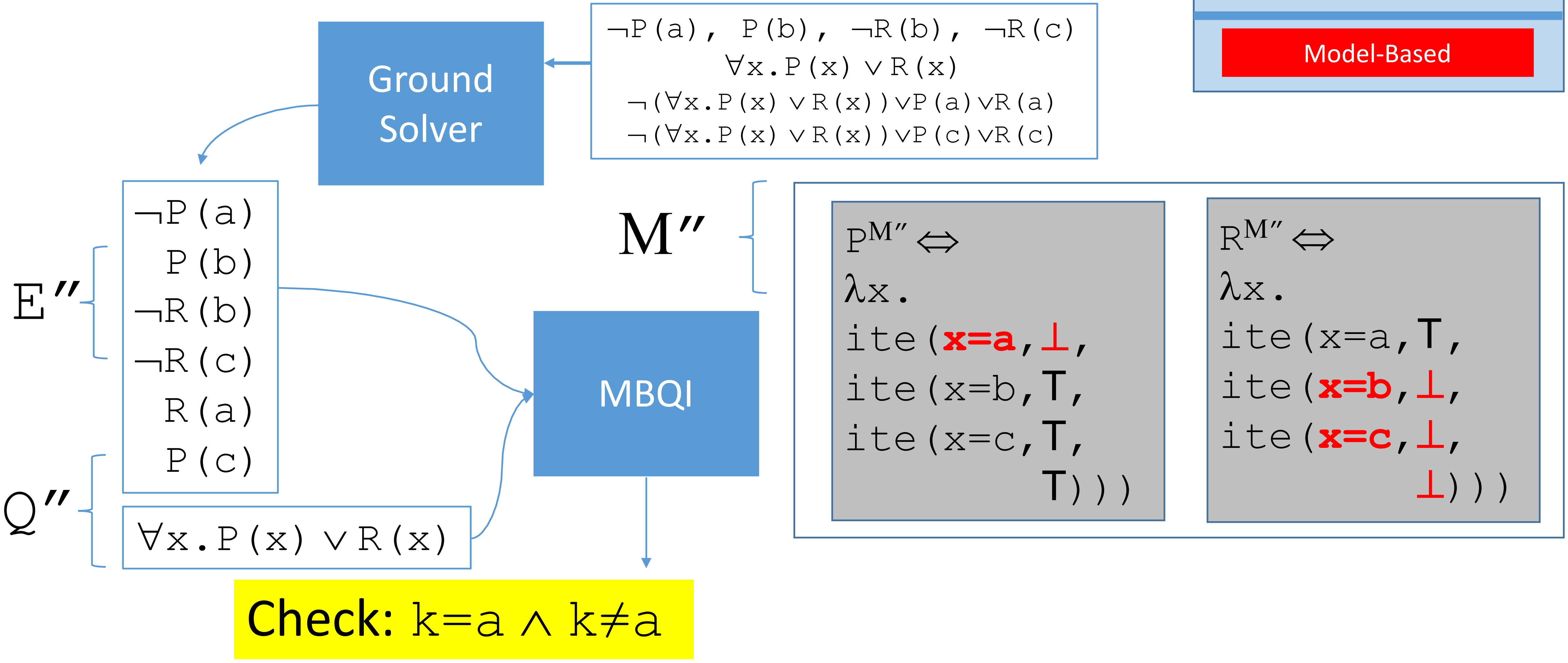
Check: $\exists x. \neg(P^{M''}(x) \lor R^{M''}(x))$

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



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

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

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

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

Ground Solver

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''

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

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

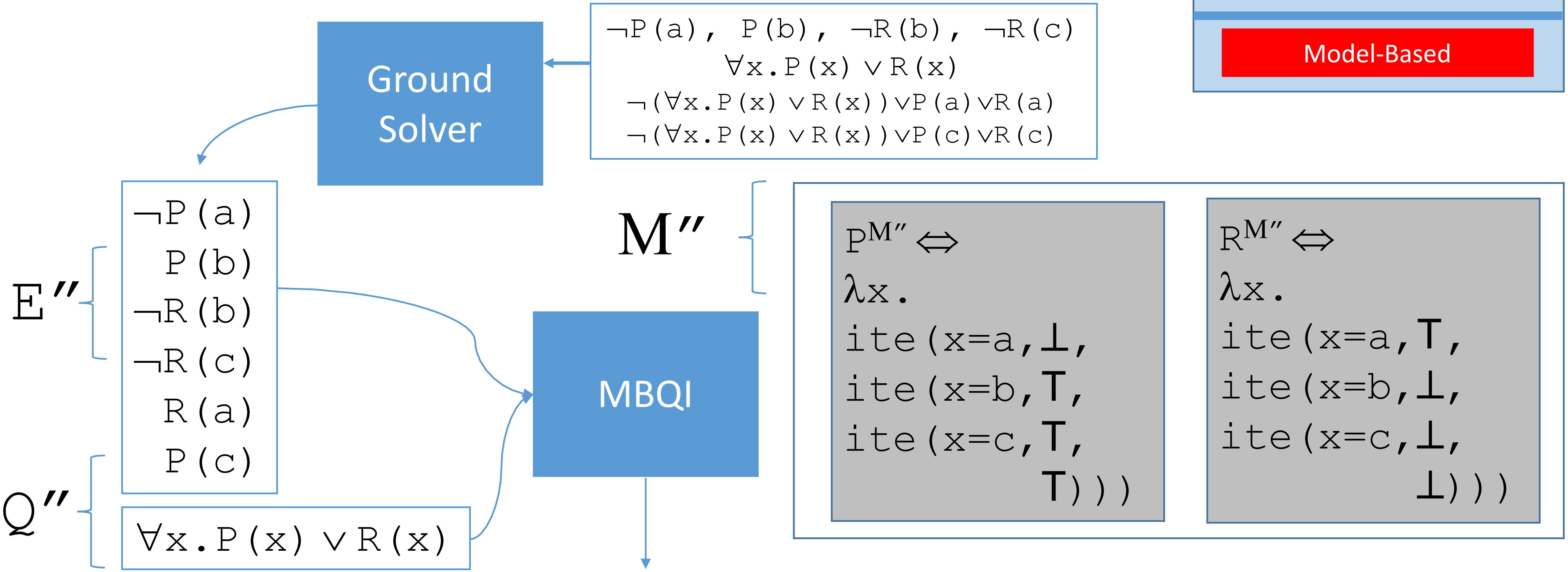
Check: $k=a \wedge k \neq a$

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



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

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

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

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

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

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

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

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

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

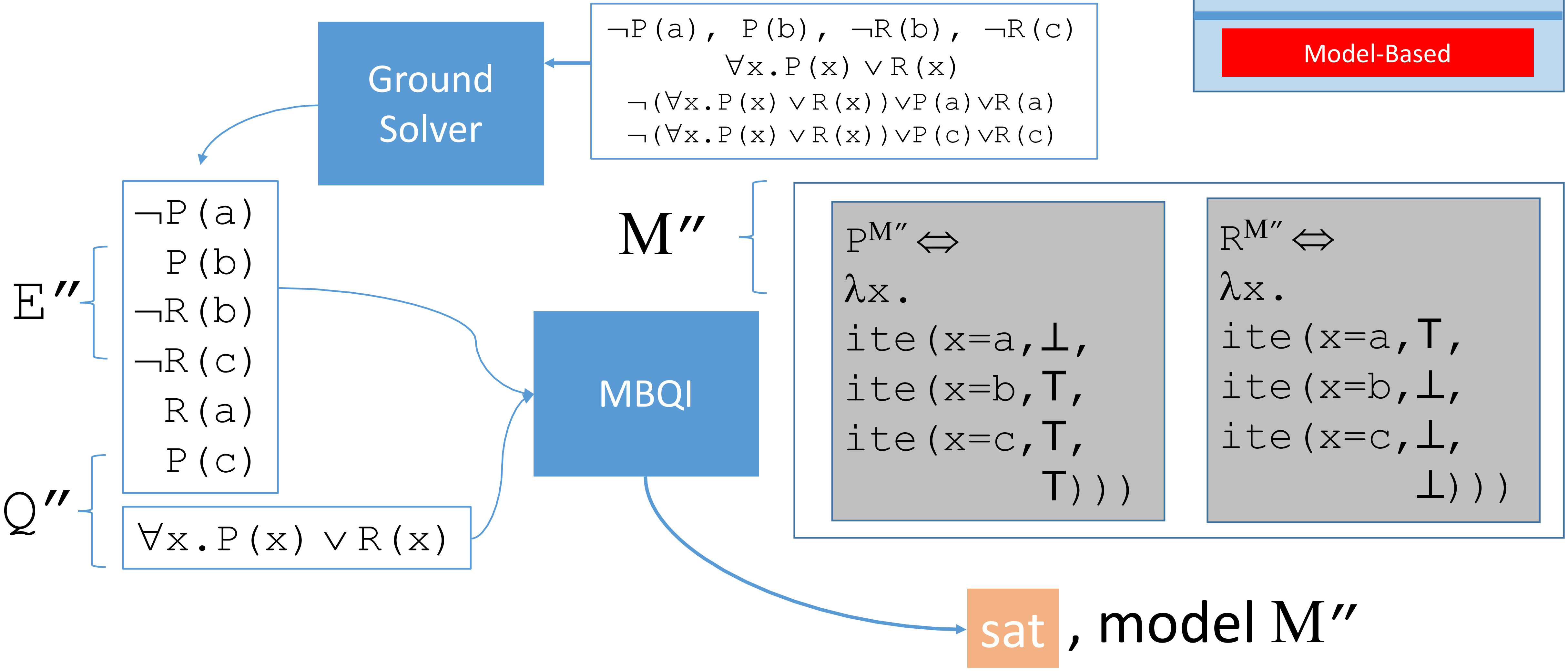
$\Longrightarrow$ Unsatisfiable, there are no values k for which M'' does not satisfy Q

Model-based Instantiation

Conflict-Based

E-matching

Model-Based



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

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

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

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

Ground Solver

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

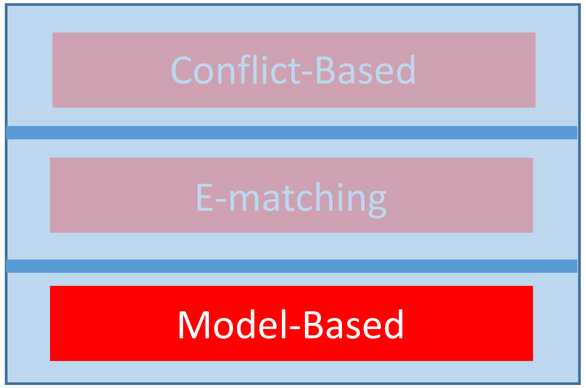
M''

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

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

sat, model M''

Model-based Instantiation: Completeness



Conflict-Based

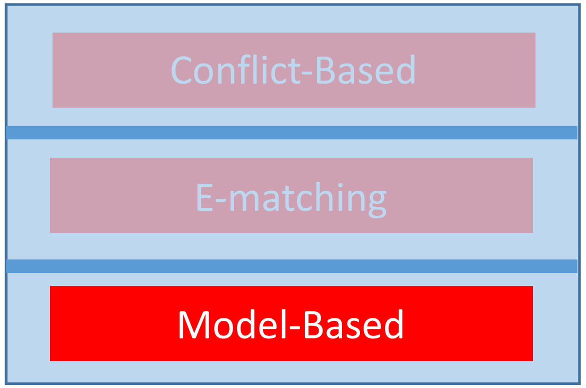
E-matching

Model-Based

- Seen techniques for which:
  - Ground Solver may answer unsat
  - Quantifiers Module (+ model-based instantiation) may answer sat

- Under what conditions are these techniques *terminating?*

Model-based Instantiation: Completeness



Conflict-Based

E-matching

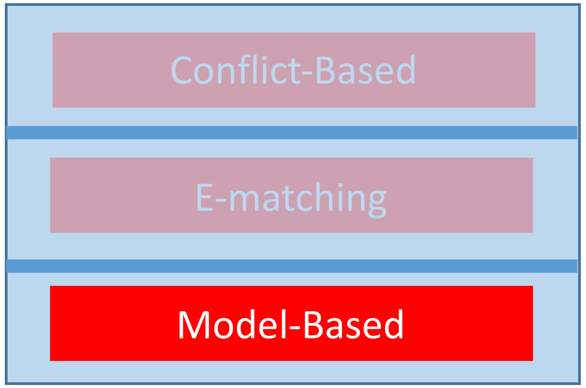
Model-Based

- Seen techniques for which:
  - Ground Solver may answer unsat
  - Quantifiers Module (+ model-based instantiation) may answer sat

- Under what conditions are these techniques *terminating?*

  A. If the domains of ∀ are interpreted as finite
     - E.g. quantified bitvectors **[Wintersteiger et al 13]**

Model-based Instantiation: Completeness



Conflict-Based

E-matching

Model-Based

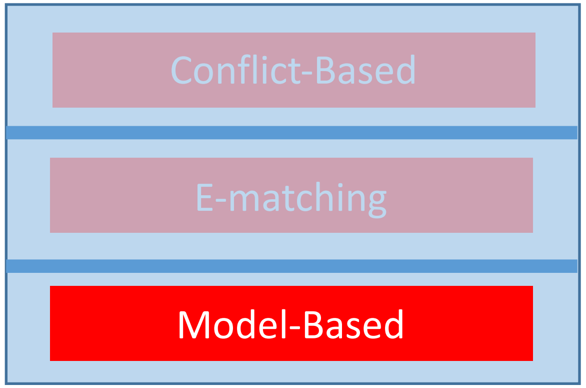
- Seen techniques for which:
  - Ground Solver may answer unsat
  - Quantifiers Module (+ model-based instantiation) may answer sat

- Under what conditions are these techniques *terminating?*

  A. If the domains of ∀ are interpreted as finite
     - E.g. quantified bitvectors **[Wintersteiger et al 13]**

  B. If the domains of ∀ may be interpreted as finite in a model
     - Finite model finding **[Reynolds et al 13]**

Model-based Instantiation: Completeness



- Seen techniques for which:
  - Ground Solver may answer unsat
  - Quantifiers Module (+ model-based instantiation) may answer sat

- Under what conditions are these techniques *terminating?*
  A. If the domains of ∀ are interpreted as finite
     - E.g. quantified bitvectors **[Wintersteiger et al 13]**
  B. If the domains of ∀ may be interpreted as finite in a model
     - Finite model finding **[Reynolds et al 13]**
  C. If the domains of ∀ are infinite

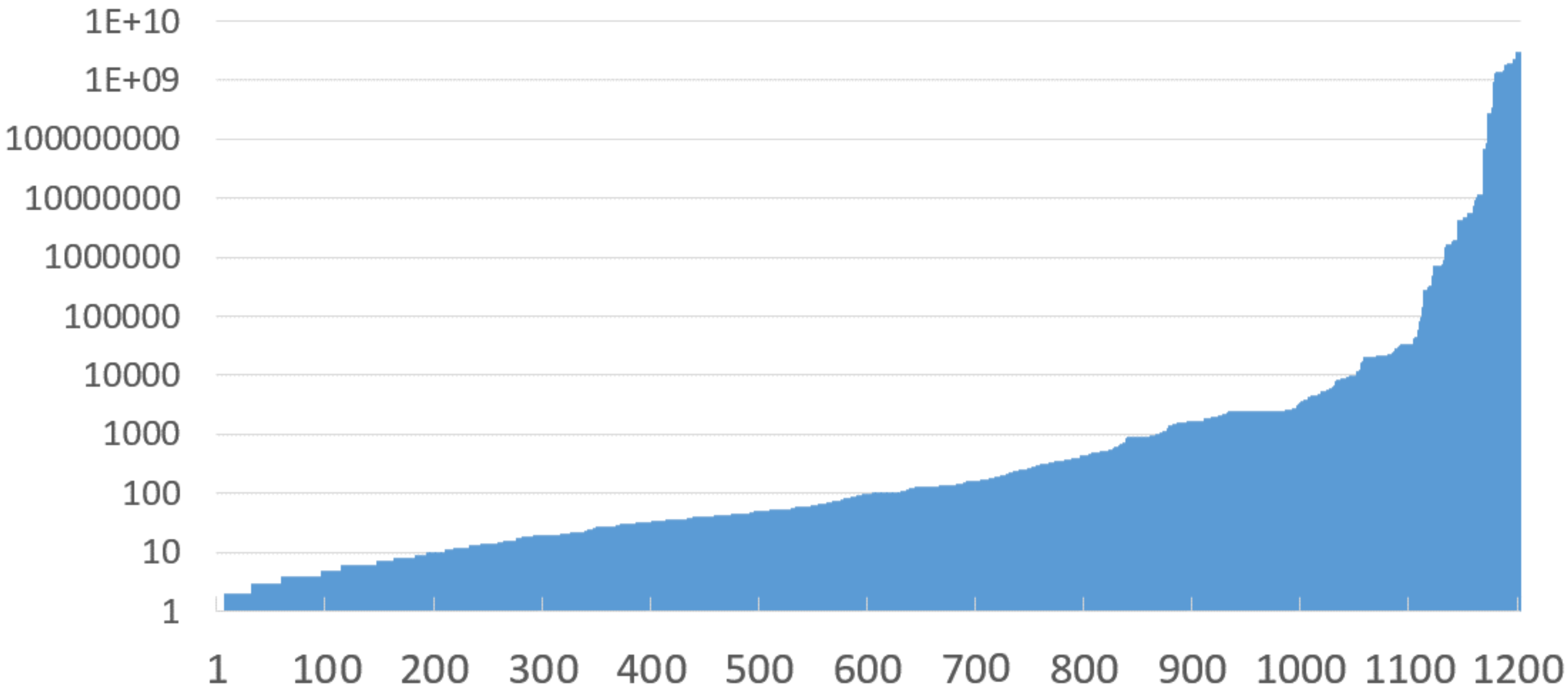
     …but it can be argued that only finitely many instances will be generated
     - E.g. essentially uninterpreted fragment **[Ge+deMoura 09]**, …

Conflict-Based

E-matching

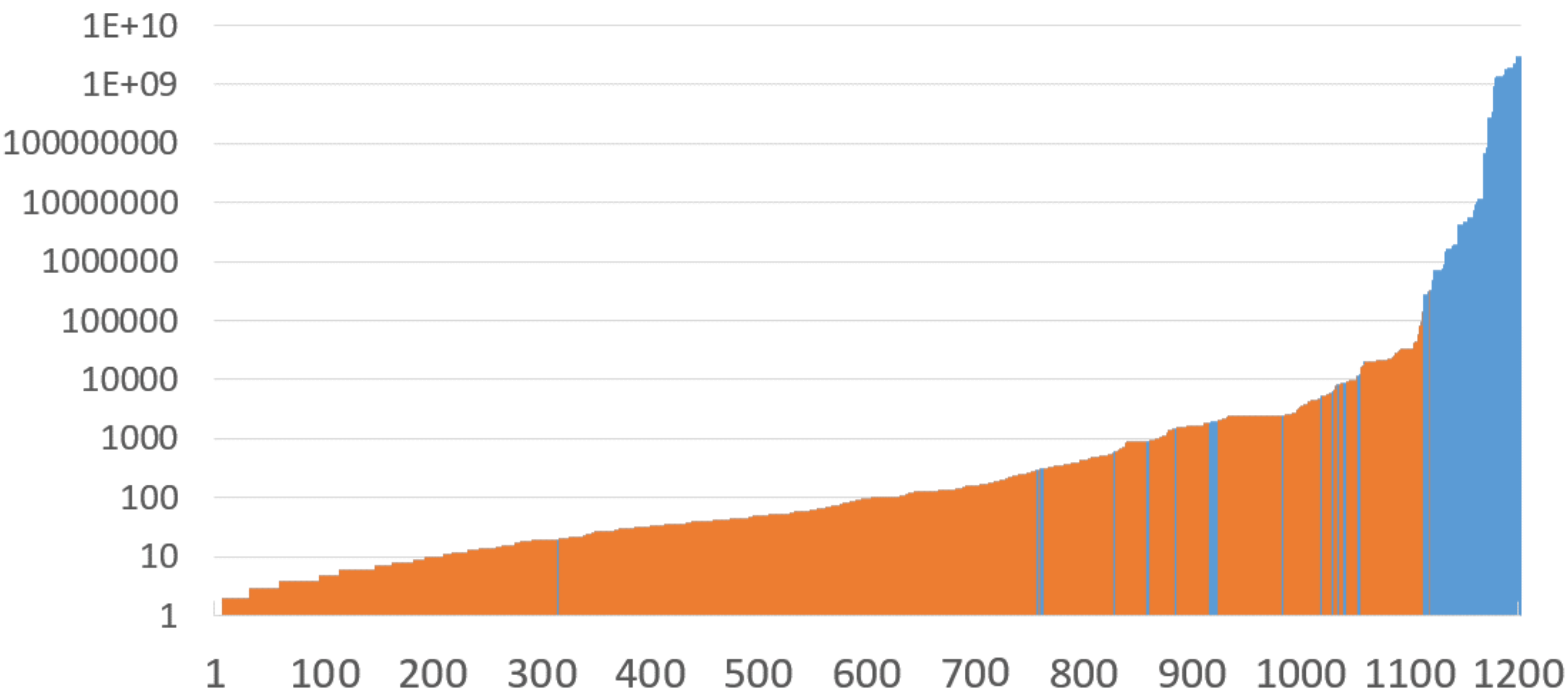
Model-Based

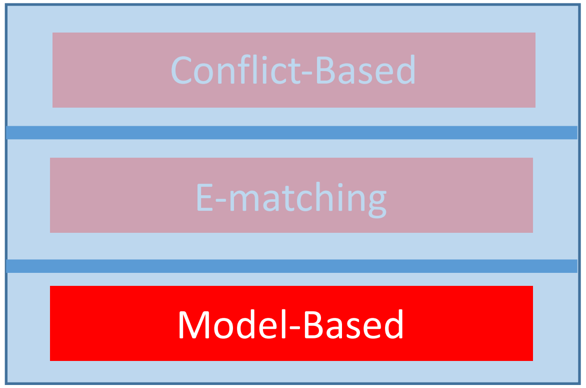
Model-based Instantiation: Impact



- 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

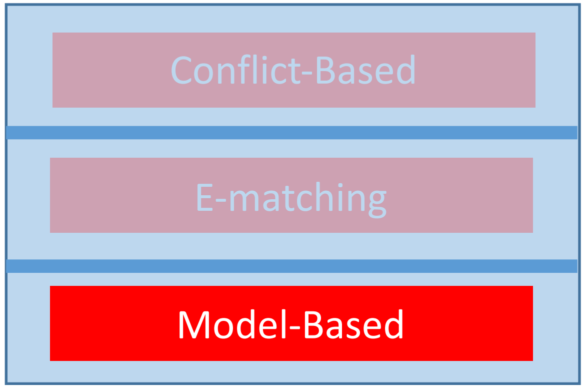
Model-based Instantiation: Impact

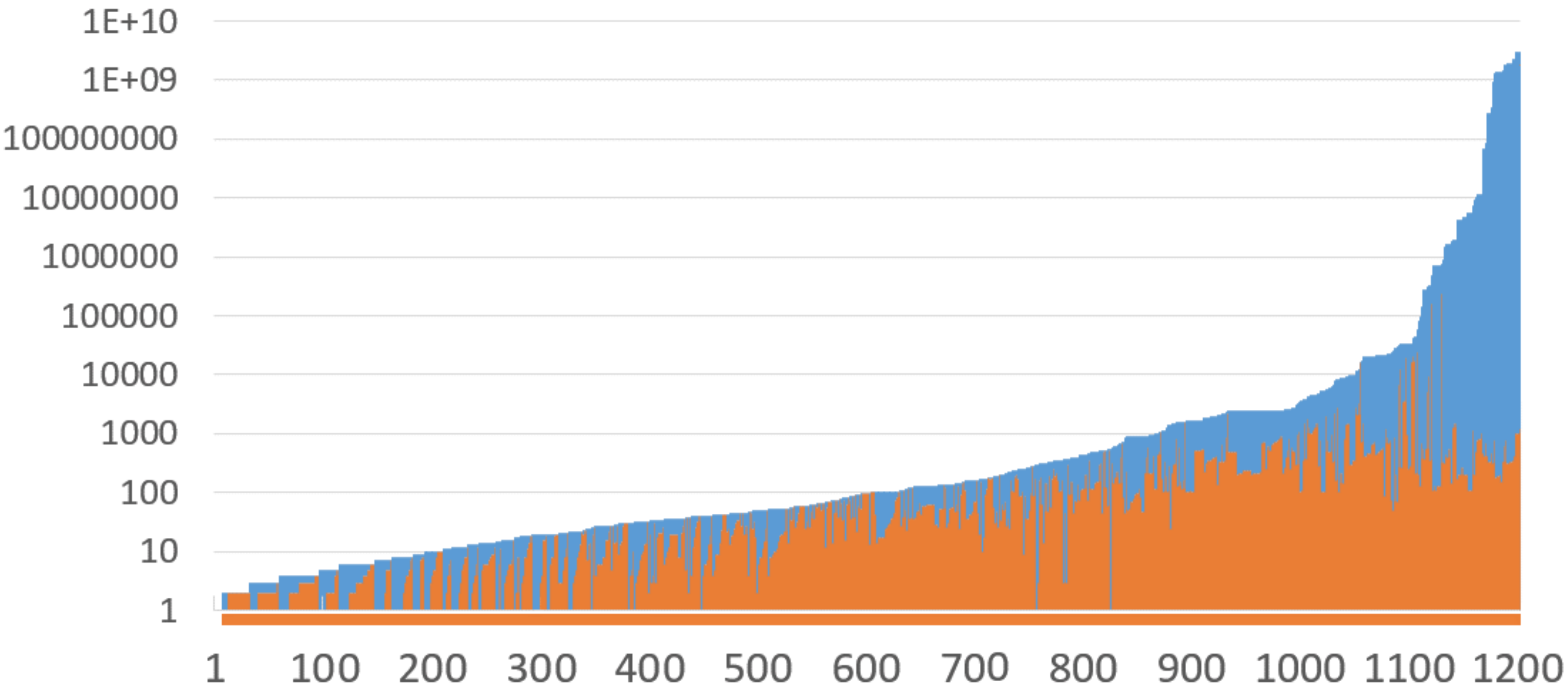




- CVC4 Finite Model Finding + Exhaustive instantiation
  - Scales only up to ~150k instances with a 30 sec timeout

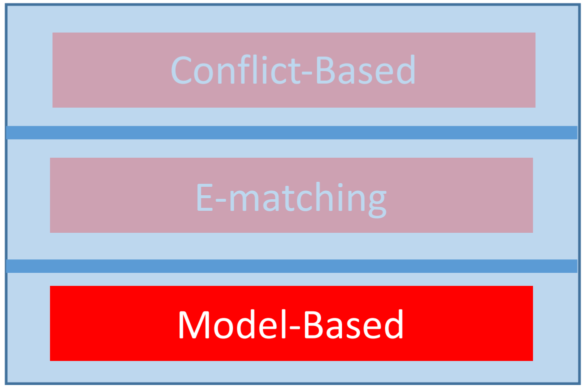
Model-based Instantiation: Impact





- 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

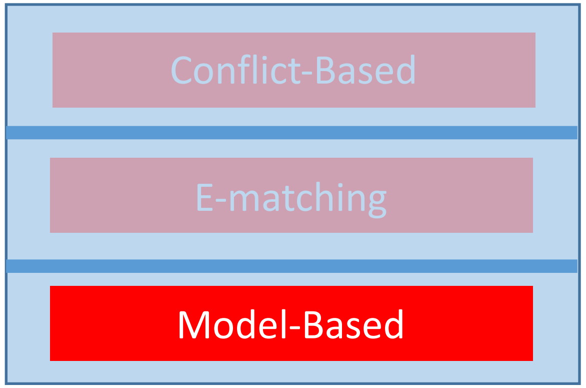


Conflict-Based

E-matching

Model-Based

Model-based Instantiation: Challenges



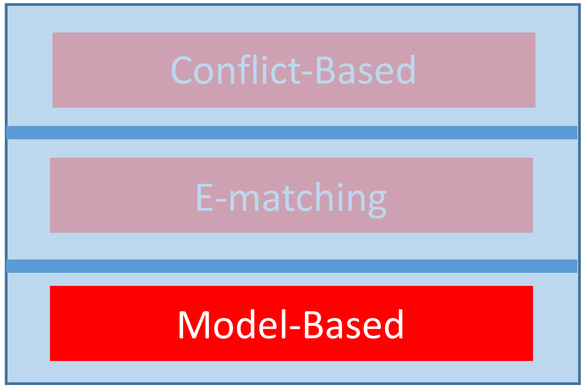
Conflict-Based

E-matching

Model-Based

- How do we build interpretations M ?
  - Typically, build interpretations $\mathtt{f}^{\mathrm{M}}$ that are almost constant:
    - e.g. $\mathtt{f}^{\mathrm{M}} := \lambda\mathtt{x}.\mathtt{ite}(\mathtt{x}=\mathtt{t}_1,\mathtt{v}_1,\mathtt{ite}(\mathtt{x}=\mathtt{t}_2,\mathtt{v}_2,\ldots,\mathtt{ite}(\mathtt{x}=\mathtt{t}_n,\mathtt{v}_n,\mathtt{v}_{\mathtt{def}})\ldots))$

Model-based Instantiation: Challenges


Conflict-Based

E-matching

Model-Based

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

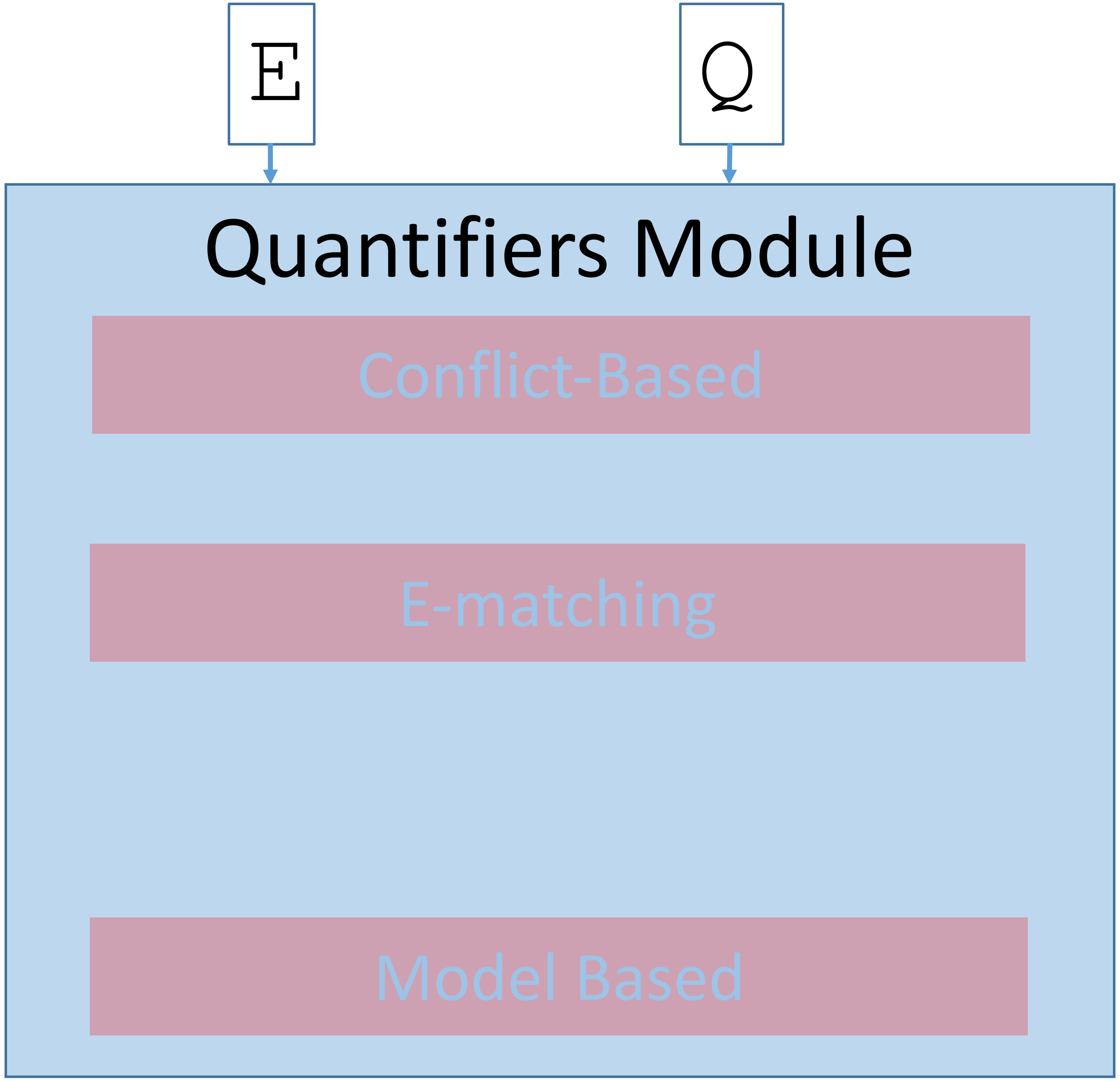
…but models may need to be more complex when *theories are present*:

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

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

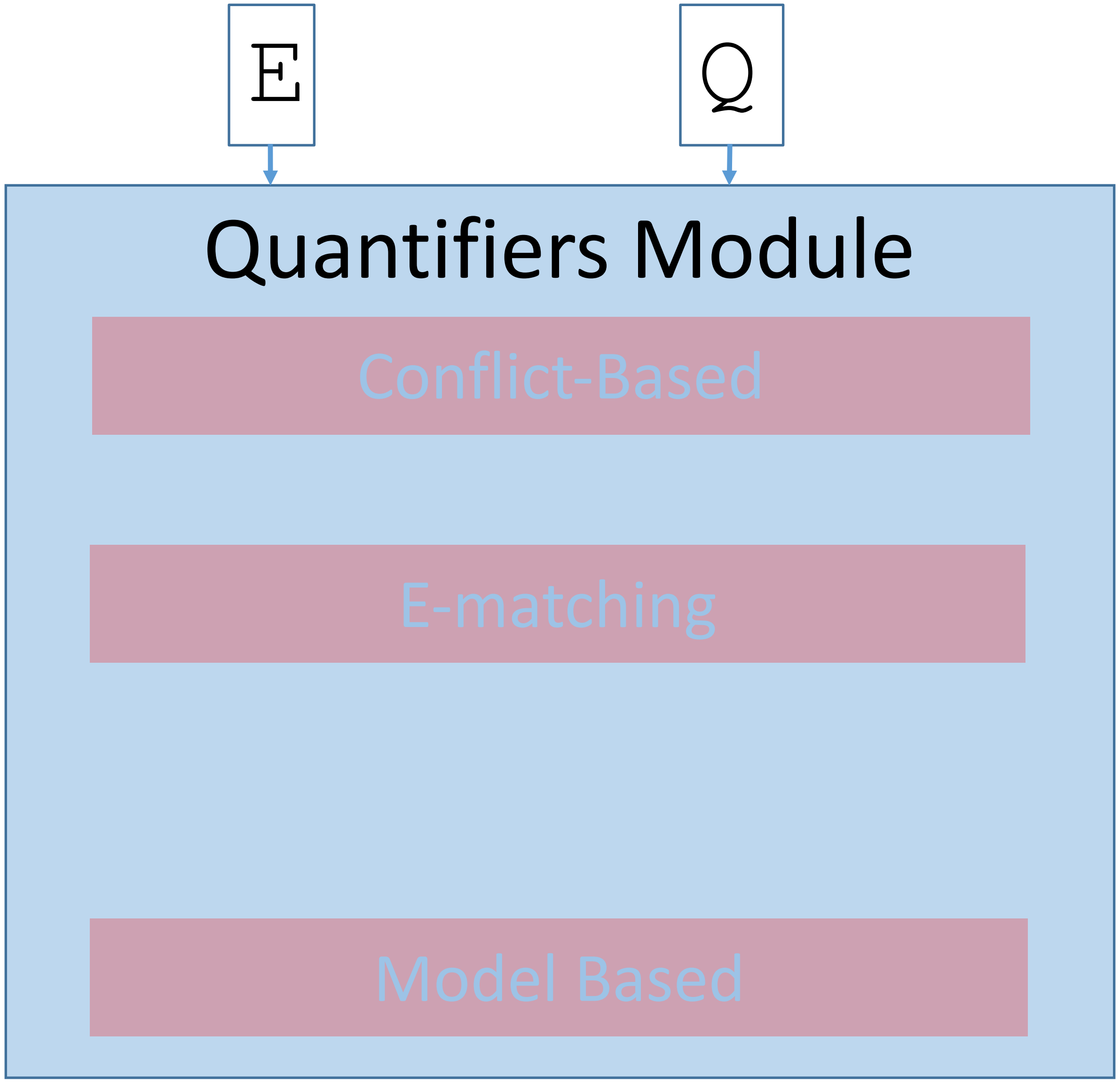
$\forall\mathtt{xy:Int.u(x+y)+11*v(w(x))=x+y}$ → ???

Putting it Together



Putting it Together

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



E

Q

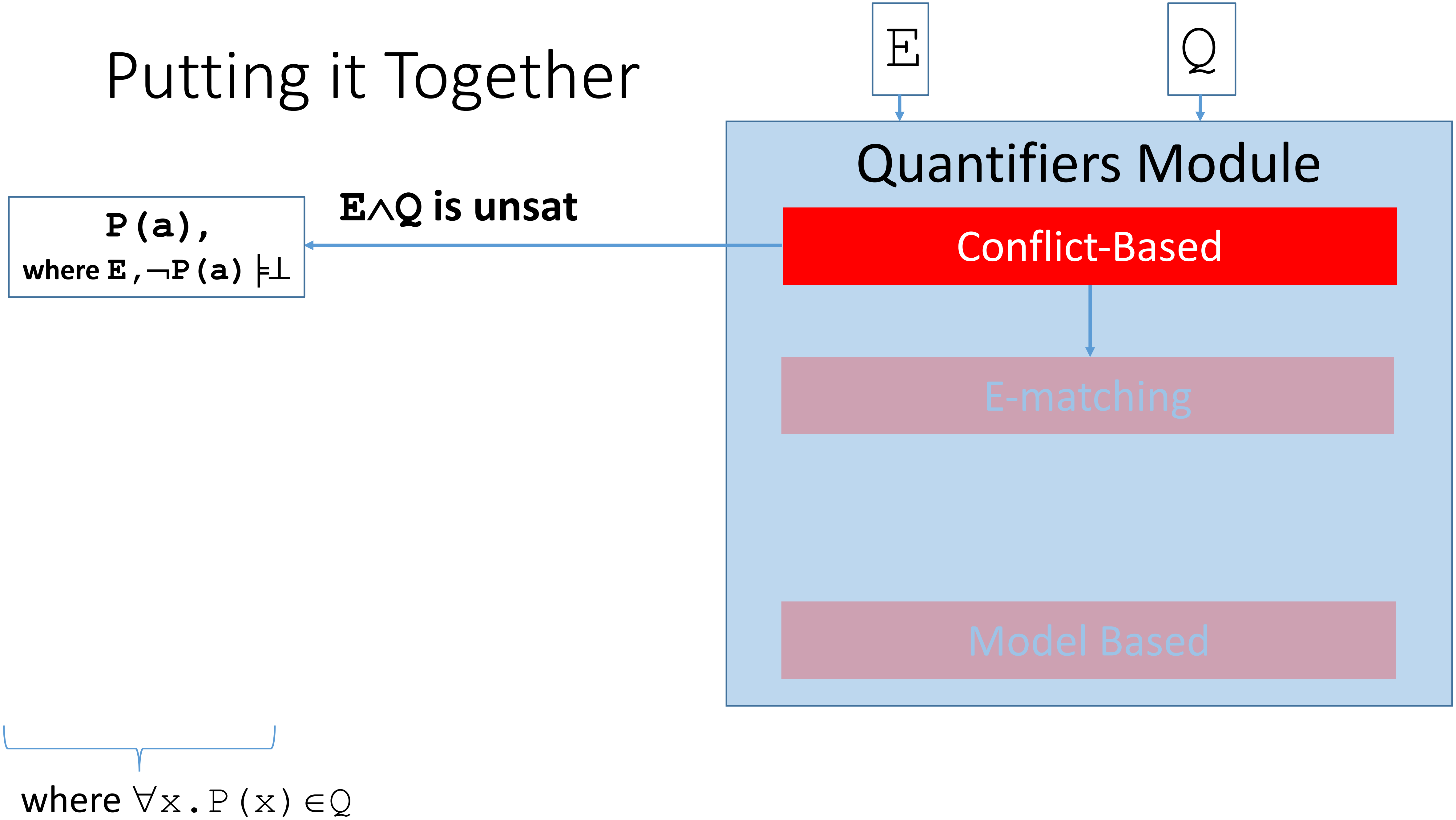
Quantifiers Module

Conflict-Based

E-matching

Model Based

Putting it Together



E

Q

Quantifiers Module

Conflict-Based

E-matching

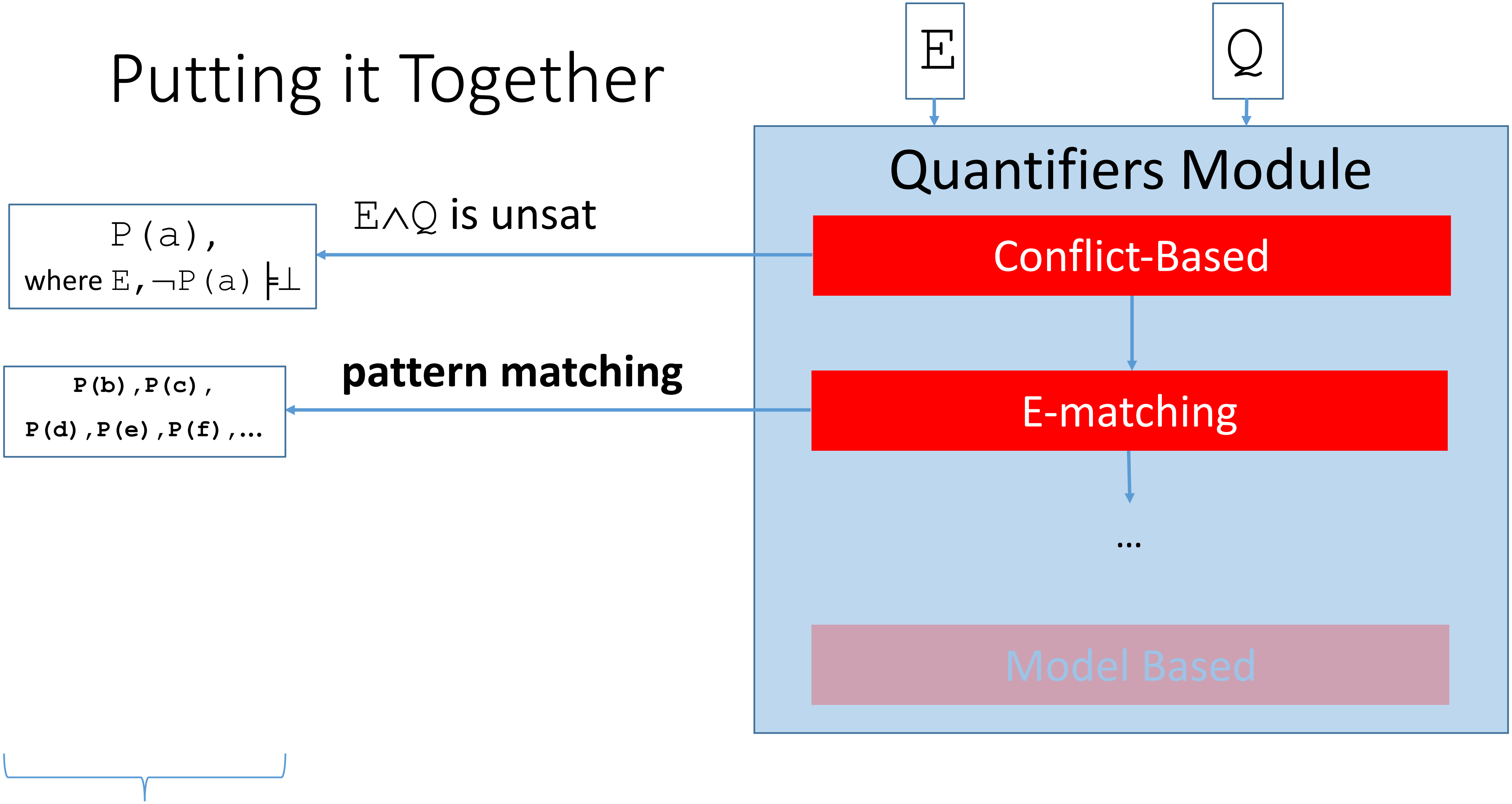
Model Based

$E \wedge Q$ is unsat

$P(a)$,
where $E, \neg P(a) \models \bot$

where $\forall x.P(x) \in Q$

Putting it Together



E

Q

Quantifiers Module

Conflict-Based

E-matching

…

Model Based

$E \wedge Q$ is unsat

P(a),

where $E, \neg P(a) \models \bot$

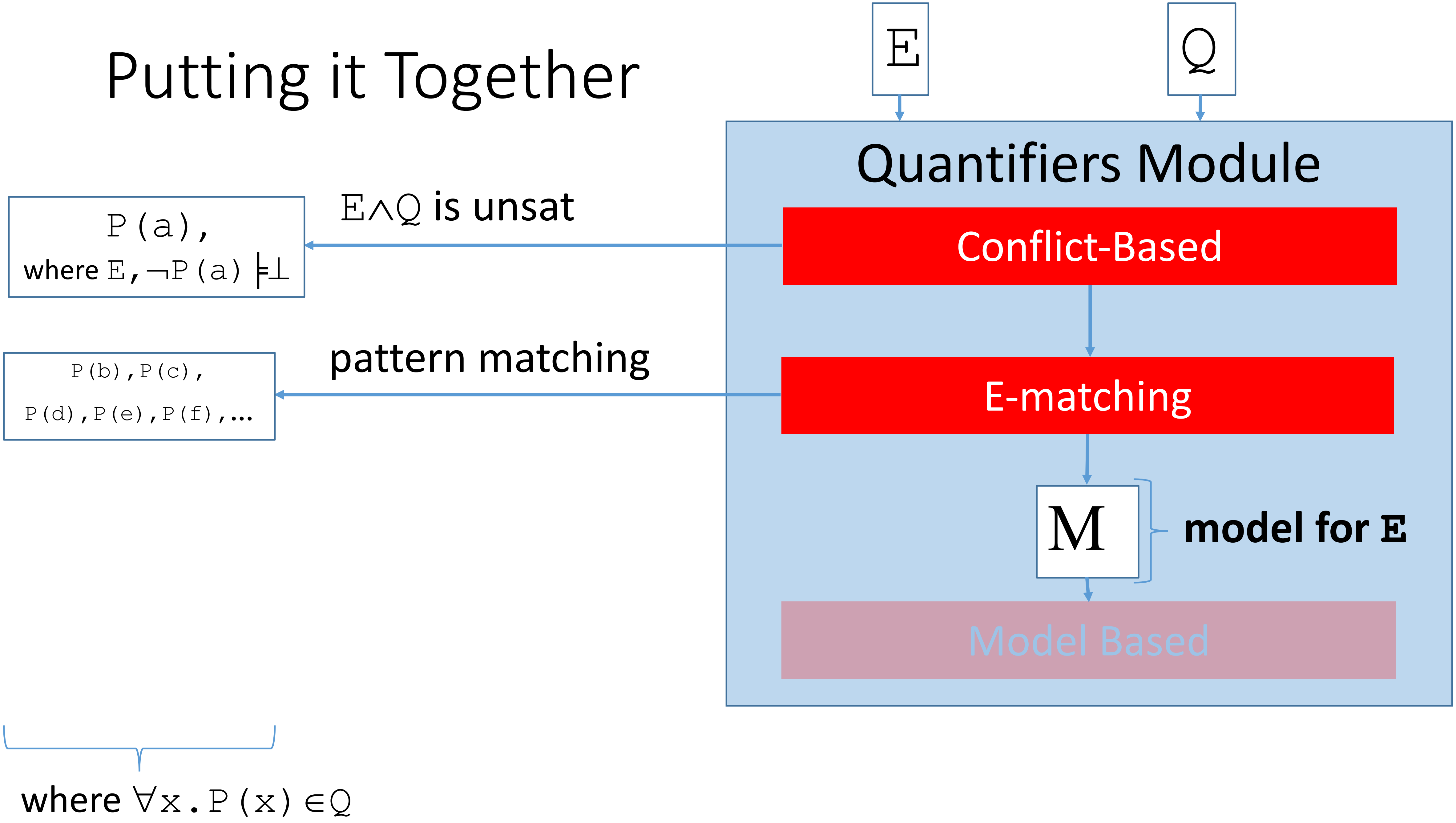
pattern matching

P(b),P(c),

P(d),P(e),P(f),…

where $\forall x.P(x) \in Q$

Putting it Together



E

Q

Quantifiers Module

Conflict-Based

E-matching

M

model for **E**

Model Based

E∧Q is unsat

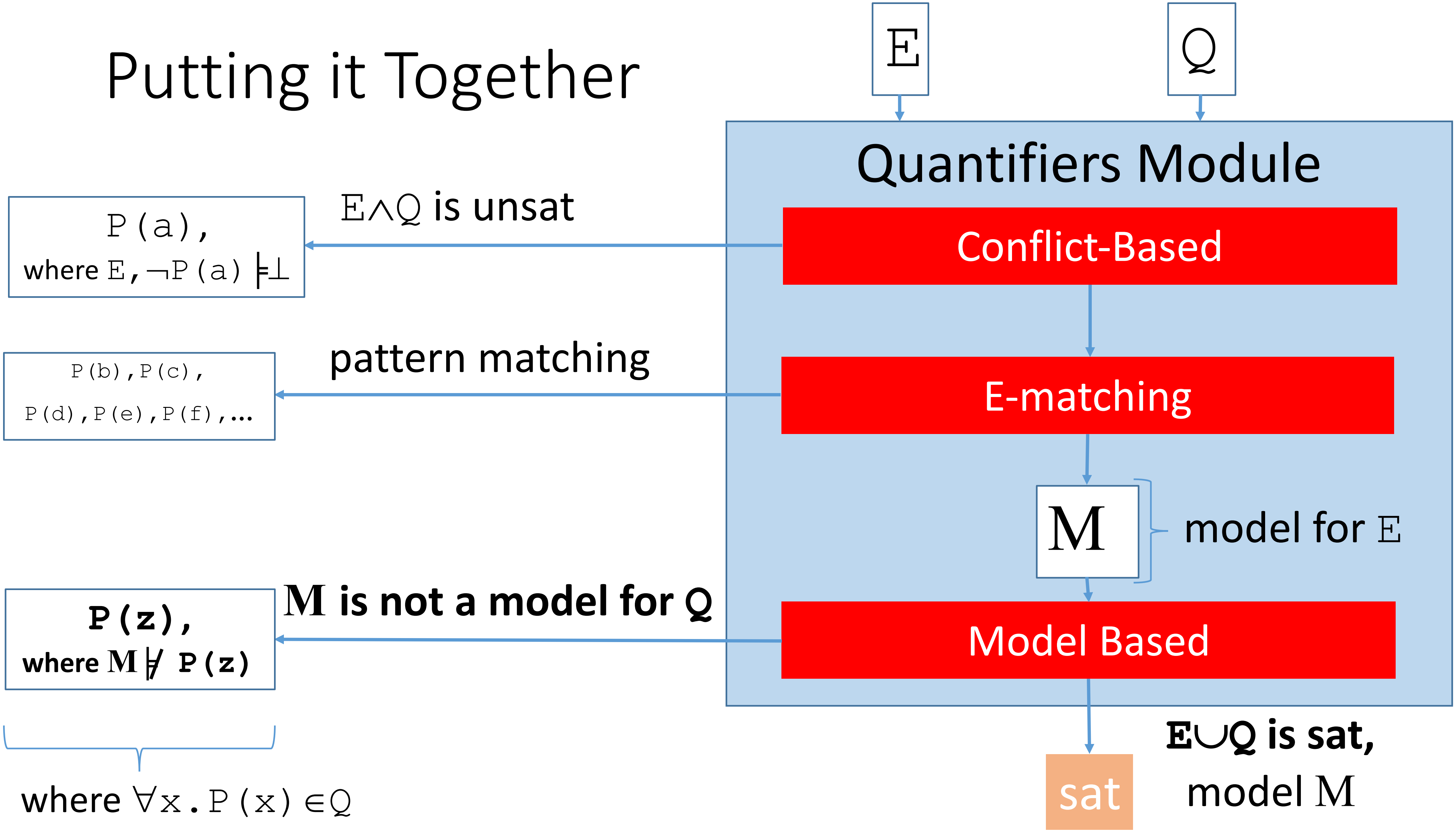
P(a),
where $E, \neg P(a) \models \bot$

pattern matching

P(b),P(c),
P(d),P(e),P(f),…

where $\forall x.P(x) \in Q$

Putting it Together



E

Q

Quantifiers Module

Conflict-Based

E-matching

M

model for E

Model Based

$E \wedge Q$ is unsat

P(a),
where $E, \neg P(a) \models \bot$

pattern matching

P(b),P(c),
P(d),P(e),P(f),…

M is not a model for Q

P(z),
where $M \not\models$ P(z)

where $\forall x.P(x) \in Q$

sat

$E \cup Q$ is sat,
model M

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

- **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

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

- **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 ∀ + *pure* theories isn't as bad:

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

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

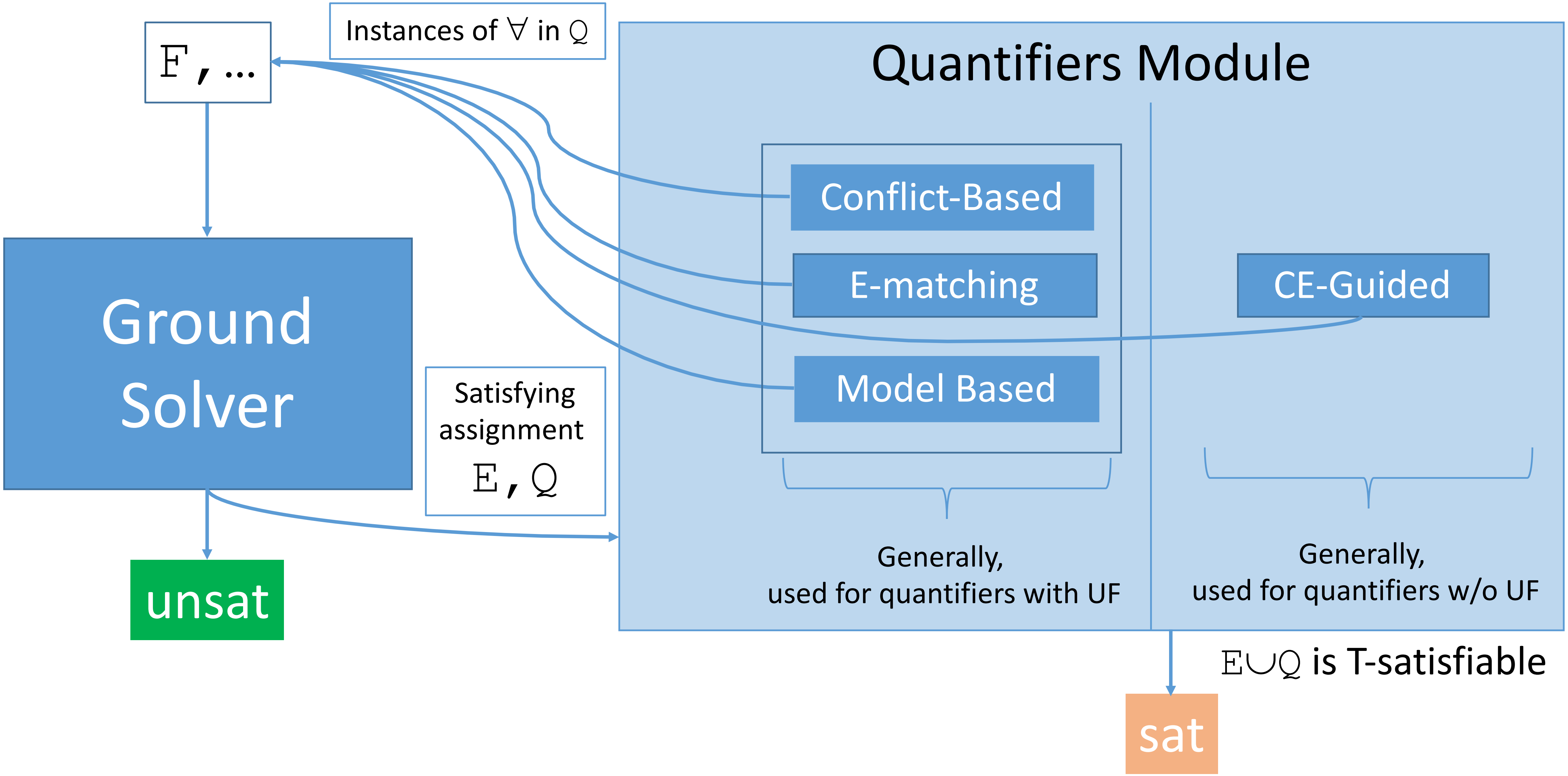
- **Common thread:** satisfiability of ∀ + UF + theories is hard!
  - E-matching:
    - Pattern selection, matching modulo theories
  - Conflict-based:
    - Matching is incomplete, entailment tests are expensive
  - Model-based:
    - Models are complex, interpreted domains (e.g. Int) may be infinite

⇒ But reasoning about ∀ + *pure* theories isn't as bad:

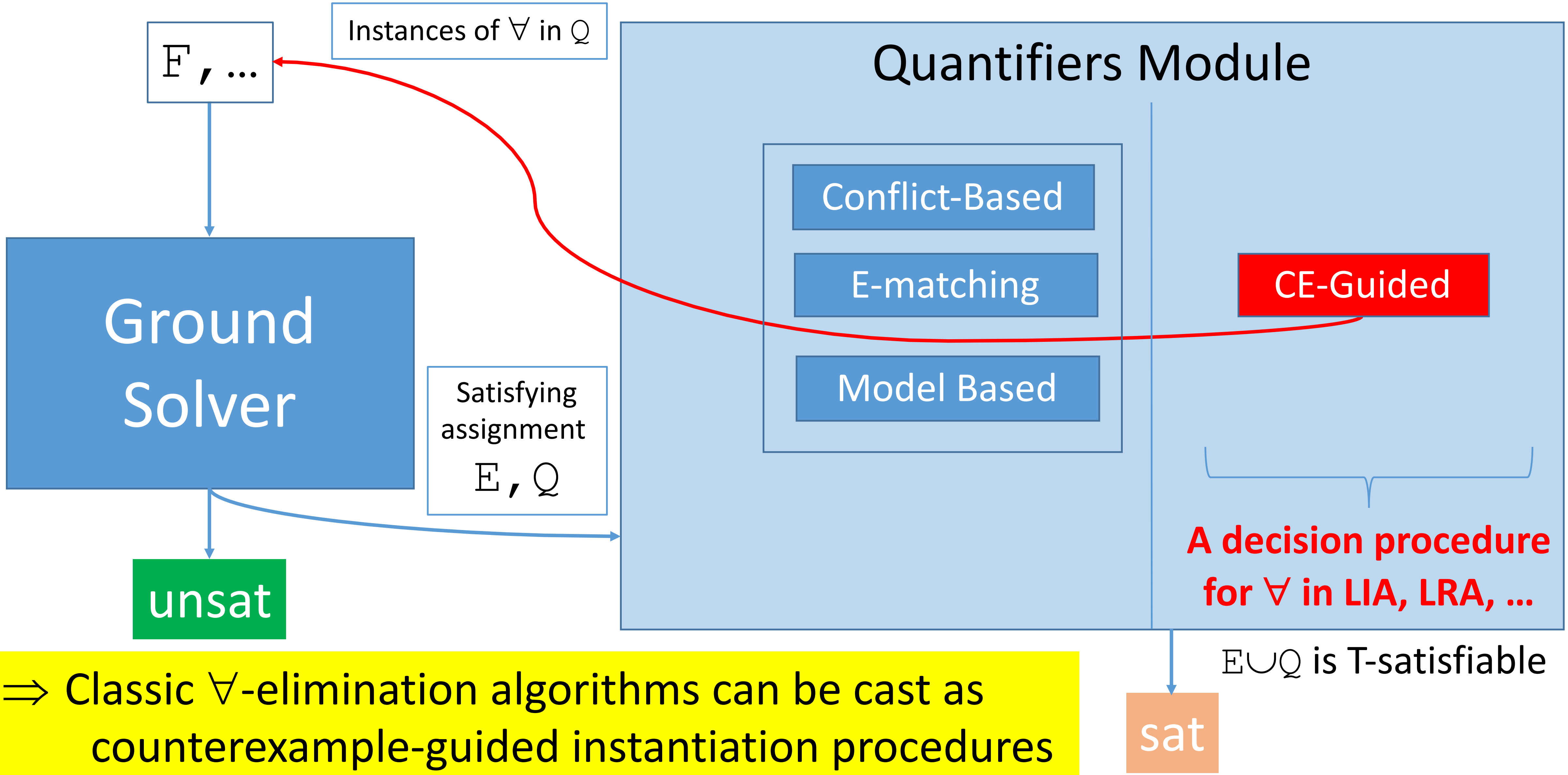
- Classic ∀-elimination algorithms are decision procedures for ∀ in:
  - LRA **[Ferrante+Rackoff 79, Loos+Wiespfenning 93]** , LIA **[Cooper 72]**, datatypes, …

- Can classic ∀-elimination algorithms be implemented in an SMT context?
  - Yes: **[Monniaux 2010, Bjorner 2012, Komuravelli et al 2014, Reynolds et al 2015, Bjorner/Janota 2016]**

Techniques for Quantifier Instantiation

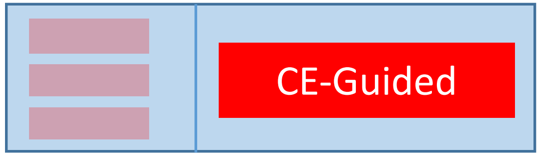


Techniques for Quantifier Instantiation



⇒ Classic ∀-elimination algorithms can be cast as counterexample-guided instantiation procedures

Counterexample-Guided Instantiation



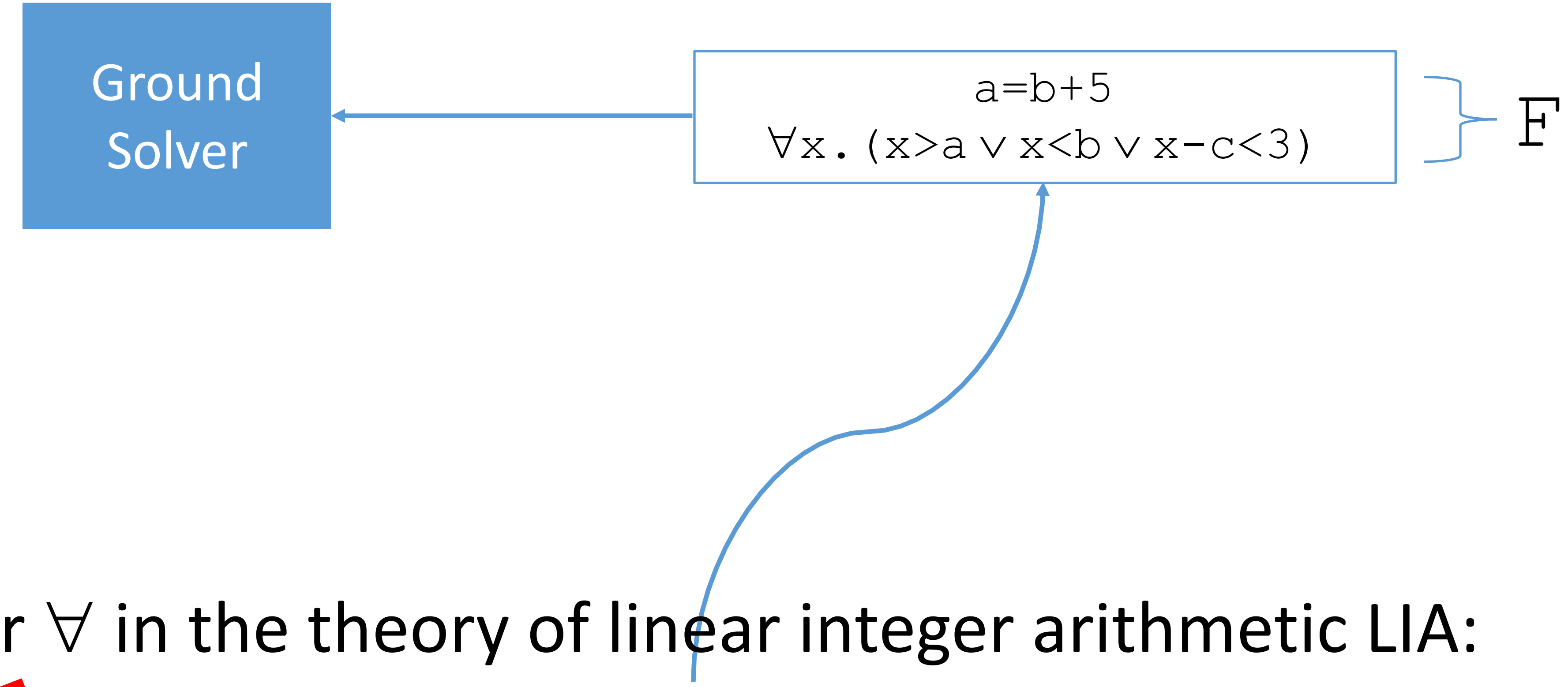
CE-Guided

⇒ Consider ∀ in the theory of linear integer arithmetic LIA:

$$\exists abc.(a=b+5 \land \forall x.(x>a \lor x<b \lor x-c<3))$$

Counterexample-Guided Instantiation

CE-Guided



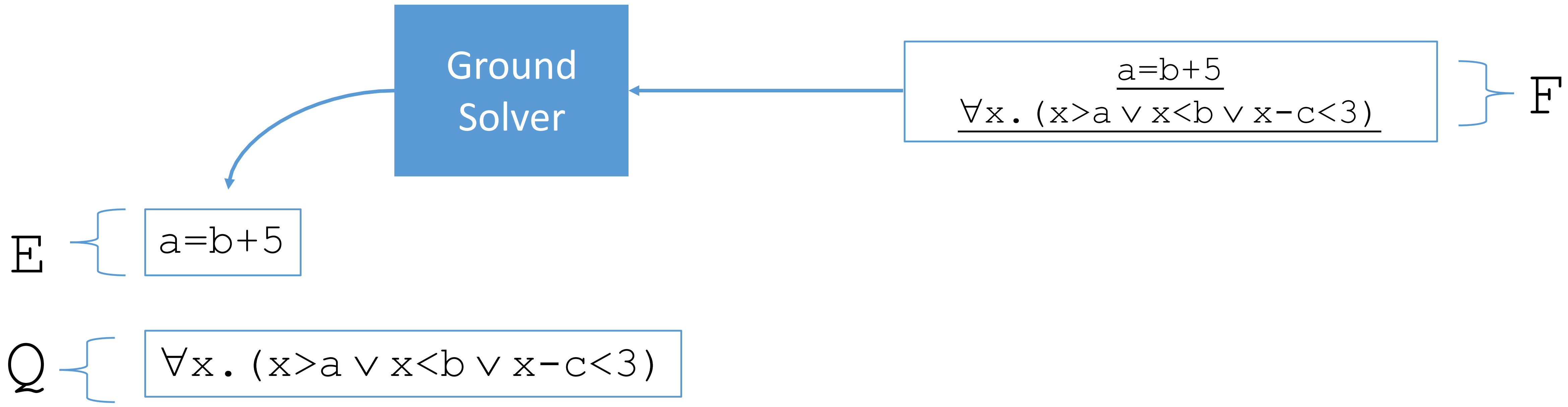
⇒ Consider ∀ in the theory of linear integer arithmetic LIA:

$$\cancel{\exists abc}.\,(\mathbf{a=b+5} \land \forall \mathbf{x}.\,(\mathbf{x>a} \lor \mathbf{x<b} \lor \mathbf{x-c<3}))$$

- Outermost existentials `a,b,c` are treated as *free constants*

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$\underline{a=b+5}$

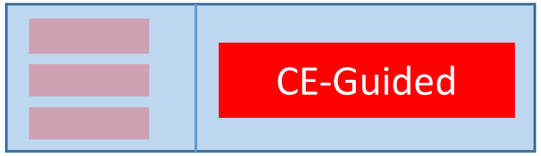
$\underline{\forall x.(x>a \lor x<b \lor x-c<3)}$

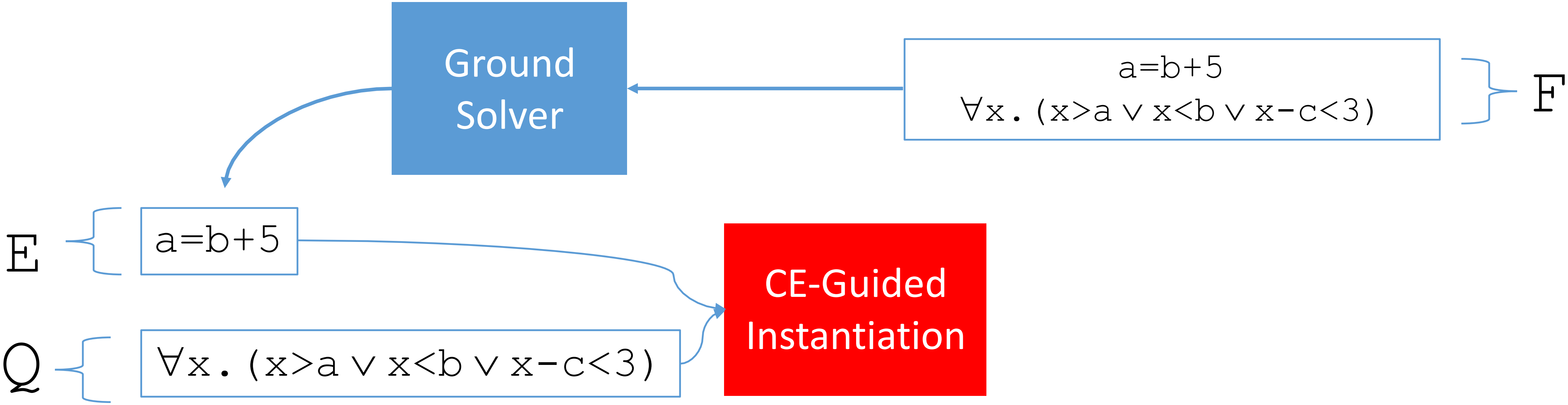
F

E $a=b+5$

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

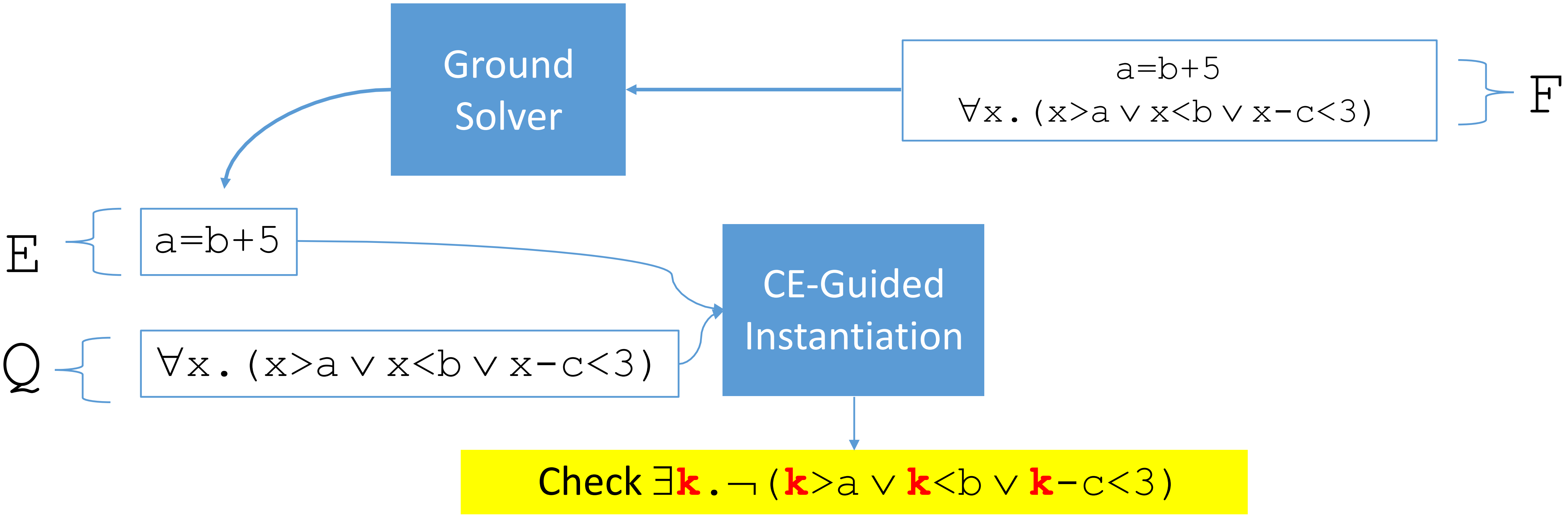
Counterexample-Guided Instantiation





⇒ Use counterexample-guided instantiation

Counterexample-Guided Instantiation

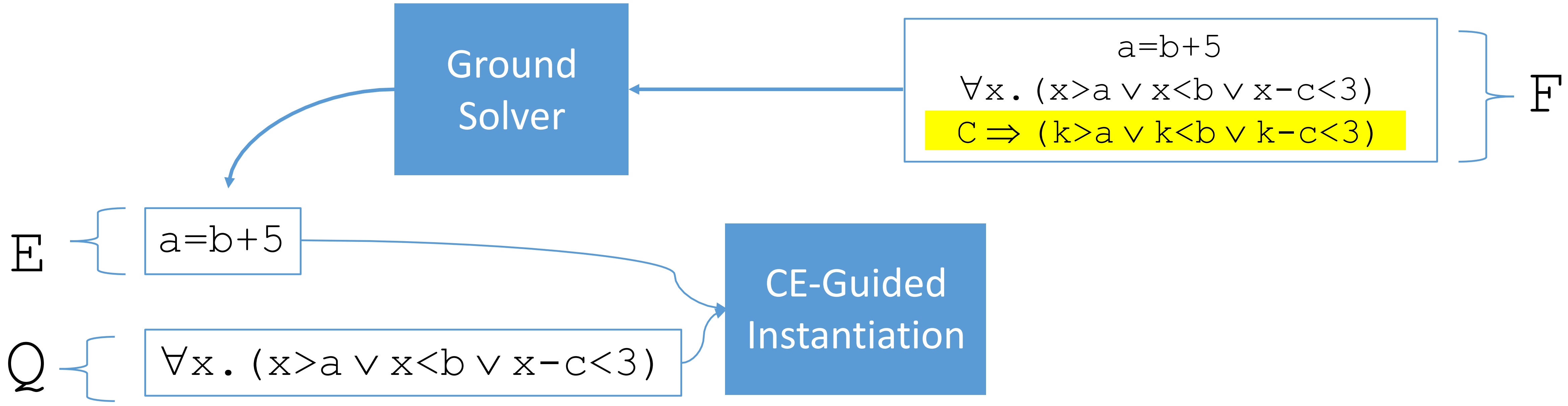


⇒With respect to *model-based instantiation*:

- Similar: check satisfiability of $\exists \mathbf{k}.\neg(\mathbf{k}>a \vee \mathbf{k}<b \vee \mathbf{k}-c<3)$

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

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

$C \Rightarrow (k>a \lor k<b \lor k-c<3)$

F

E

$a=b+5$

Q

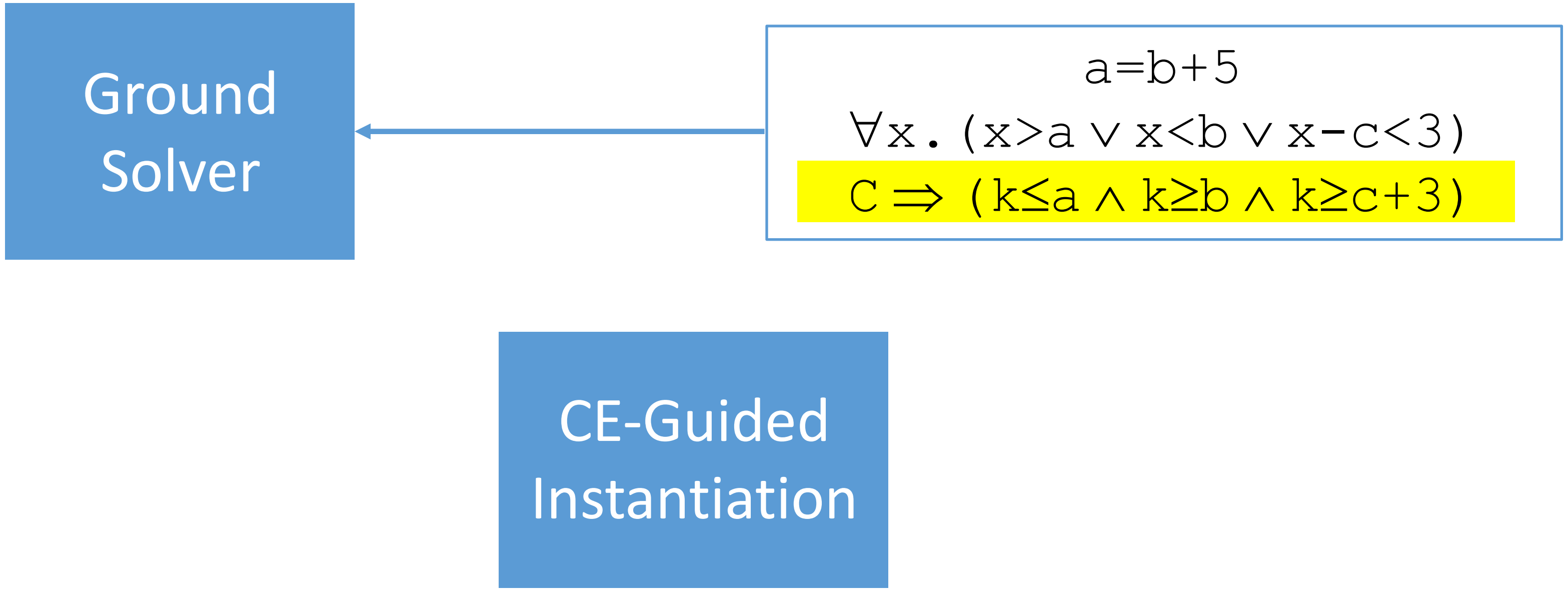
$\forall x.(x>a \lor x<b \lor x-c<3)$

CE-Guided Instantiation

⇒With respect *to model-based instantiation*:

- Similar: check satisfiability of $\exists k.\neg(k>a \lor k<b \lor k-c<3)$
- *Key difference:* use the same (ground) solver for F and *counterexample* k *for* Q

Counterexample-Guided Instantiation



Ground Solver

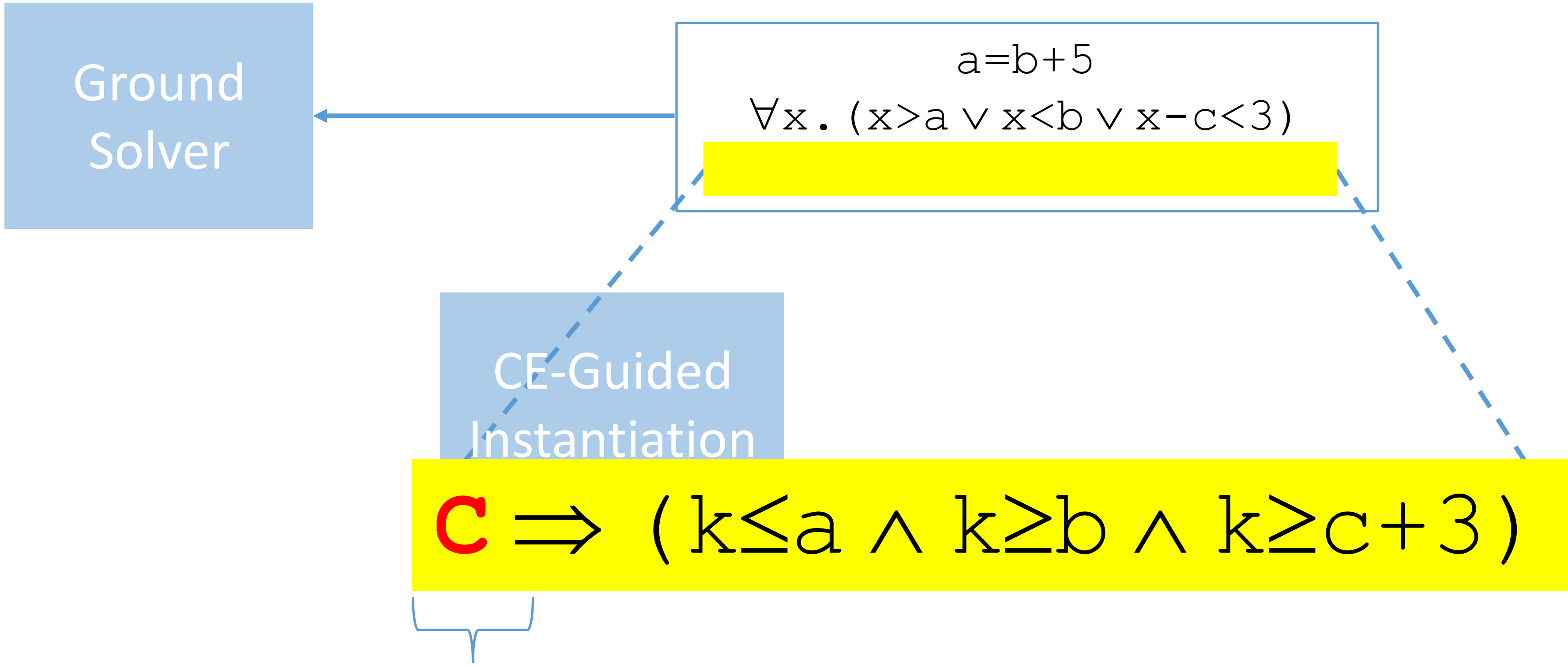
$a=b+5$

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

$C \Rightarrow (k \le a \wedge k \ge b \wedge k \ge c+3)$

CE-Guided Instantiation

Counterexample-Guided Instantiation



$a=b+5$

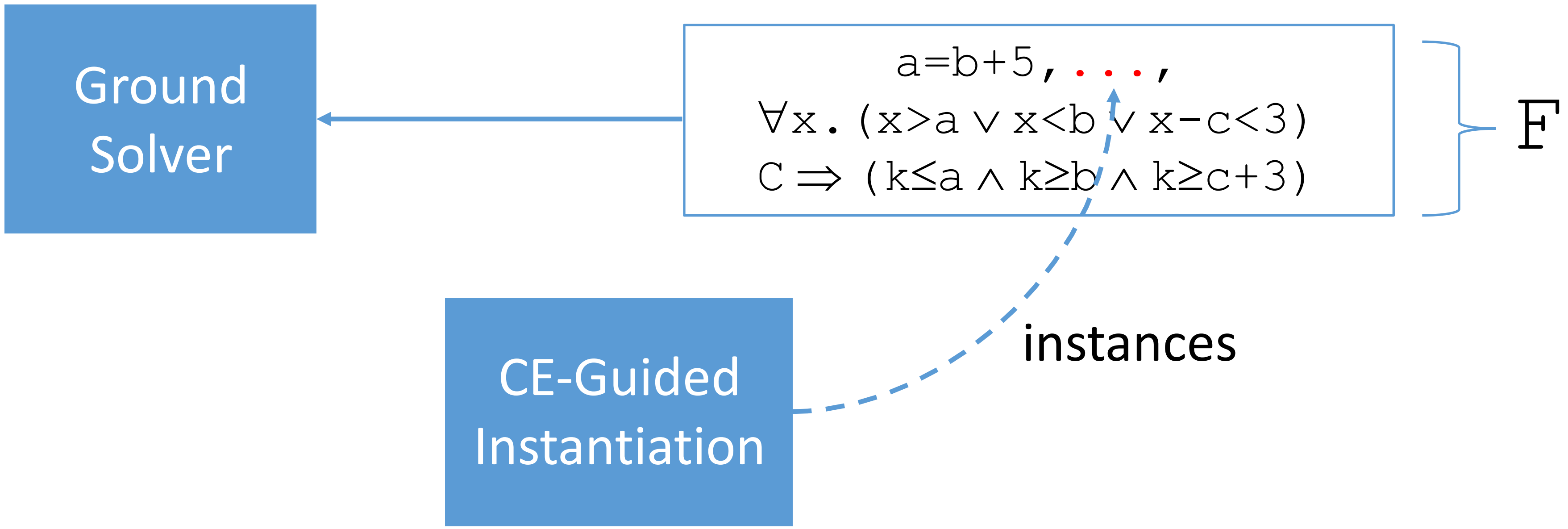
$\forall x.\,(x>a \vee x<b \vee x-c<3)$

$$C \Rightarrow (k \leq a \wedge k \geq b \wedge k \geq c+3)$$

C is a fresh Boolean variable:

“A counterexample k exists for $\forall x.\,(x>a \vee x<b \vee x-c<3)$”

Counterexample-Guided Instantiation



Ground Solver

a=b+5, ...,

∀x. (x>a ∨ x<b ∨ x−c<3)

C ⇒ (k≤a ∧ k≥b ∧ k≥c+3)

F

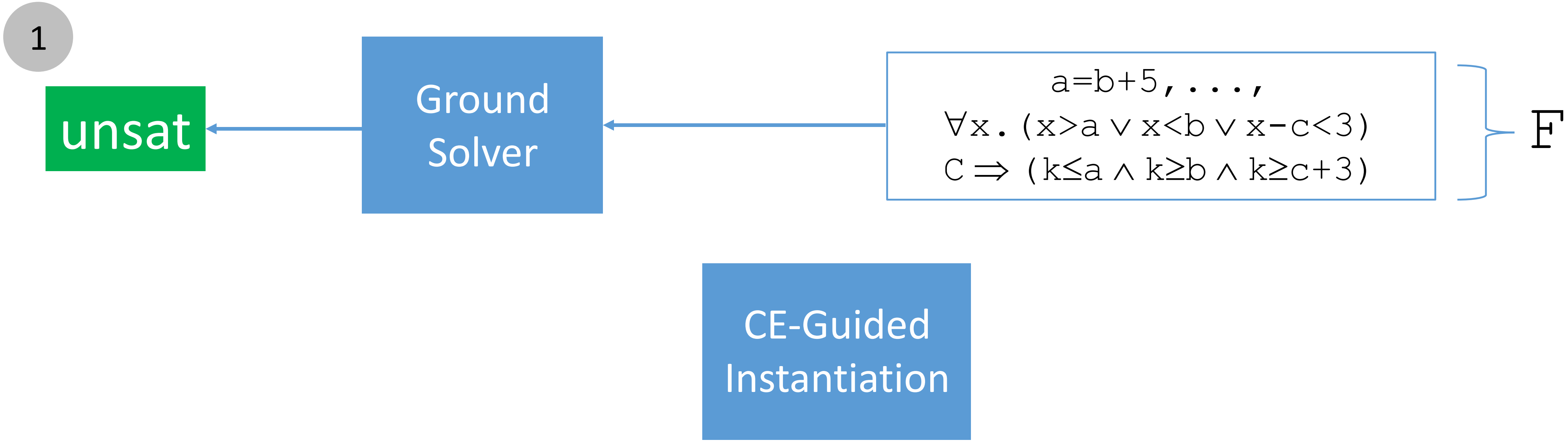
instances

CE-Guided Instantiation

- Three cases:

Counterexample-Guided Instantiation

CE-Guided



1

unsat

Ground Solver

$a=b+5, ...,$

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

$C \Rightarrow (k\leq a \wedge k\geq b \wedge k\geq c+3)$

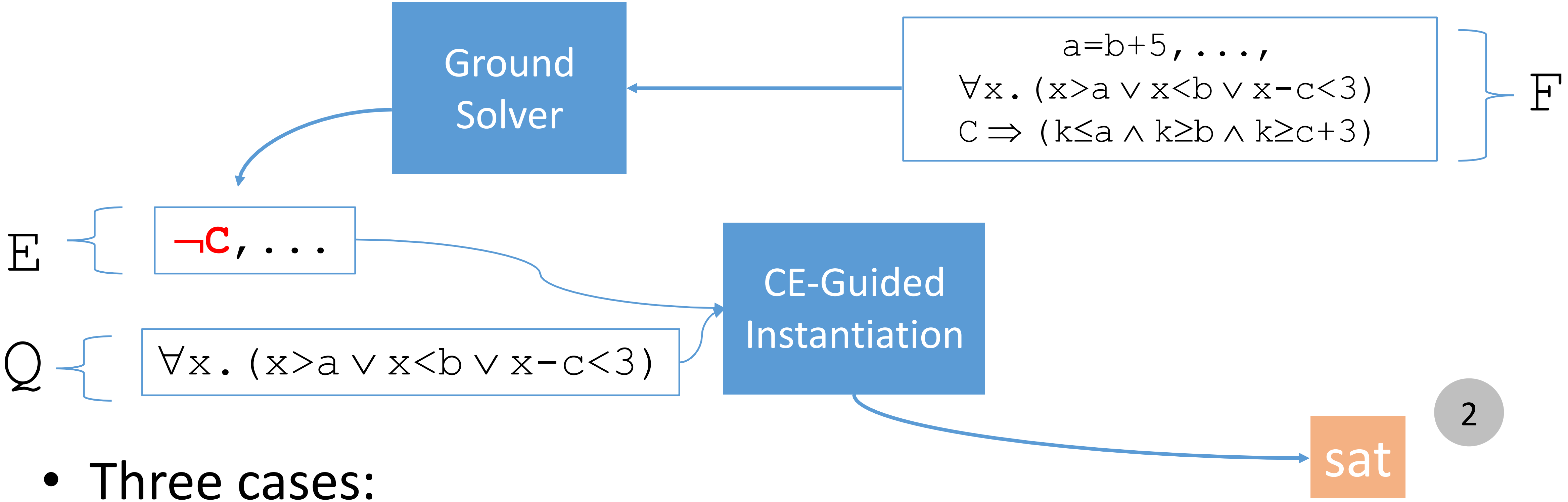
F

CE-Guided Instantiation

- Three cases:

  1. **F is unsatisfiable** ⇒ **answer "unsat"**

Counterexample-Guided Instantiation



Ground Solver

$a=b+5, \ldots,$

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

$C \Rightarrow (k\leq a \land k\geq b \land k\geq c+3)$

F

E

$\neg C, \ldots$

Q

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

CE-Guided Instantiation

sat

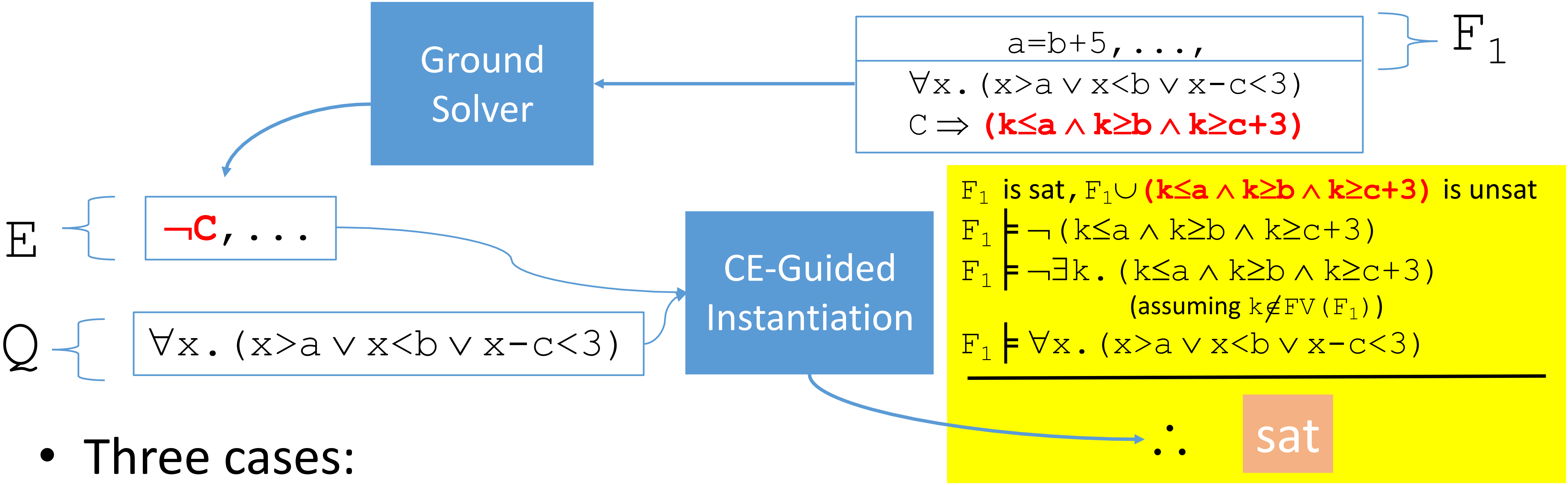
2

- Three cases:

2. F is satisfiable, ¬C∈E for *all* assignments E ⇒ answer “sat”

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5, \ldots,$

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

$C \Rightarrow \mathbf{(k\leq a \wedge k\geq b \wedge k\geq c+3)}$

F_1

E $\neg C, \ldots$

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

CE-Guided Instantiation

F_1 is sat, $F_1 \cup \mathbf{(k\leq a \wedge k\geq b \wedge k\geq c+3)}$ is unsat

$F_1 \models \neg(k\leq a \wedge k\geq b \wedge k\geq c+3)$

$F_1 \models \neg\exists k.(k\leq a \wedge k\geq b \wedge k\geq c+3)$

(assuming $k \notin FV(F_1)$)

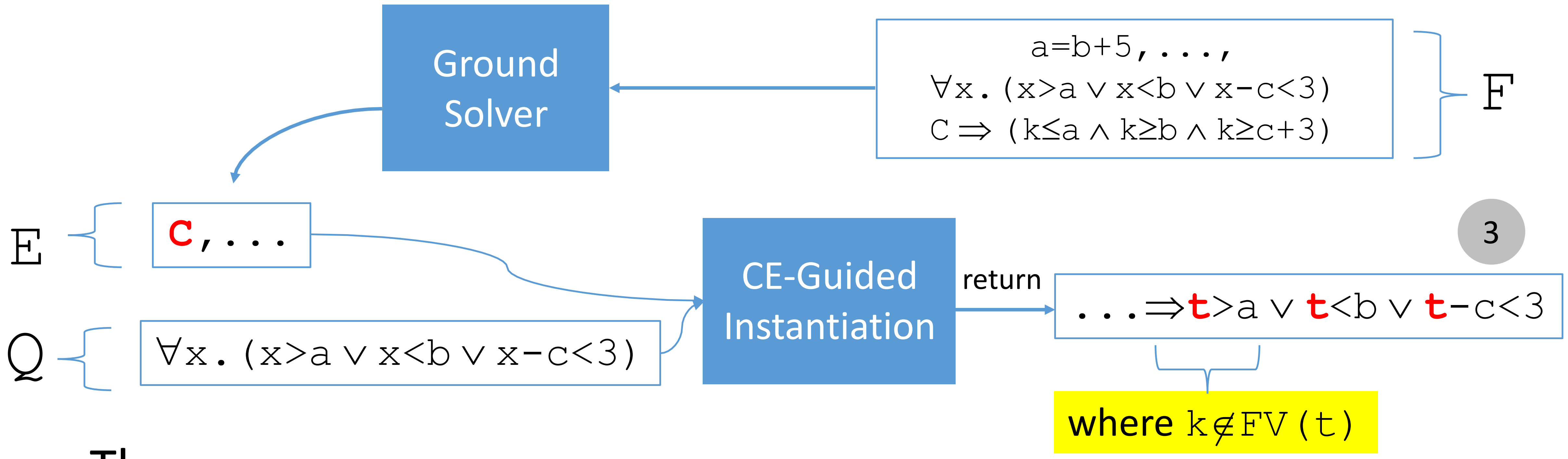
$F_1 \models \forall x.(x>a \vee x<b \vee x-c<3)$

$\therefore$ sat

- Three cases:

2. F is satisfiable, ¬C∈E for *all* assignments E ⇒ answer "sat"

Counterexample-Guided Instantiation



Ground Solver

$a=b+5, \ldots,$
$\forall x.(x>a \lor x<b \lor x-c<3)$
$C \Rightarrow (k \le a \land k \ge b \land k \ge c+3)$

F

E

$C, \ldots$

Q

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

CE-Guided Instantiation

return

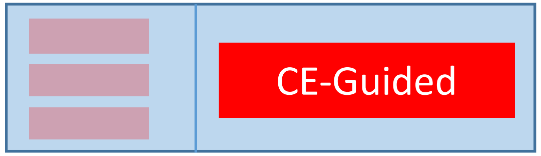
3

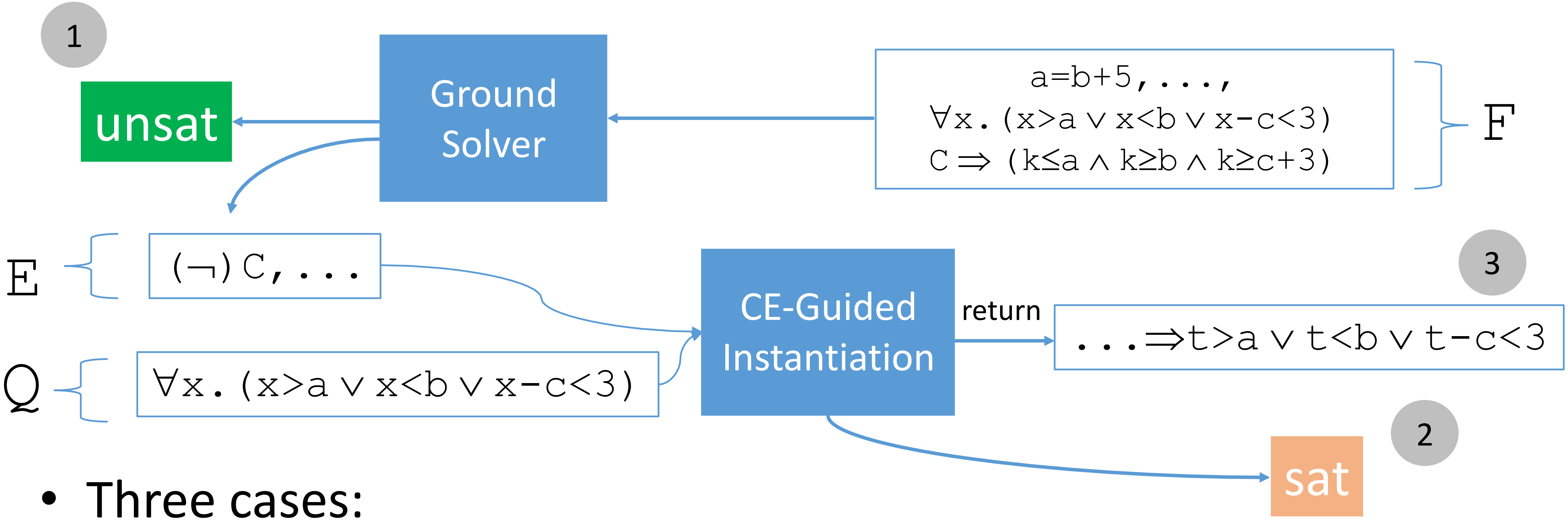
$\ldots \Rightarrow \underbrace{t>a}_{\text{where } k \notin FV(t)} \lor t<b \lor t-c<3$

- Three cases:

3. F is satisfiable, C∈E for *some* assignment E ⇒ add an instance to F

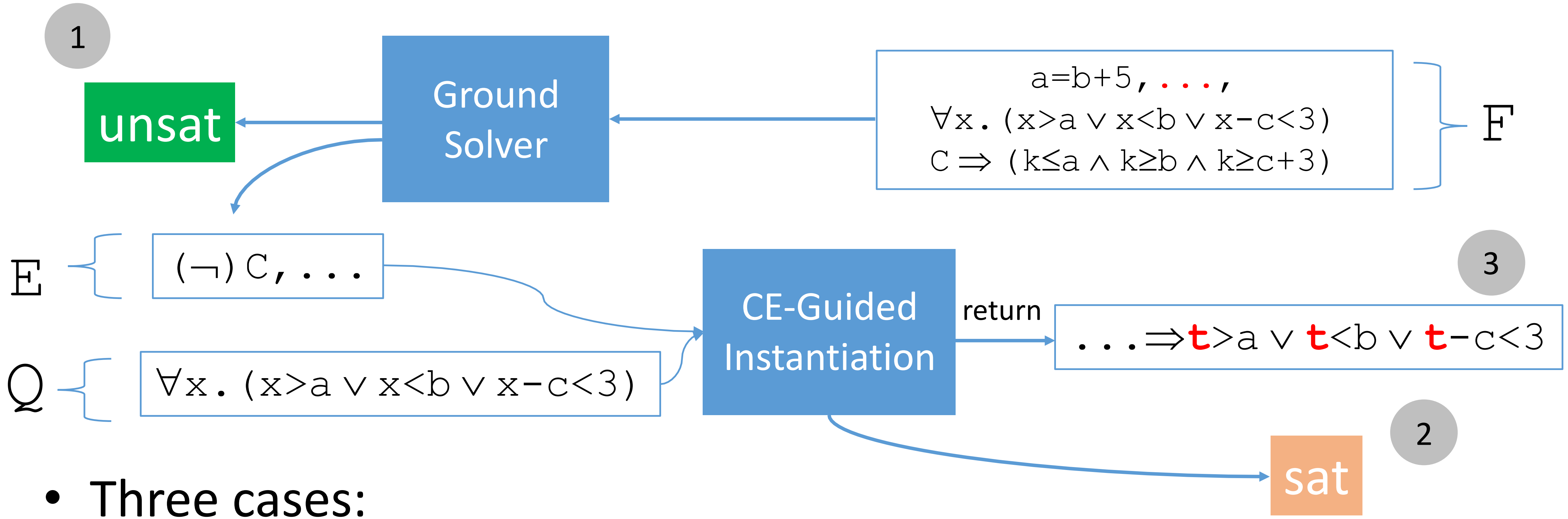
Counterexample-Guided Instantiation





- Three cases:
  1. F is unsatisfiable ⇒ answer "unsat"
  2. F is satisfiable, ¬C∈E for all assignments E ⇒ answer "sat"
  3. F is satisfiable, C∈E for some assignment E ⇒ add an instance to F

Counterexample-Guided Instantiation

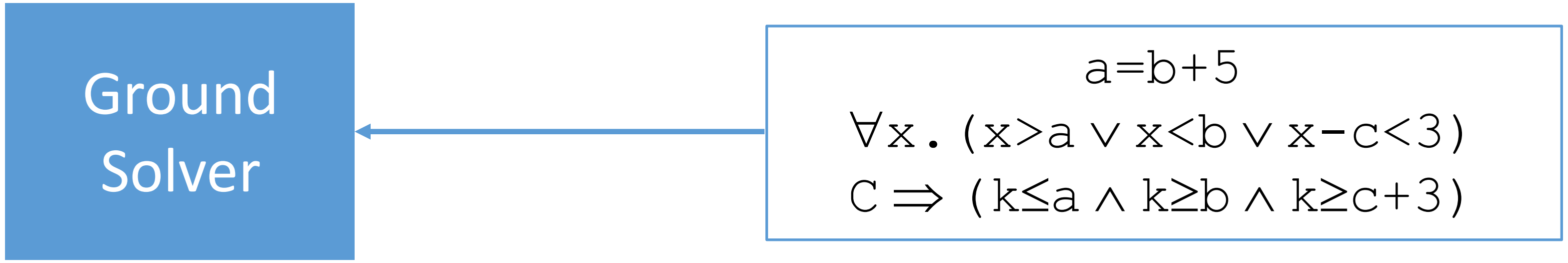


- Three cases:

  1. F is unsatisfiable ⇒ answer “unsat”
  2. F is satisfiable, $\neg C \in E$ for all assignments E ⇒ answer “sat”
  3. F is satisfiable, $C \in E$ for some assignment E ⇒ add **an instance** to F (…which **t**?)

Counterexample-Guided Instantiation

CE-Guided



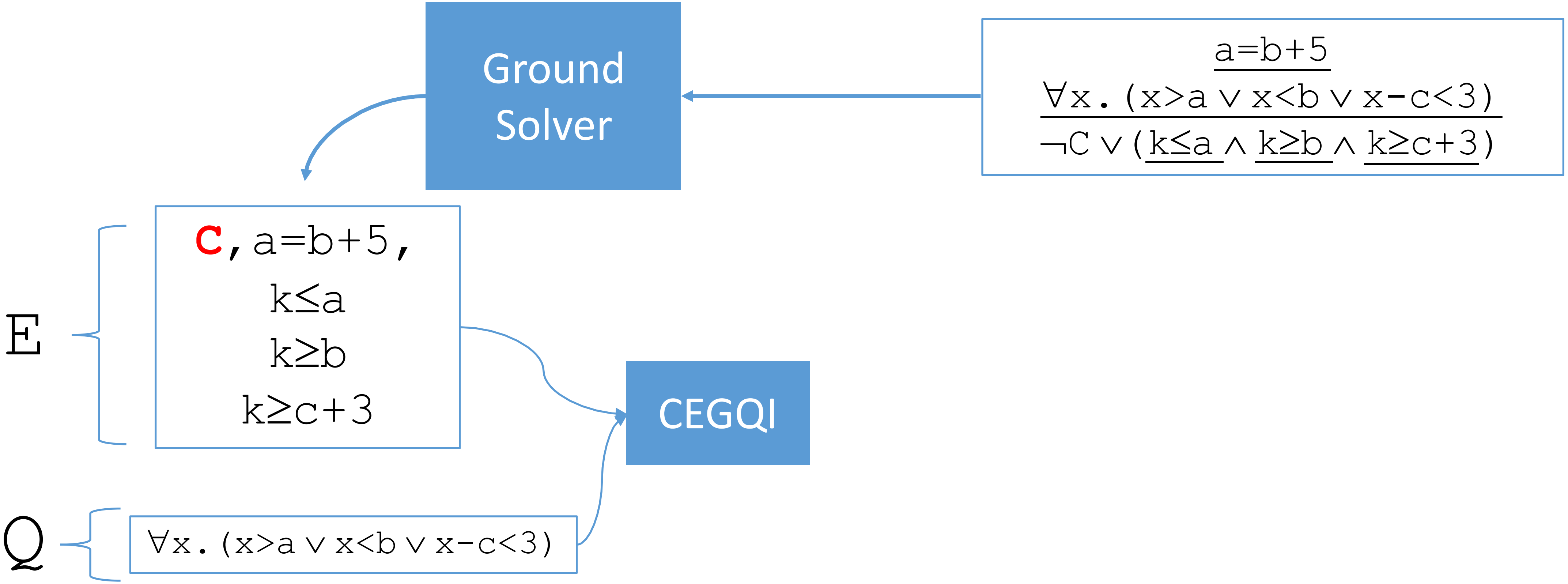
$a=b+5$

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

$C \Rightarrow (k\leq a \wedge k\geq b \wedge k\geq c+3)$

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$\underline{a=b+5}$

$\underline{\forall x.\,(x>a \lor x<b \lor x-c<3)}$

$\neg C \lor (\underline{k\leq a} \land \underline{k\geq b} \land \underline{k\geq c+3})$

E

$C,\ a=b+5,$

$k\leq a$

$k\geq b$

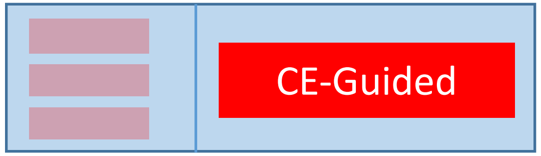
$k\geq c+3$

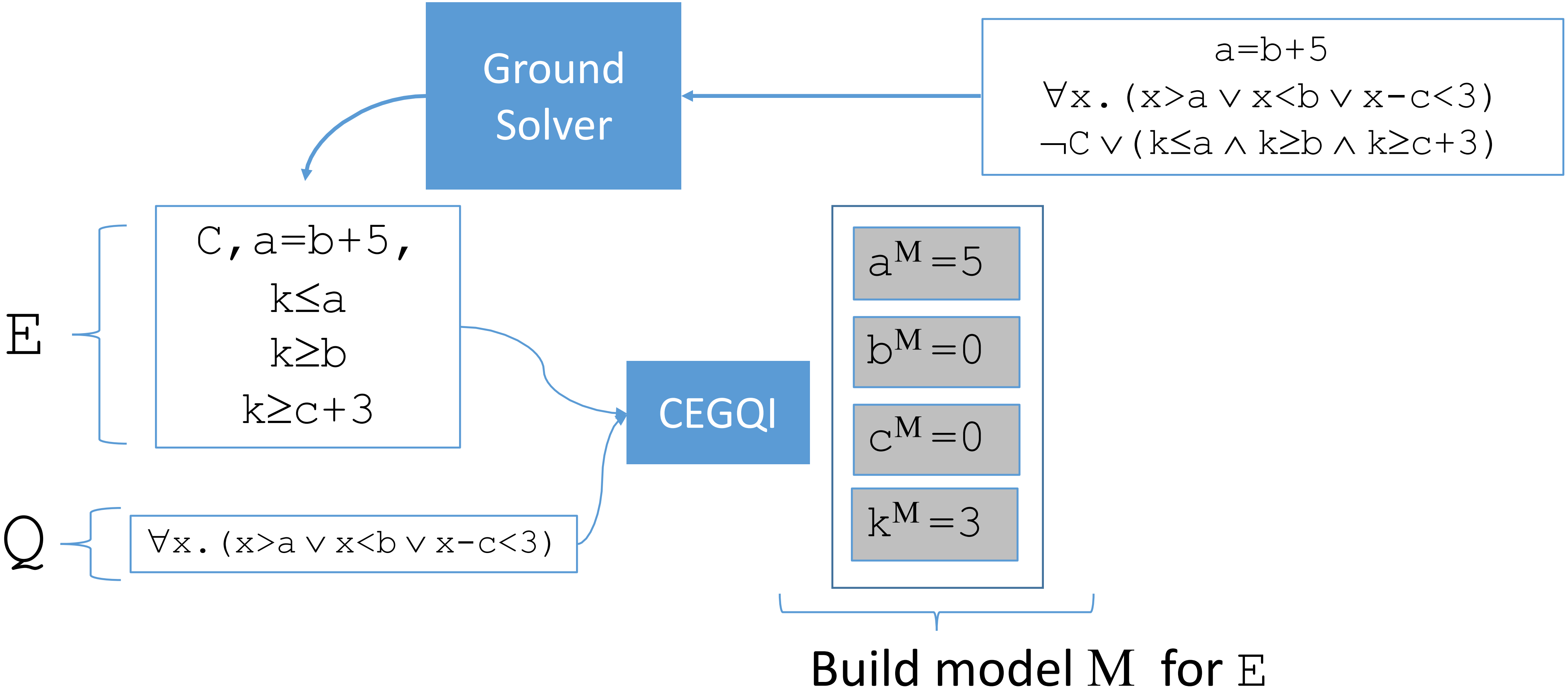
CEGQI

Q

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

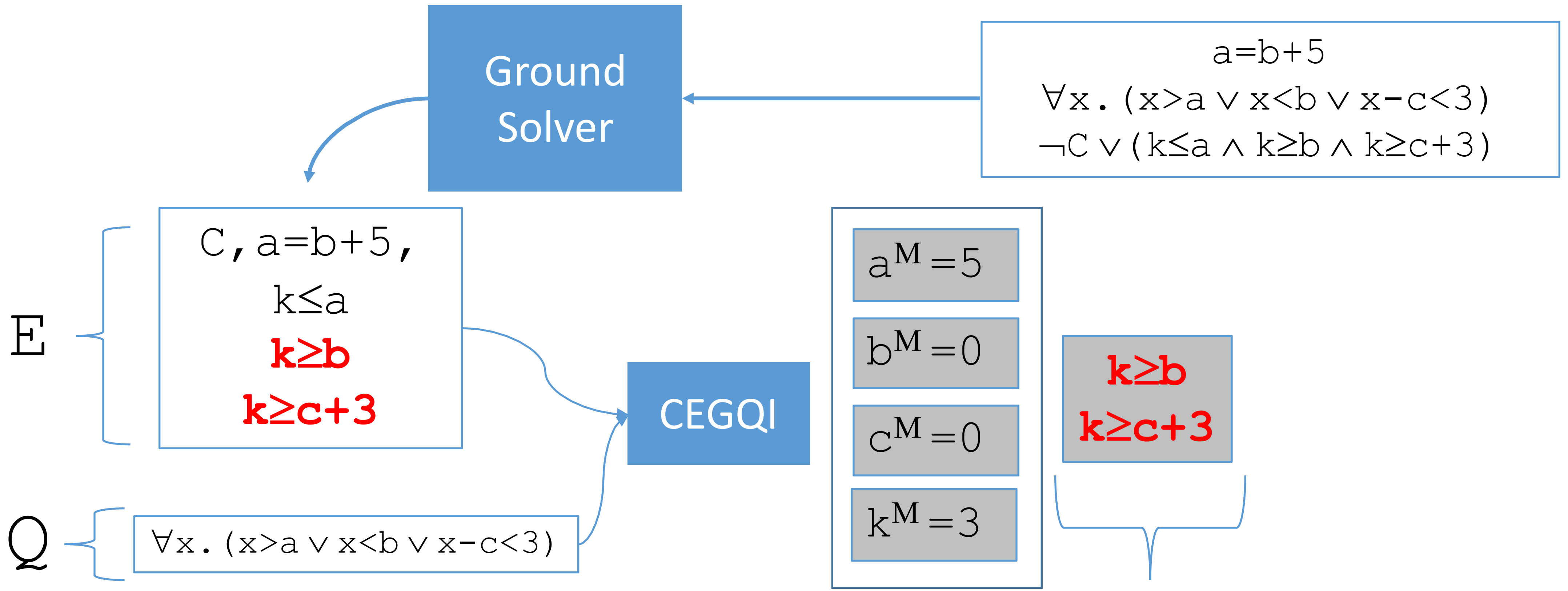
Counterexample-Guided Instantiation




Ground Solver
$a=b+5$
$\forall x.(x>a \lor x<b \lor x-c<3)$
$\neg C \lor (k\leq a \land k\geq b \land k\geq c+3)$
E
$C, a=b+5,$
$k\leq a$
$k\geq b$
$k\geq c+3$
CEGQI
Q
$\forall x.(x>a \lor x<b \lor x-c<3)$
$a^M=5$
$b^M=0$
$c^M=0$
$k^M=3$
Build model M for E

Counterexample-Guided Instantiation

CE-Guided



$a=b+5$

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

$\neg C \lor (k\leq a \land k\geq b \land k\geq c+3)$

$C, a=b+5,$

$k\leq a$

$k\geq b$

$k\geq c+3$

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

$a^M=5$

$b^M=0$

$c^M=0$

$k^M=3$

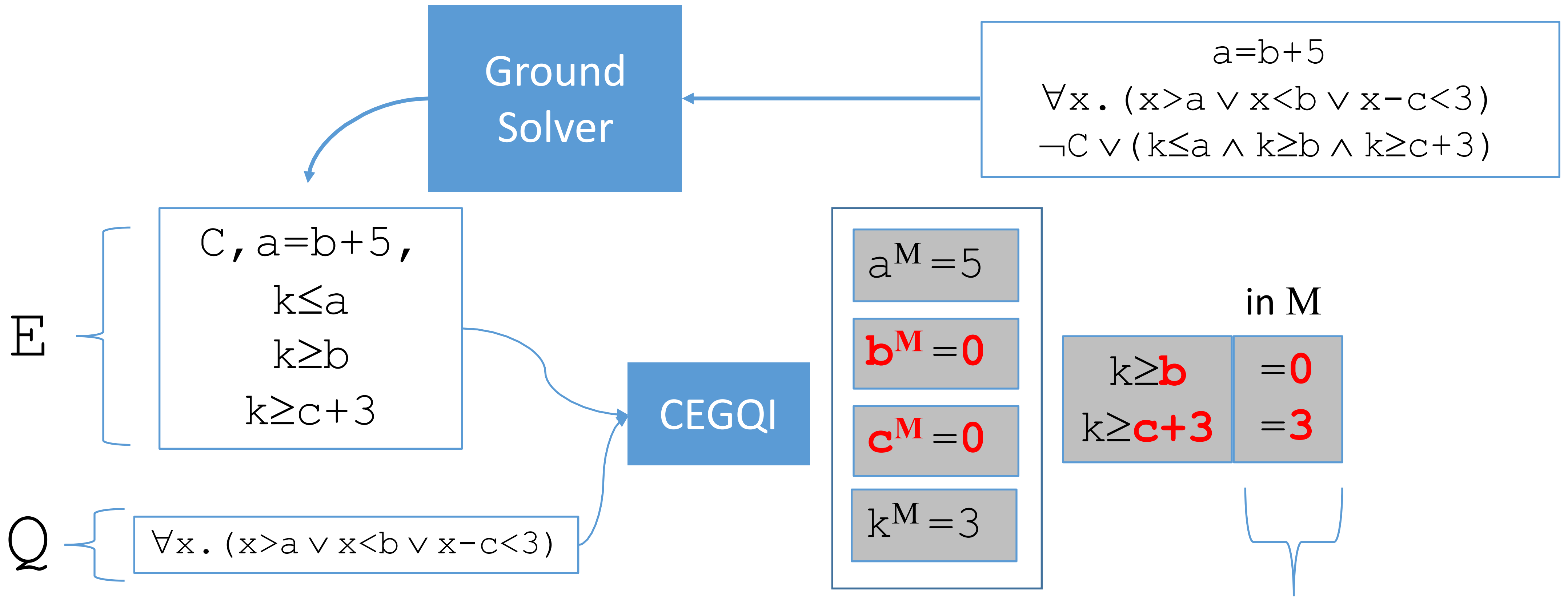
$k\geq b$

$k\geq c+3$

Take lower bounds of k in E

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

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

$\neg C \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

E

$C, a=b+5,$

$k\leq a$

$k\geq b$

$k\geq c+3$

Q

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

CEGQI

$a^M=5$

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

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

$k^M=3$

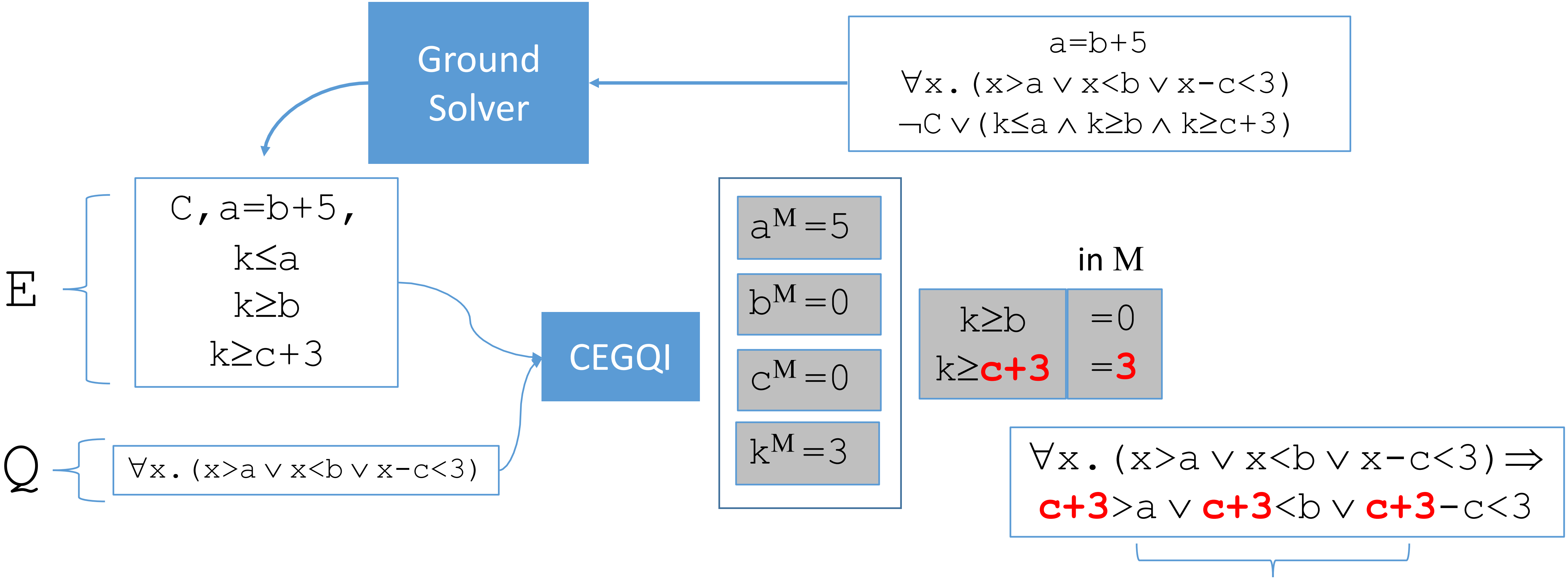
in M

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

Compute their value in M

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

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

$\neg C \lor (k\leq a \land k\geq b \land k\geq c+3)$

E

$C, a=b+5,$

$k\leq a$

$k\geq b$

$k\geq c+3$

CEGQI

Q

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

$a^M=5$

$b^M=0$

$c^M=0$

$k^M=3$

in M

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

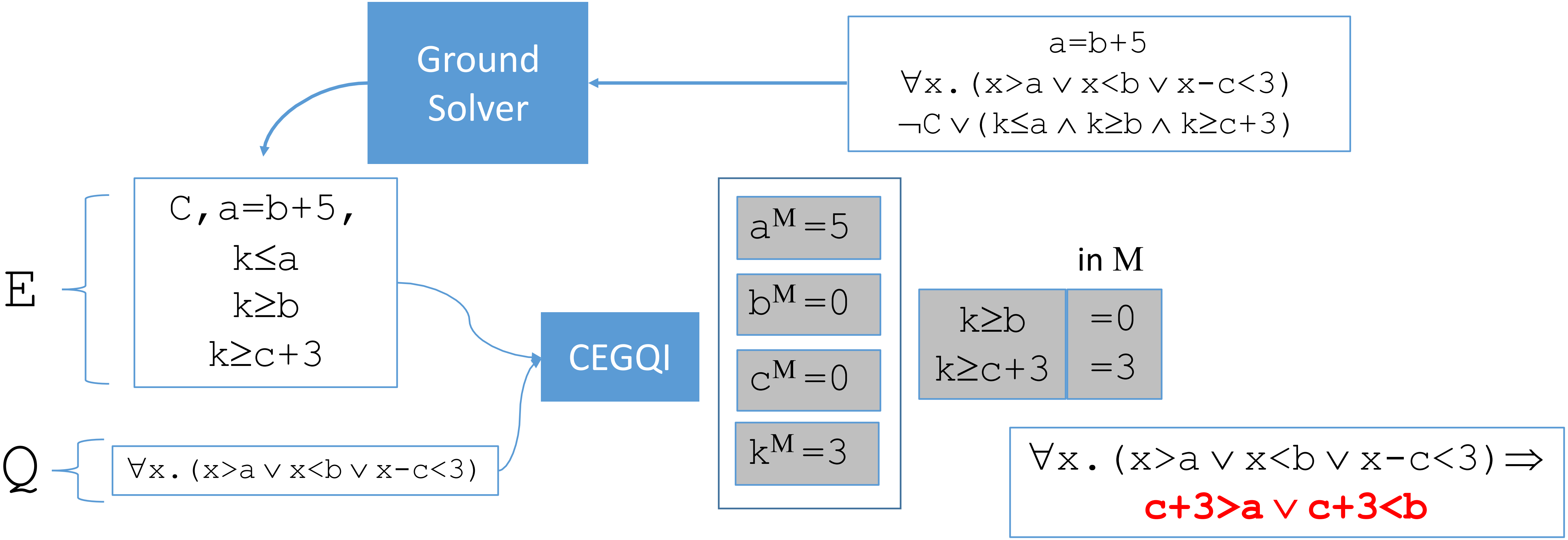
$\forall x.(x>a \lor x<b \lor x-c<3) \Rightarrow$

$c+3>a \lor c+3<b \lor c+3-c<3$

Add instance for lower bound that is maximal in M

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

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

$\neg C \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

E

$C, a=b+5,$

$k\leq a$

$k\geq b$

$k\geq c+3$

Q

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

CEGQI

$a^M=5$

$b^M=0$

$c^M=0$

$k^M=3$

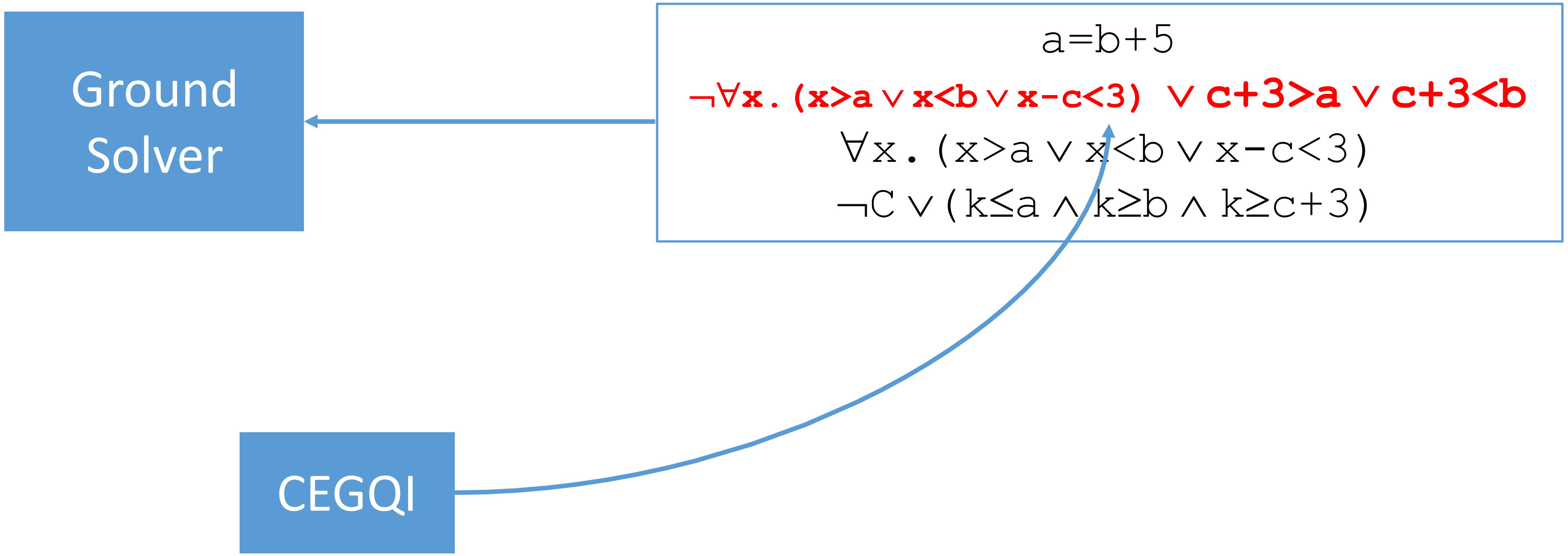
in M

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

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

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

Counterexample-Guided Instantiation



Ground Solver

$a=b+5$

$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor c+3>a \lor c+3<b$

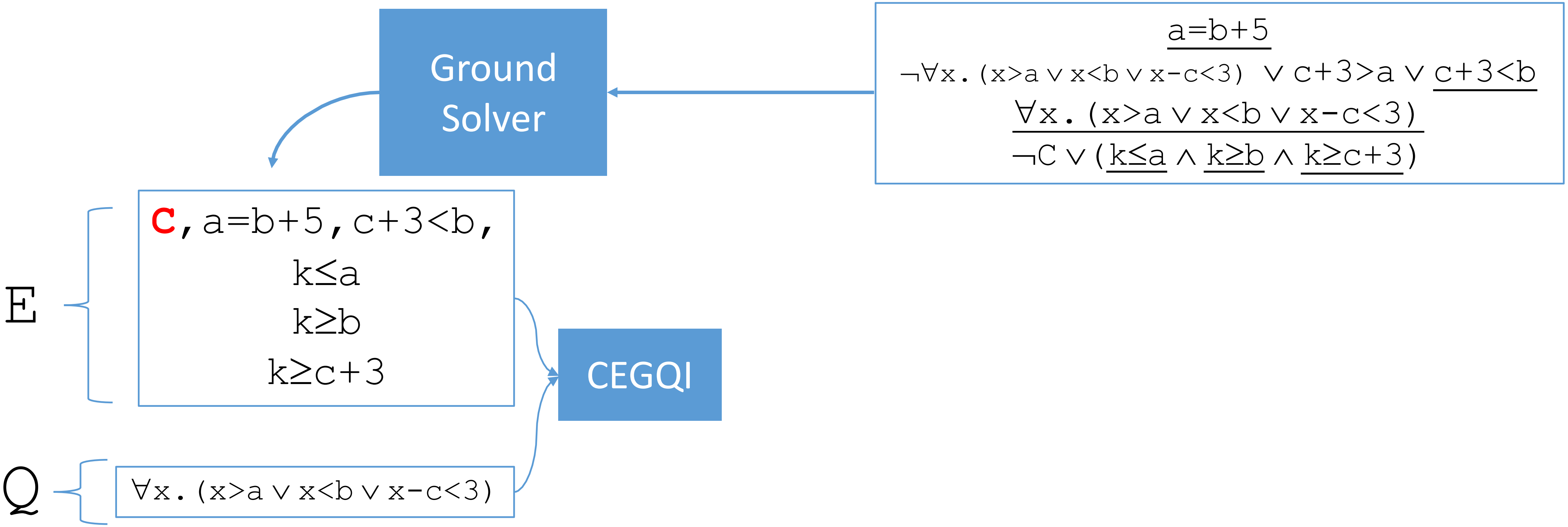
$\forall x.(x>a \lor x<b \lor x-c<3)$

$\neg C \lor (k\leq a \land k\geq b \land k\geq c+3)$

CEGQI

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$\underline{a=b+5}$

$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor c+3>a \lor \underline{c+3<b}$

$\underline{\forall x.(x>a \lor x<b \lor x-c<3)}$

$\neg C \lor (\underline{k\leq a} \land \underline{k\geq b} \land \underline{k\geq c+3})$

E

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

$k\leq a$

$k\geq b$

$k\geq c+3$

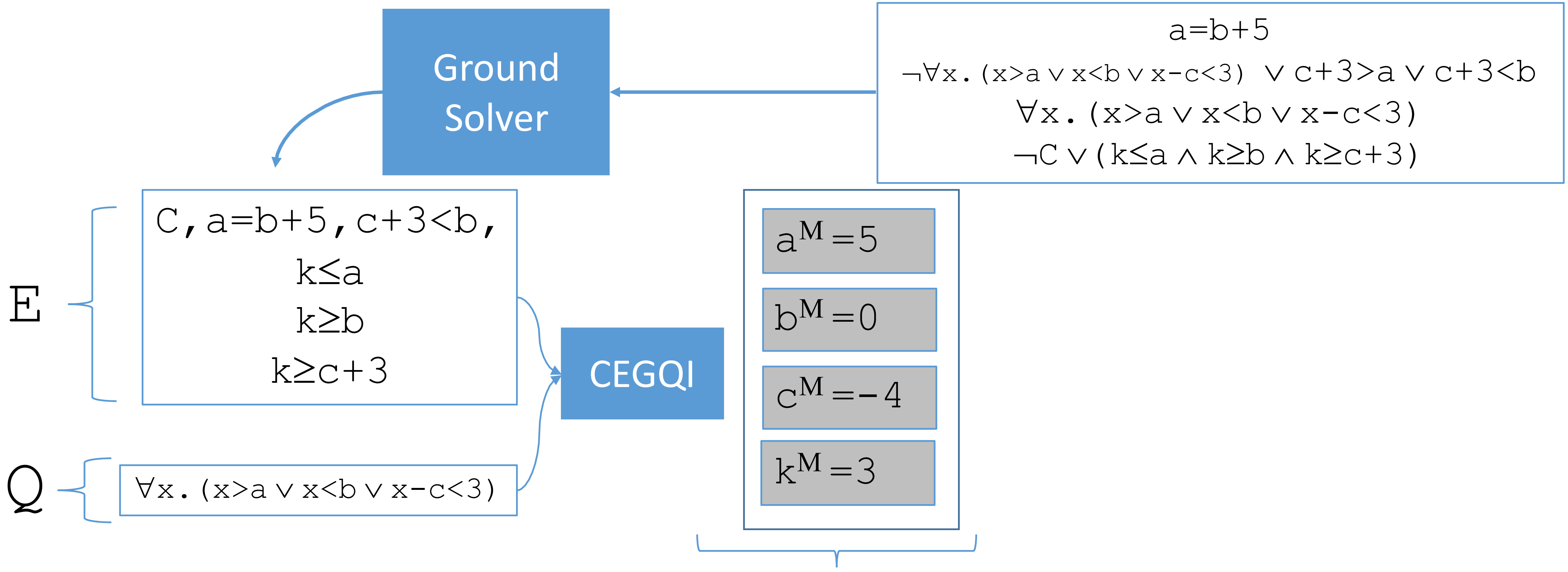
CEGQI

Q

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

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

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

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

$\neg C \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

E

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

$k\leq a$

$k\geq b$

$k\geq c+3$

Q

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

CEGQI

$a^M=5$

$b^M=0$

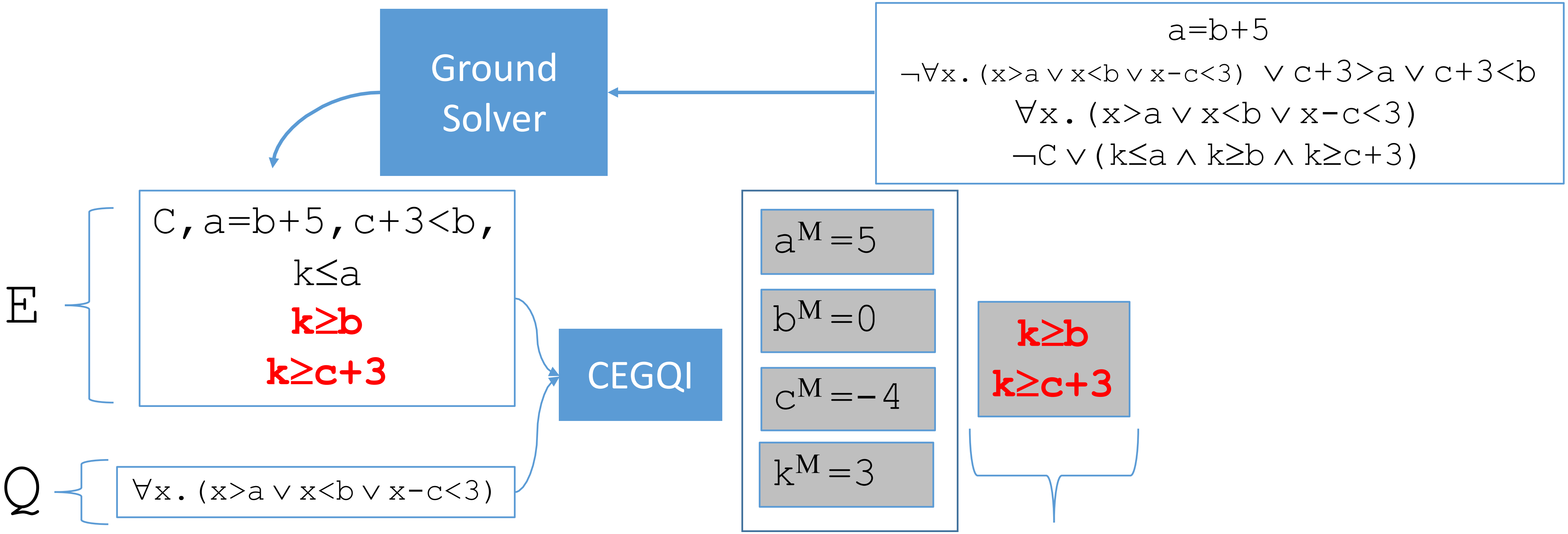
$c^M=-4$

$k^M=3$

Build model M for E

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor c+3>a \lor c+3<b$

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

$\neg C \lor (k\leq a \land k\geq b \land k\geq c+3)$

E

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

$k\leq a$

$\mathbf{k\geq b}$

$\mathbf{k\geq c+3}$

Q

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

CEGQI

$a^M=5$

$b^M=0$

$c^M=-4$

$k^M=3$

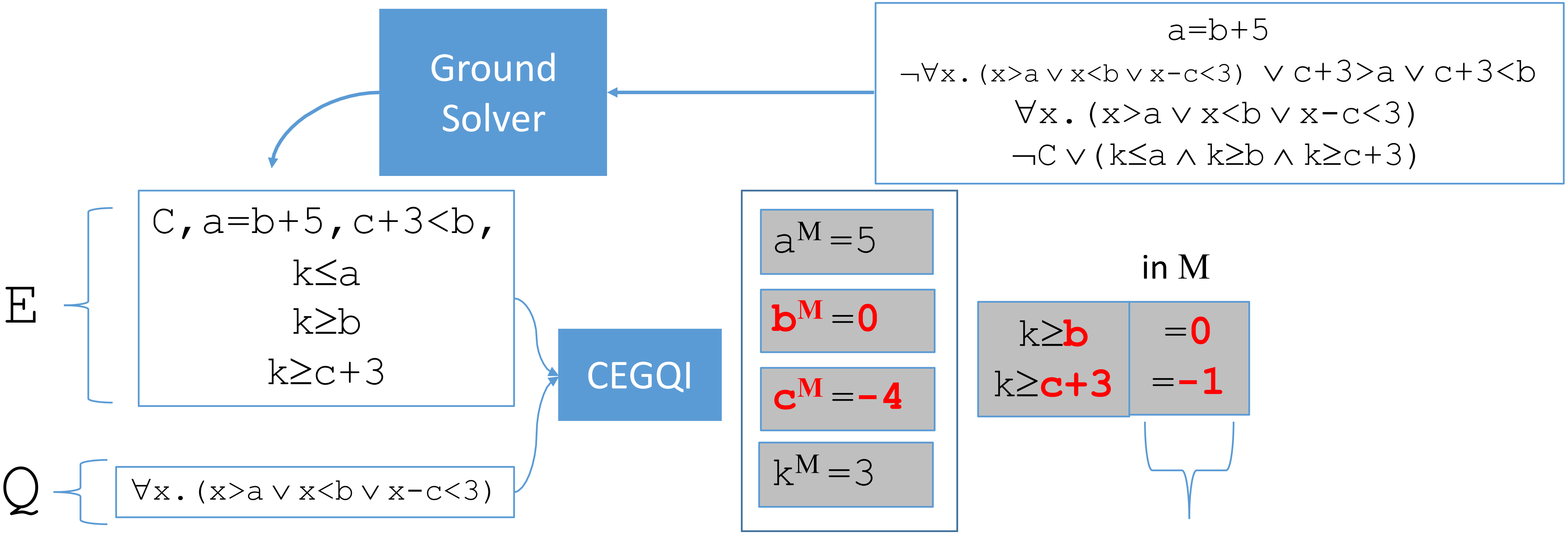
$\mathbf{k\geq b}$

$\mathbf{k\geq c+3}$

Take lower bounds of k in E

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

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

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

$\neg C \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

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

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

CEGQI

$a^M=5$

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

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

$k^M=3$

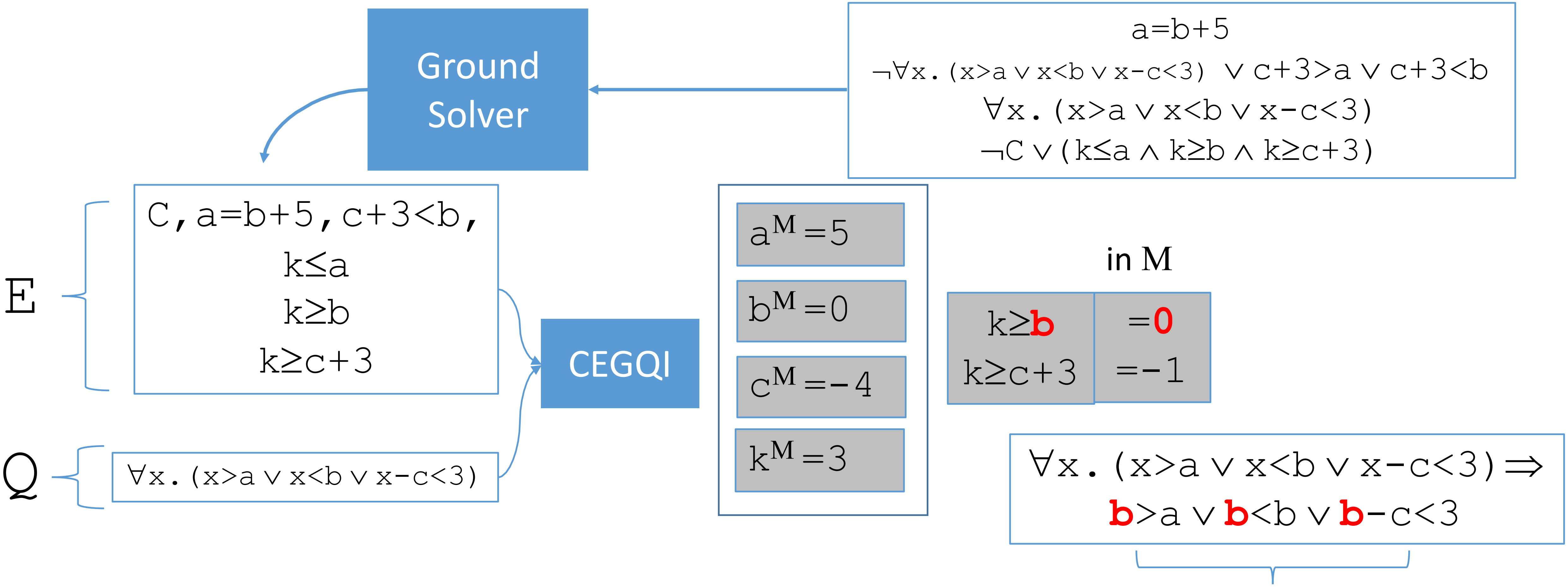
in M

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

Compute their value in M

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$a=b+5$

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

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

$\neg C \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

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

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

CEGQI

$a^M=5$

$b^M=0$

$c^M=-4$

$k^M=3$

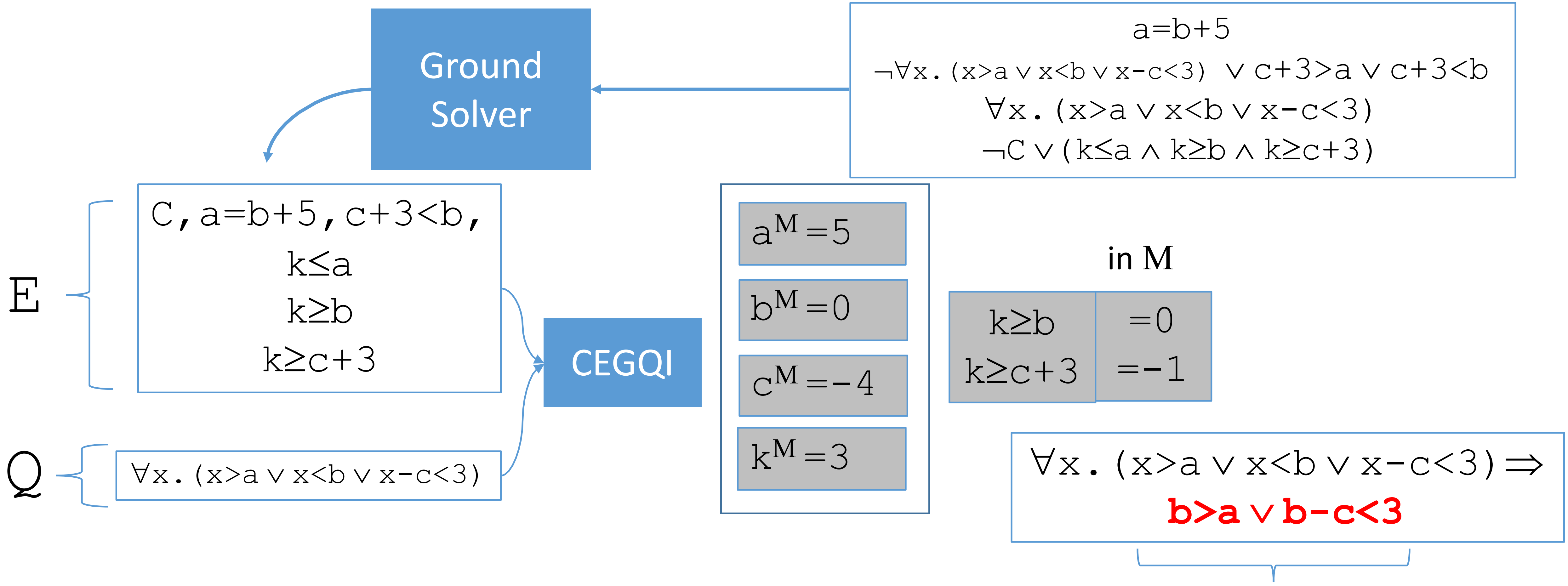
in M

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

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

Add instance for lower bound that is maximal in M

Counterexample-Guided Instantiation



Ground Solver

$a=b+5$

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

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

$\neg C \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

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

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

CEGQI

$a^M=5$

$b^M=0$

$c^M=-4$

$k^M=3$

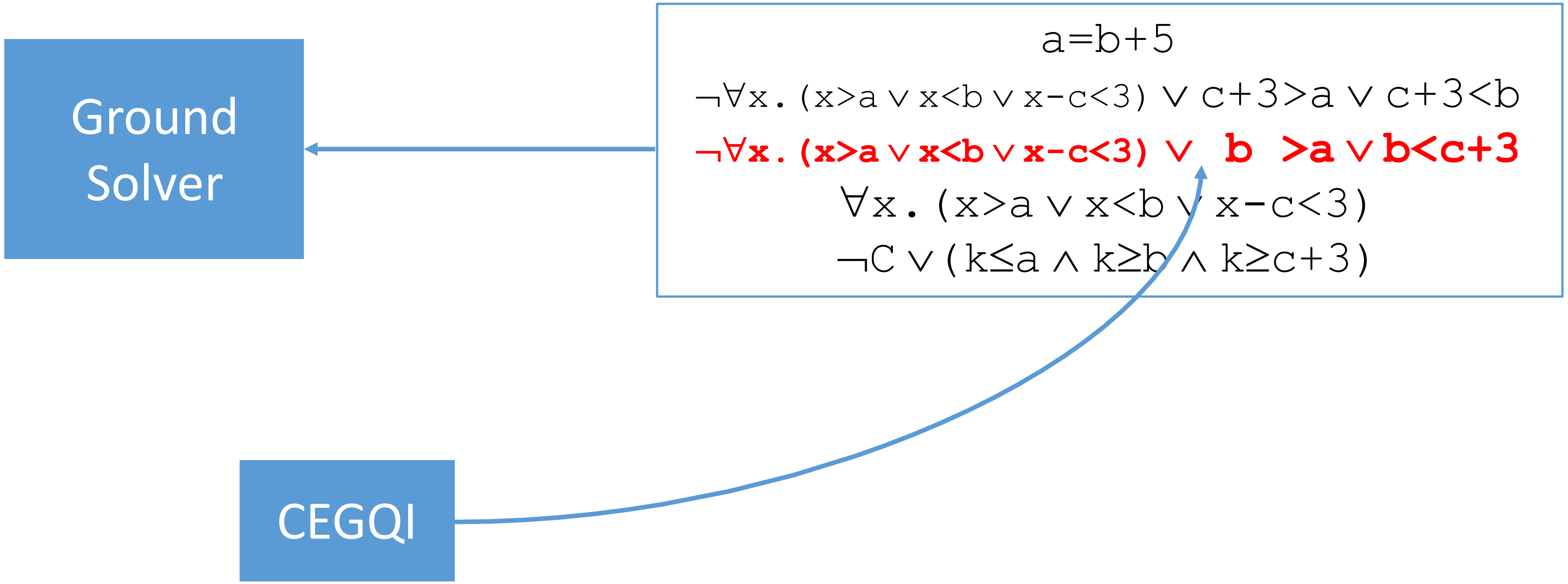
in M

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

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

Add instance for lower bound that is maximal in M

Counterexample-Guided Instantiation



Ground Solver

$a=b+5$

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

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

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

$\neg C \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

CEGQI

Counterexample-Guided Instantiation

Ground Solver

$\underline{a=b+5}$

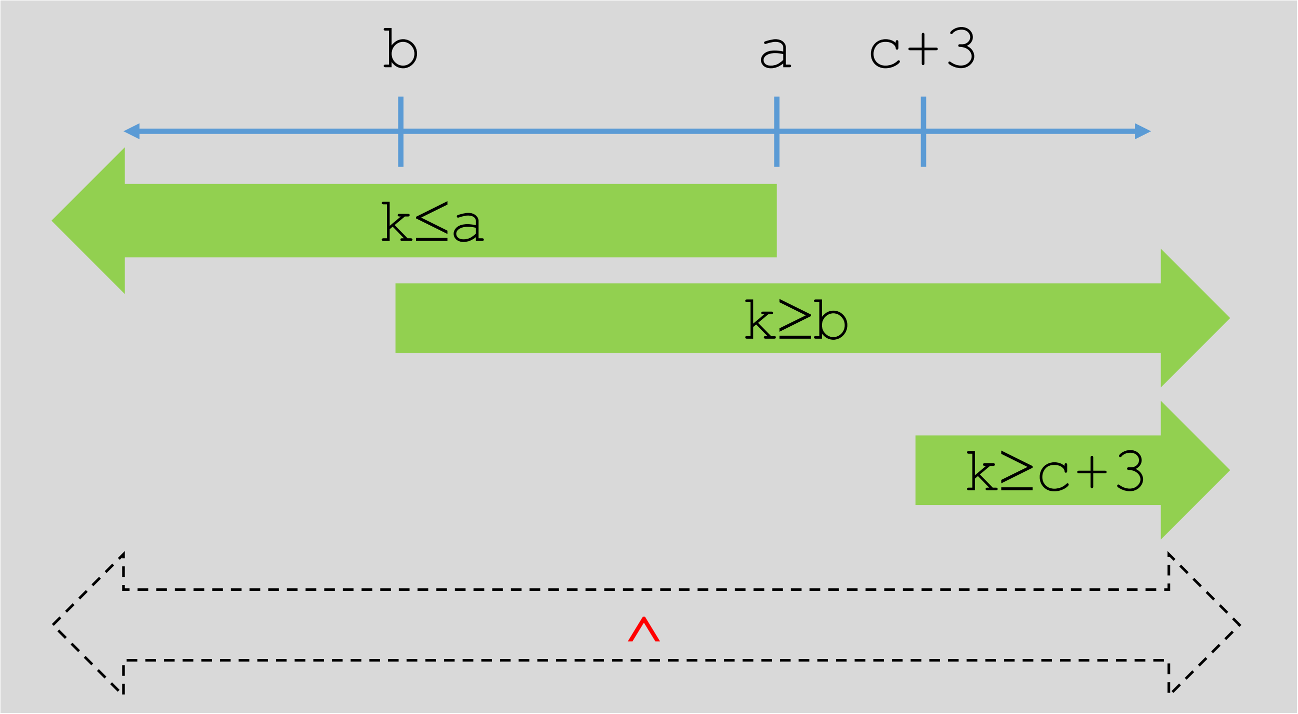
$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor \underline{c+3>a} \lor c+3<b$

$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor \; b \; >a \lor \underline{b<c+3}$

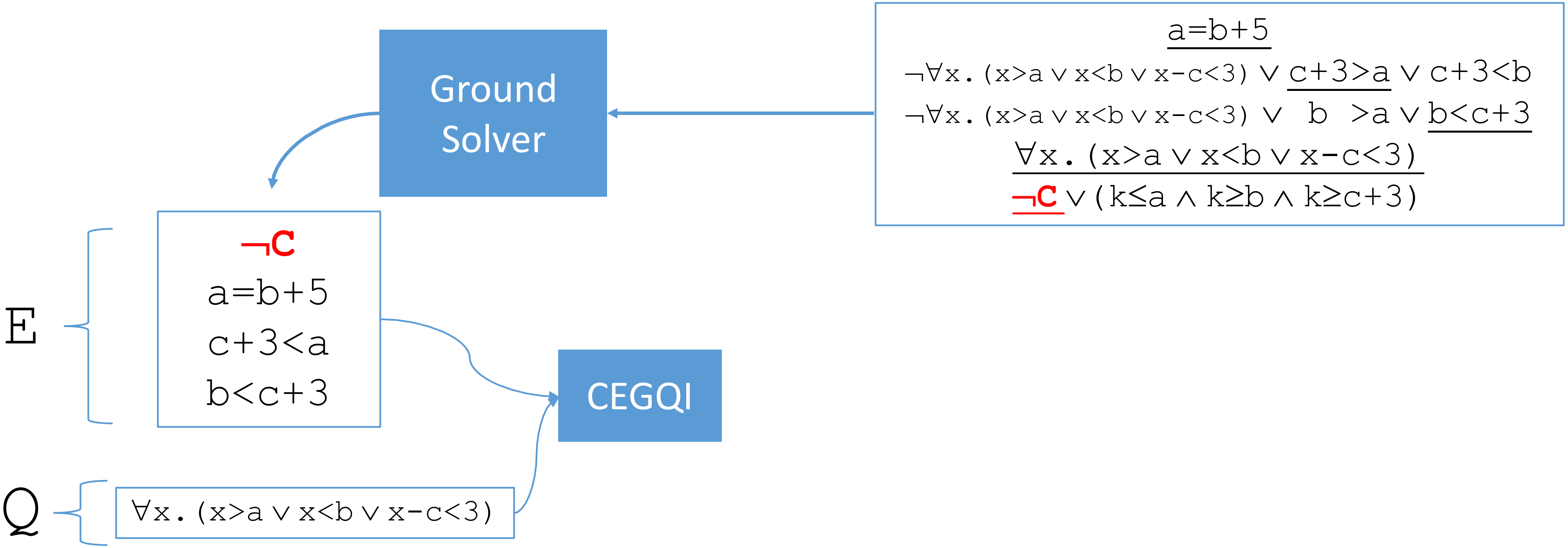
$\underline{\forall x.(x>a \lor x<b \lor x-c<3)}$

$\neg C \lor (\mathbf{k\leq a} \land \mathbf{k\geq b} \land \mathbf{k\geq c+3})$

CEGQI



Counterexample-Guided Instantiation



$\underline{a=b+5}$

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

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

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

$\underline{\neg C} \vee (k\leq a \wedge k\geq b \wedge k\geq c+3)$

$\neg C$

$a=b+5$

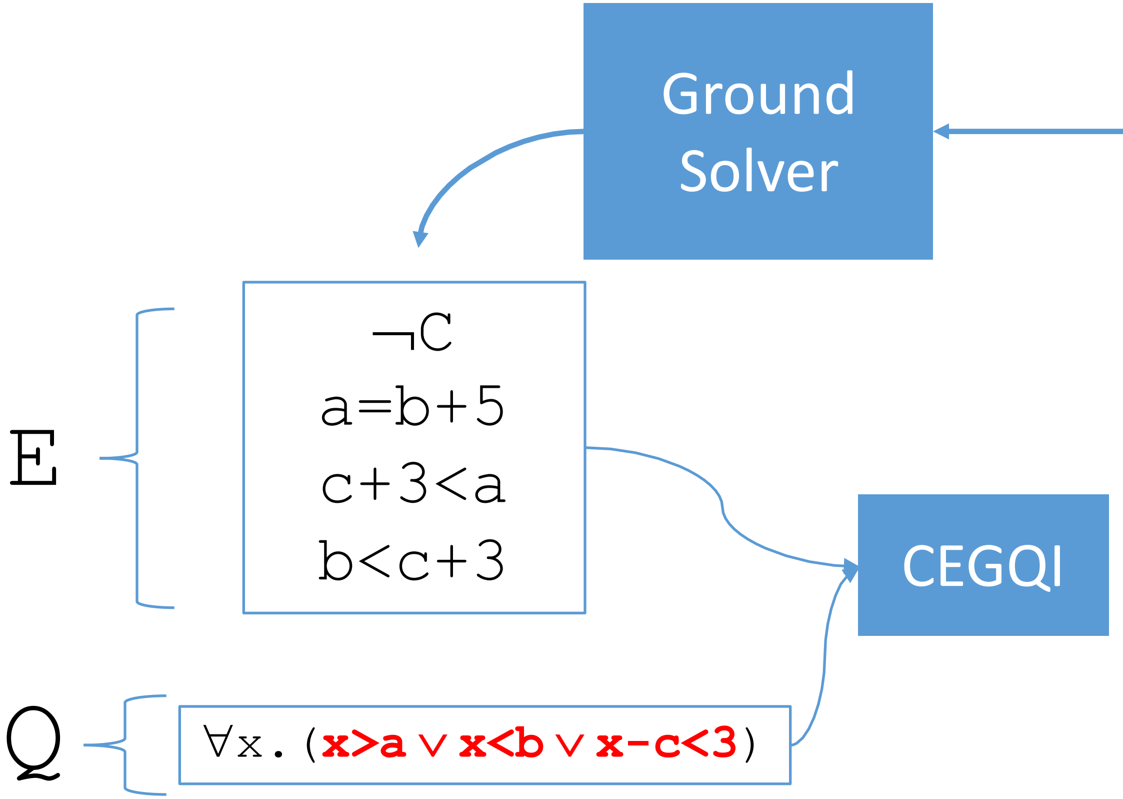
$c+3<a$

$b<c+3$

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

Counterexample-Guided Instantiation

CE-Guided



Ground Solver

E

$\neg C$

$a=b+5$

$c+3<a$

$b<c+3$

CEGQI

Q

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

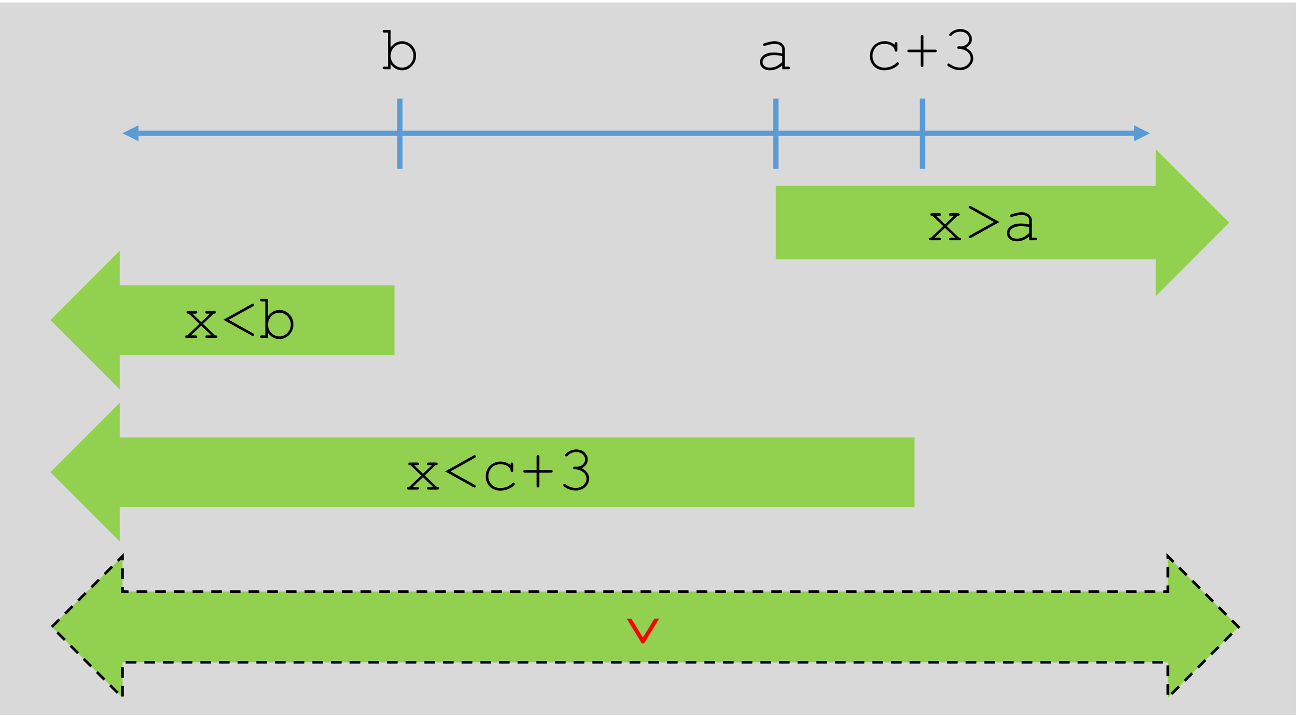
$$\underline{a=b+5}$$

$$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor \underline{c+3>a} \lor c+3<b$$

$$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor b>a \lor \underline{b<c+3}$$

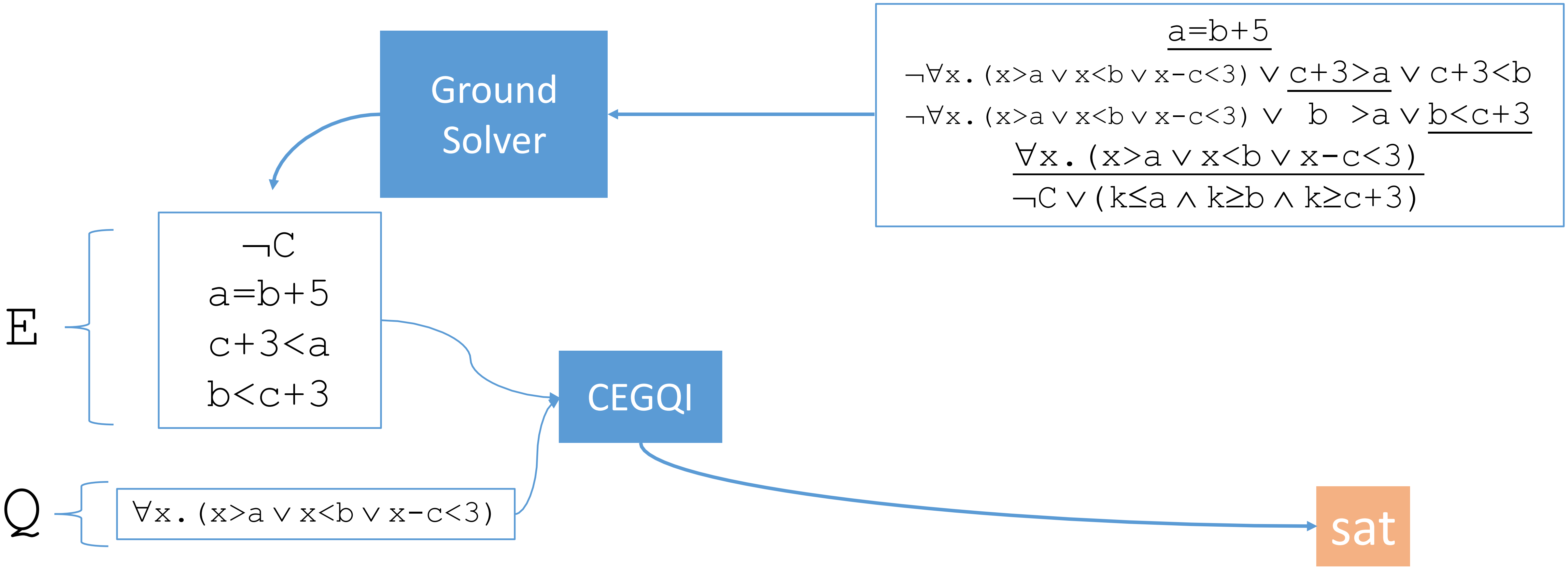
$$\underline{\forall x.(x>a \lor x<b \lor x-c<3)}$$

$$\neg C \lor (k\leq a \land k\geq b \land k\geq c+3)$$



Counterexample-Guided Instantiation

CE-Guided



Ground Solver

$\underline{a=b+5}$

$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor \underline{c+3>a} \lor c+3<b$

$\neg\forall x.(x>a \lor x<b \lor x-c<3) \lor b>a \lor \underline{b<c+3}$

$\underline{\forall x.(x>a \lor x<b \lor x-c<3)}$

$\neg C \lor (k\leq a \land k\geq b \land k\geq c+3)$

E

$\neg C$

$a=b+5$

$c+3<a$

$b<c+3$

Q

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

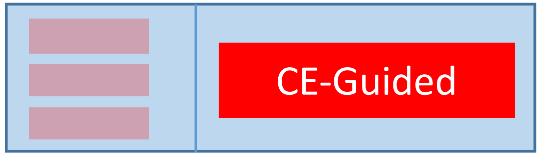
CEGQI

sat

$\Rightarrow \exists abc.(a=b+5 \land \forall x.(x>a \lor x<b \lor x-c<3))$

is LIA-satisfiable

Counterexample-Guided Instantiation

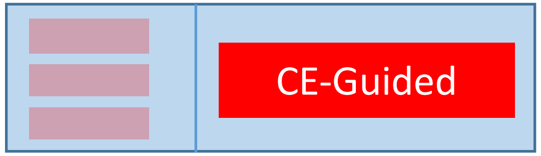


- Decision procedure for ∀ in various theories:
  - Linear real arithmetic (LRA)
    - Maximal lower (minimal upper) bounds $l_1 < k, \ldots, l_n < k \rightarrow \{x \rightarrow l_{max} + \delta\}$
      - [Loos+Wiespfenning 93]

      …may involve virtual terms δ, ∞
    - Interior point method: $l_{max} < k < u_{min} \rightarrow \{x \rightarrow (l_{max} - u_{min})/2\}$
      - [Ferrante+Rackoff 79]
  - Linear integer arithmetic (LIA)
    - Maximal lower (minimal upper) bounds (+c) $l_1 < k, \ldots, l_n < k \rightarrow \{x \rightarrow l_{max} + c\}$
      - [Cooper 72]
  - Bitvectors/finite domains
    - Value instantiations $F[k] \rightarrow \{x \rightarrow k^M\}$
  - Datatypes, …

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

Counterexample-Guided Instantiation



$$\forall \mathtt{x}\,.\,\psi[\mathtt{x}]$$

- Can be used for:
  - Quantifier elimination

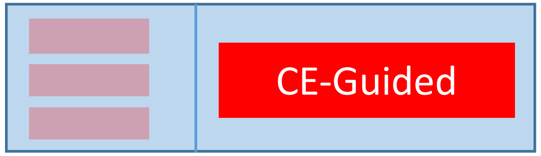
    $$\psi[\mathtt{t}_1] \wedge \ldots \wedge \psi[\mathtt{t}_n] \text{ is (un)sat}$$

    - $\exists \mathtt{x}\,.\,\neg\psi[\mathtt{x}]$ **is equivalent to** $\neg\psi[\mathtt{t}_1] \vee \ldots \vee \neg\psi[\mathtt{t}_n]$
  - Function Synthesis

    $$\psi[\mathtt{t}_1] \wedge \ldots \wedge \psi[\mathtt{t}_n] \text{ is unsat}$$

    - $\lambda\mathtt{x}.\mathtt{ite}(\psi[\mathtt{t}_1], \mathtt{t}_1, \ldots, \mathtt{ite}(\psi[\mathtt{t}_{n-1}], \mathtt{t}_{n-1}, \mathtt{t}_n)\ldots)$ **is a solution for** $\mathtt{f}$ **in** $\forall\mathtt{x}\,.\,\psi[\mathtt{f}(\mathtt{x})]$

Counterexample-Guided Instantiation

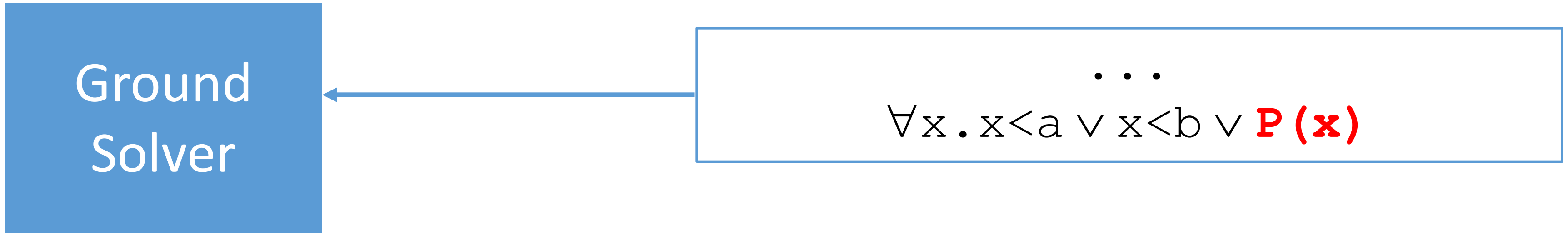


CE-Guided

- Challenge:

Counterexample-Guided Instantiation

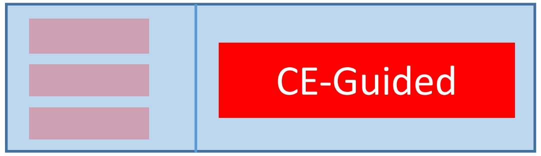
- Challenge: does not work in presence of uninterpreted functions!



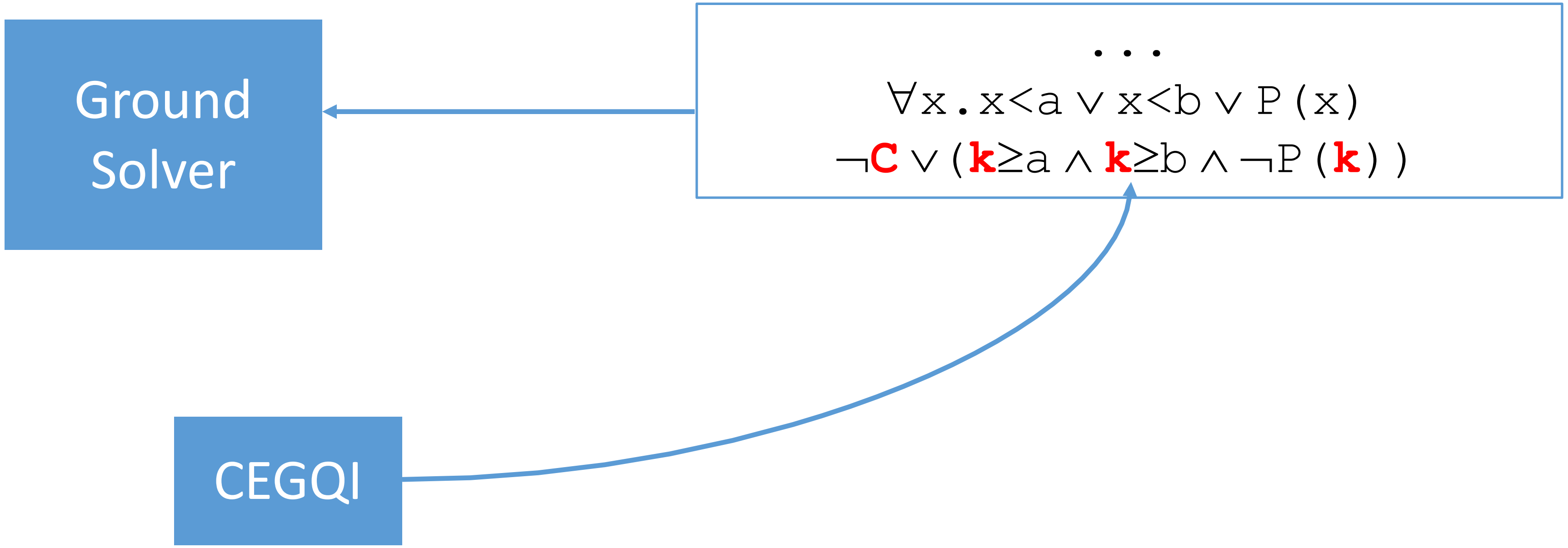
...

$\forall x. x<a \lor x<b \lor \mathbf{P(x)}$

Counterexample-Guided Instantiation



- Challenge: does not work in presence of uninterpreted functions!



...

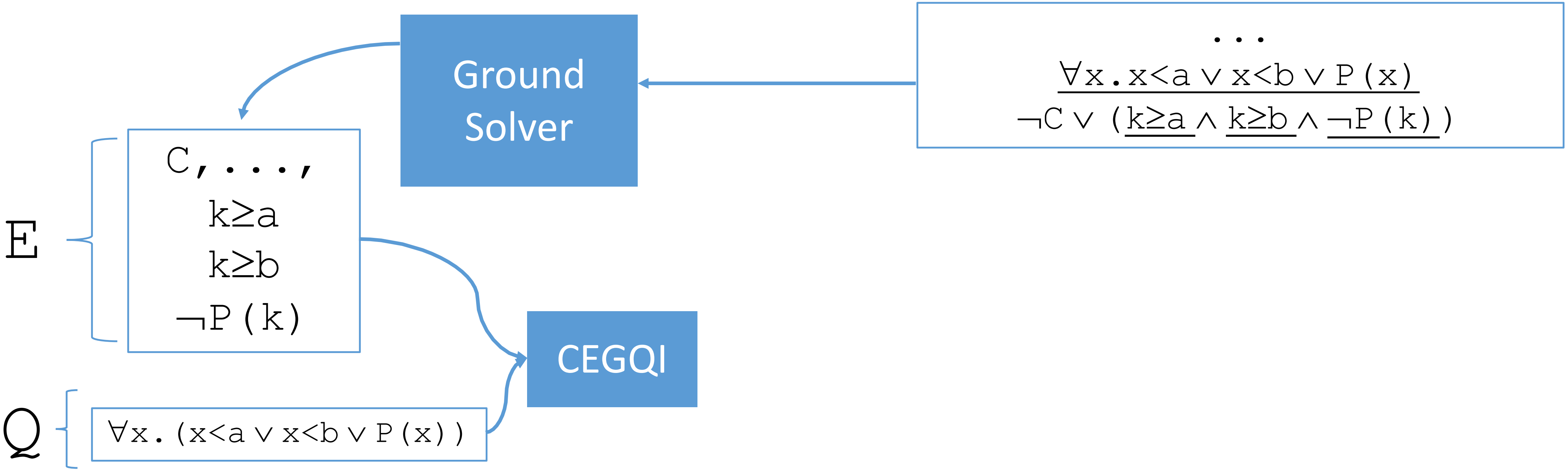
$\forall x. x<a \lor x<b \lor P(x)$

$\neg \mathbf{C} \lor (\mathbf{k} \geq a \land \mathbf{k} \geq b \land \neg P(\mathbf{k}))$

Counterexample-Guided Instantiation

CE-Guided

- Challenge: does not work in presence of uninterpreted functions!



...

$\underline{\forall x. x<a \vee x<b \vee P(x)}$

$\neg C \vee (\underline{k \geq a} \wedge \underline{k \geq b} \wedge \underline{\neg P(k)})$

Ground Solver

E

$C, \ldots,$

$k \geq a$

$k \geq b$

$\neg P(k)$

CEGQI

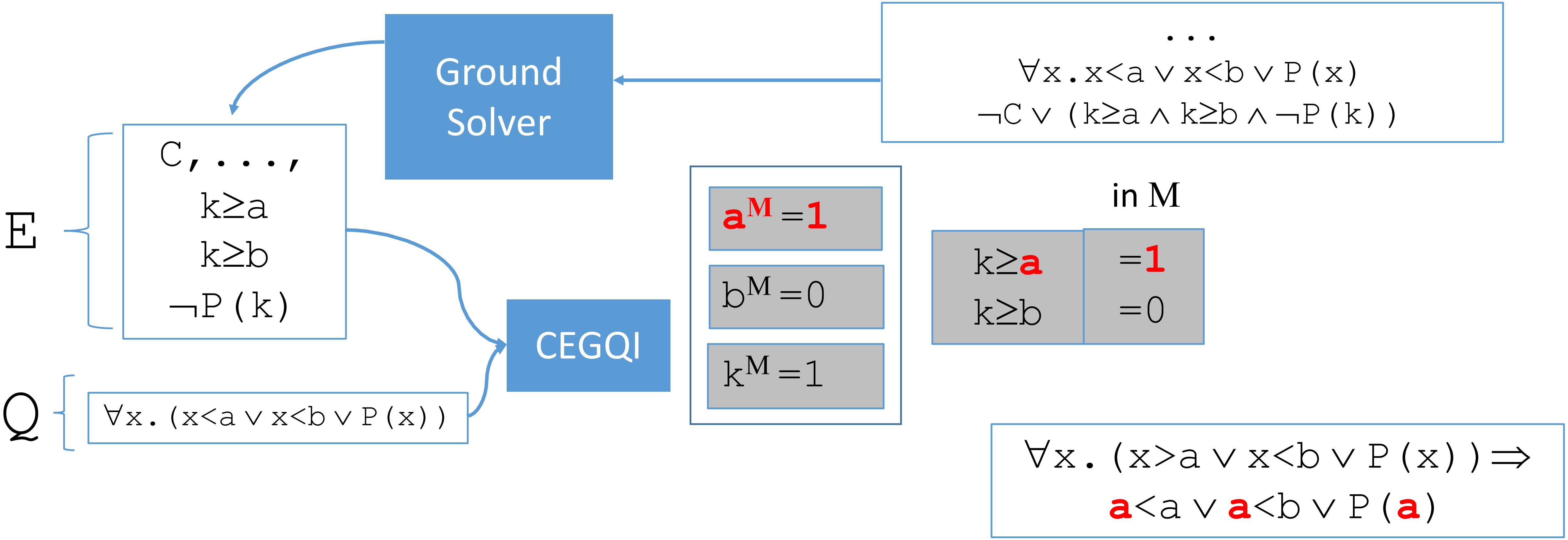
Q

$\forall x. (x<a \vee x<b \vee P(x))$

Counterexample-Guided Instantiation

CE-Guided

- Challenge: does not work in presence of uninterpreted functions!



Ground Solver

...

$\forall x. x<a \vee x<b \vee P(x)$

$\neg C \vee (k \geq a \wedge k \geq b \wedge \neg P(k))$

E

$C, \ldots,$

$k \geq a$

$k \geq b$

$\neg P(k)$

Q

$\forall x. (x<a \vee x<b \vee P(x))$

CEGQI

$a^M = 1$

$b^M = 0$

$k^M = 1$

in M

$k \geq a = 1$

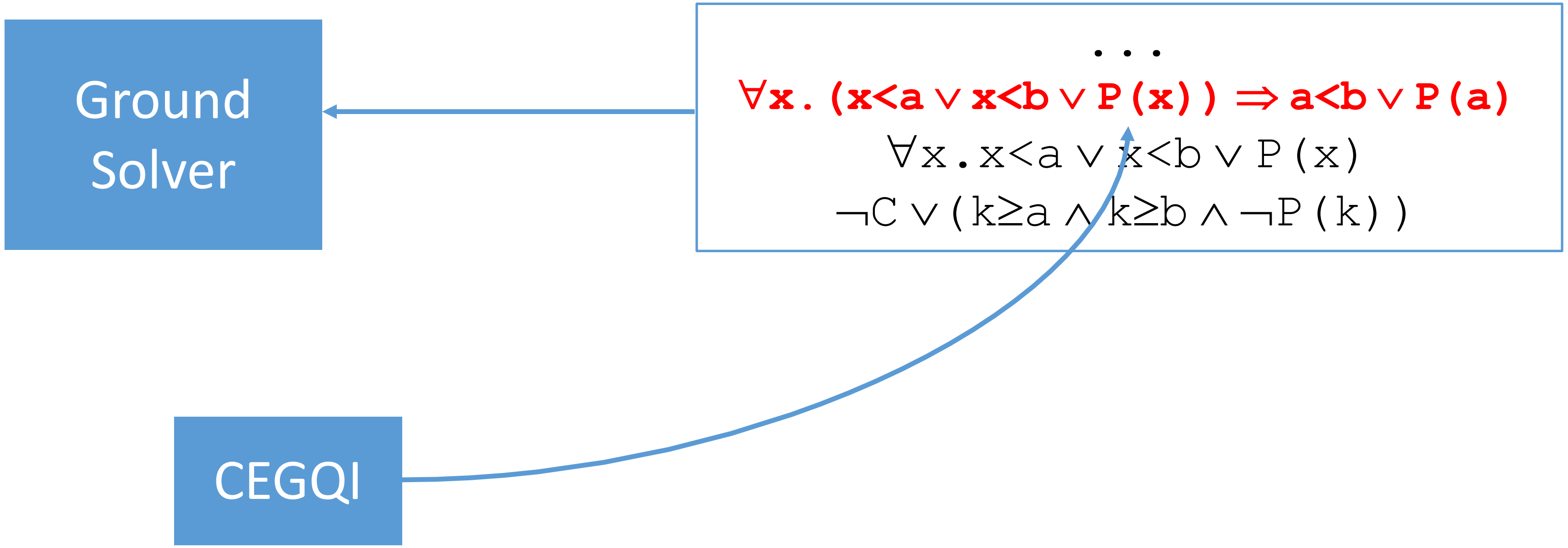
$k \geq b = 0$

$\forall x. (x>a \vee x<b \vee P(x)) \Rightarrow$

$a<a \vee a<b \vee P(a)$

Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!



. . .

$\forall x.(x<a \vee x<b \vee P(x)) \Rightarrow a<b \vee P(a)$

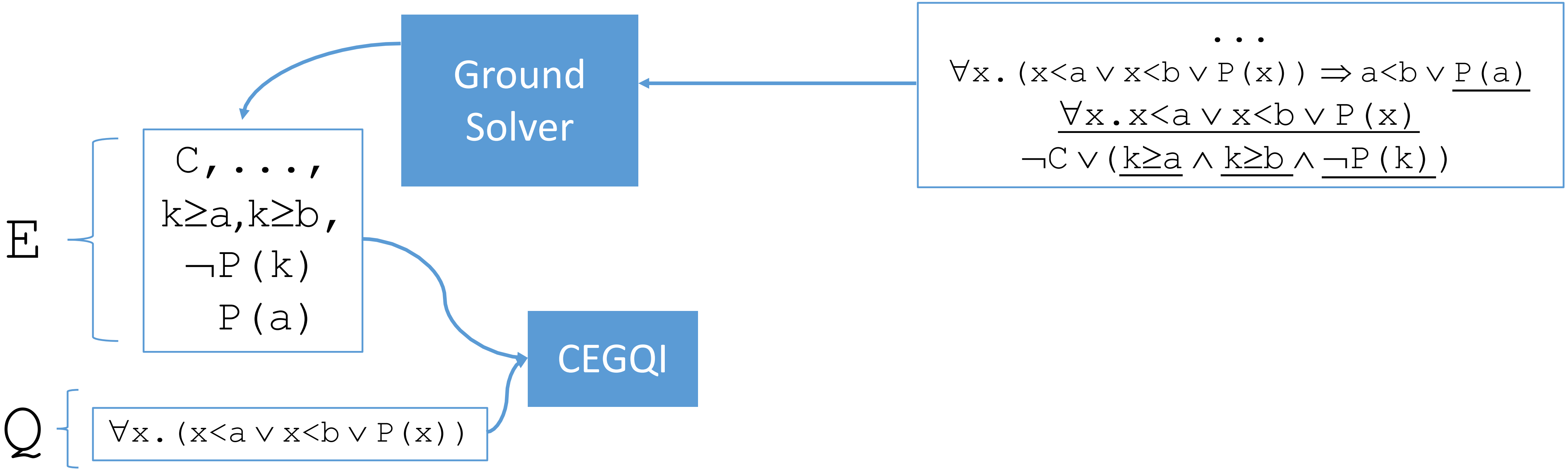
$\forall x. x<a \vee x<b \vee P(x)$

$\neg C \vee (k \geq a \wedge k \geq b \wedge \neg P(k))$

Counterexample-Guided Instantiation

CE-Guided

- Challenge: does not work in presence of uninterpreted functions!



Ground Solver

CEGQI

E: $C, \ldots,$ $k \geq a, k \geq b,$ $\neg P(k)$ $P(a)$

Q: $\forall x.(x<a \vee x<b \vee P(x))$

$\ldots$

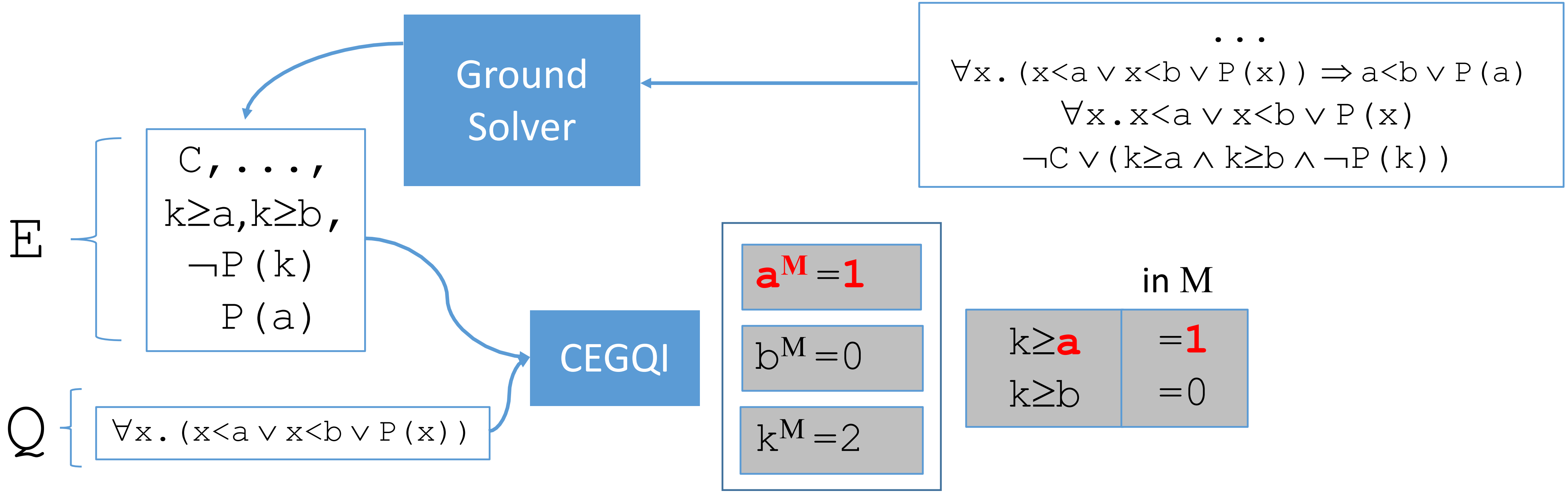
$\forall x.(x<a \vee x<b \vee P(x)) \Rightarrow a<b \vee \underline{P(a)}$

$\underline{\forall x. x<a \vee x<b \vee P(x)}$

$\neg C \vee (\underline{k \geq a} \wedge \underline{k \geq b} \wedge \underline{\neg P(k)})$

Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!



Ground Solver

...

$\forall x.(x<a \lor x<b \lor P(x)) \Rightarrow a<b \lor P(a)$

$\forall x. x<a \lor x<b \lor P(x)$

$\neg C \lor (k \geq a \land k \geq b \land \neg P(k))$

E

$C, \ldots,$
$k \geq a, k \geq b,$
$\neg P(k)$
$P(a)$

Q

$\forall x.(x<a \lor x<b \lor P(x))$

CEGQI

$a^M = 1$

$b^M = 0$

$k^M = 2$

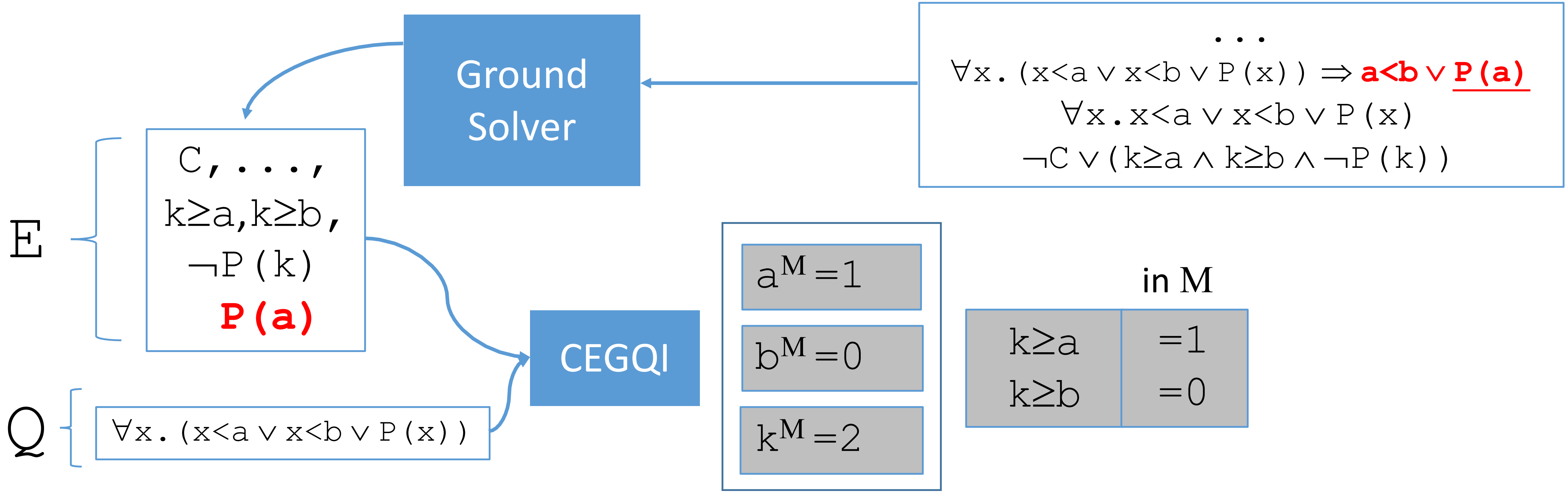
in M

$k \geq a$	$= 1$
$k \geq b$	$= 0$

$\Rightarrow$ a is still the maximal lower bound in M !

Counterexample-Guided Instantiation

- Challenge: does not work in presence of uninterpreted functions!



Ground Solver
...
$\forall x.(x<a \lor x<b \lor P(x)) \Rightarrow a<b \lor P(a)$
$\forall x.x<a \lor x<b \lor P(x)$
$\neg C \lor (k \geq a \land k \geq b \land \neg P(k))$
E
C, ...,
$k \geq a, k \geq b,$
$\neg P(k)$
$P(a)$
Q
$\forall x.(x<a \lor x<b \lor P(x))$
CEGQI
$a^M = 1$
$b^M = 0$
$k^M = 2$
in M
$k \geq a$
$k \geq b$
$= 1$
$= 0$

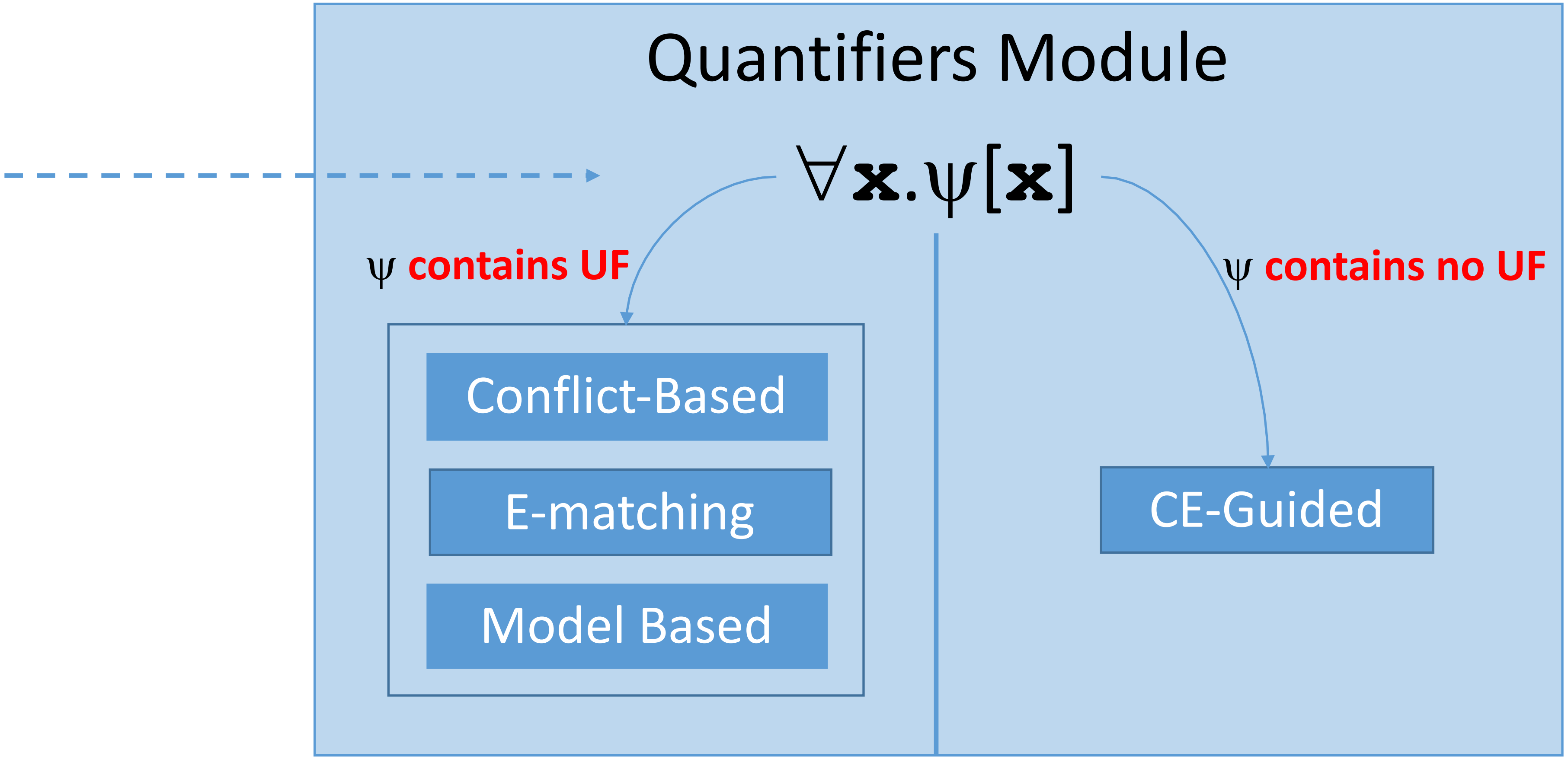
$\Rightarrow$ *Unlike the pure arithmetic case:*

- *Instance does not suffice to rule out* a *as maximal lower bound*

Summary

- SMT solvers handle quantifiers+theories via combination of:
  - DPLL(T)-based ground solver
  - Instantiation via:
    - **Conflict-based, E-matching, Model-Based Instantiation**
      - Effective in practice for ∀+UF, ∀+UFLIA, ∀+UFLRA, …
      - Can be decision procedure for limited fragments, e.g. Bernays-Shonfinkel
      - Conflict-Based, E-matching are useful for “unsat”
      - Model-Based is useful for “sat”
    - **Counterexample-guided Instantiation**
      - Decision procedure for ∀+LRA, ∀+LIA, ∀+BV, …

In practice: Distribute ∀ to proper strategy



Quantifiers Module

$\forall \mathbf{x}.\psi[\mathbf{x}]$

ψ contains UF

ψ contains no UF

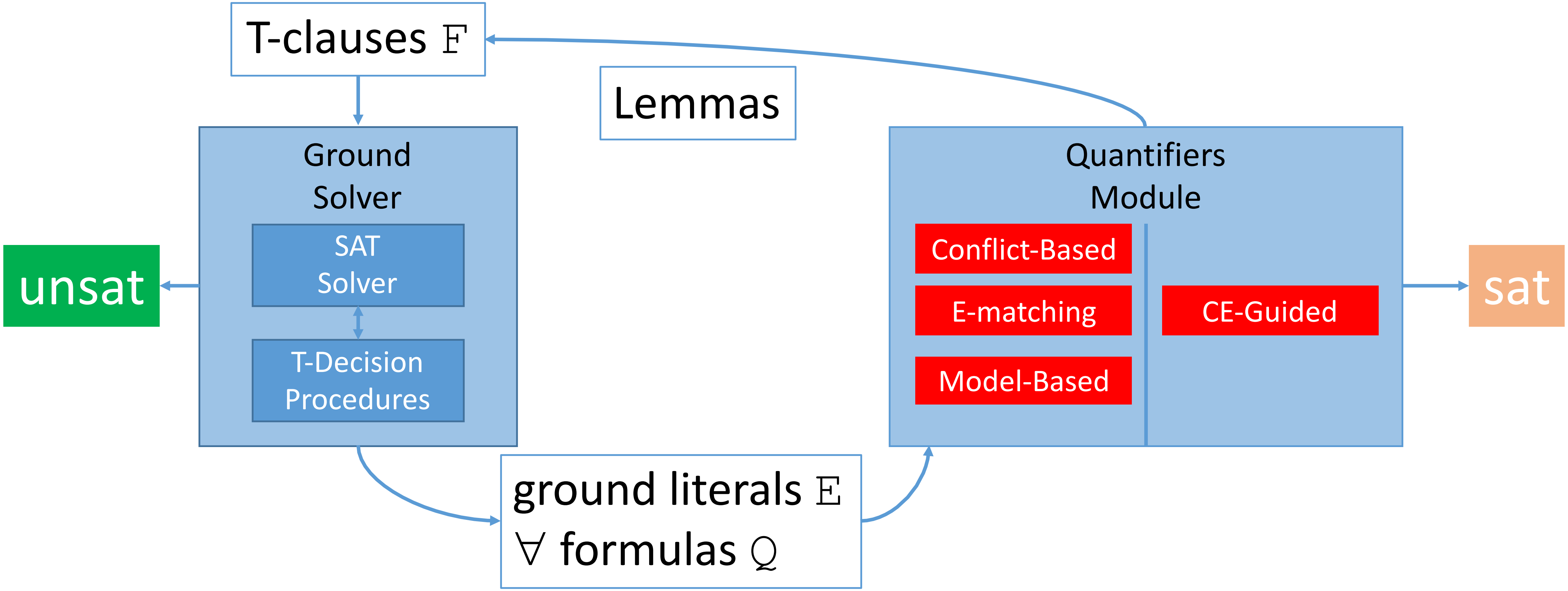
Conflict-Based

E-matching

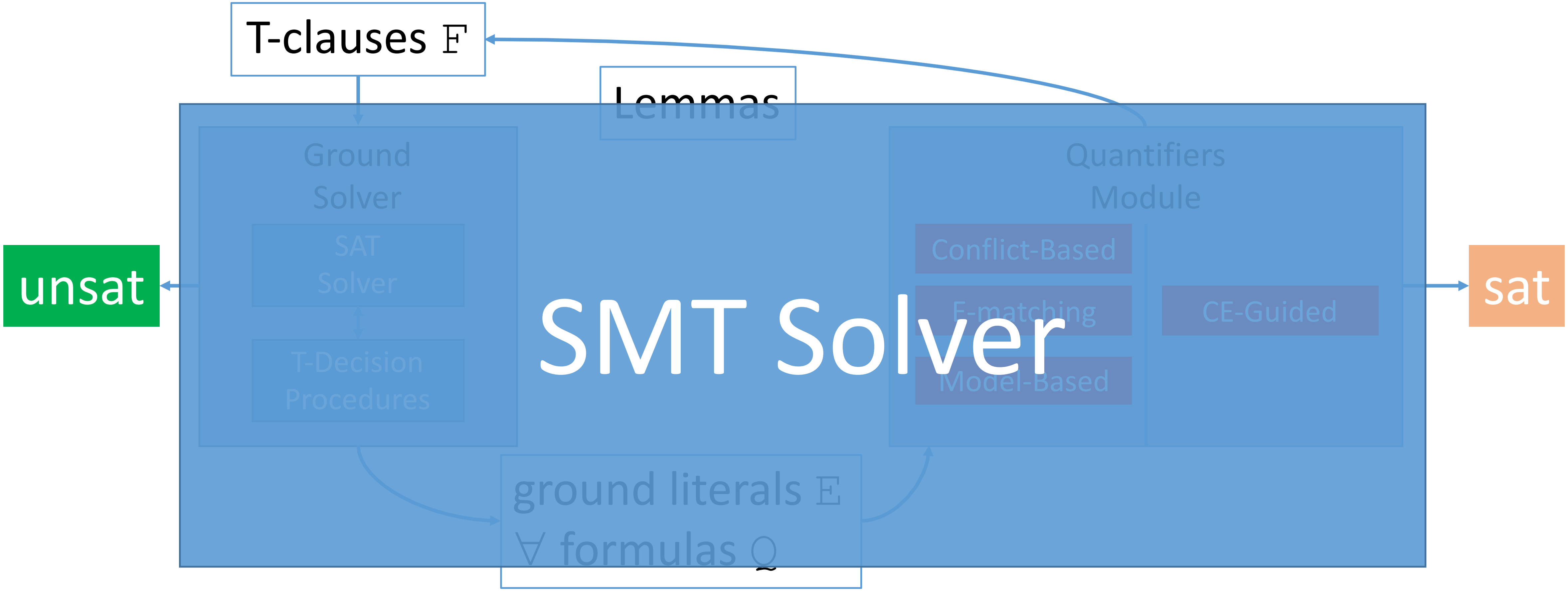
Model Based

CE-Guided

Summary: DPLL(T)+Instantiation



Summary: DPLL(T)+Instantiation


T-clauses $\mathbb{F}$
Lemmas
Ground
Solver
SAT
Solver
T-Decision
Procedures
Quantifiers
Module
Conflict-Based
E-matching
Model-Based
CE-Guided
SMT Solver
unsat
sat
ground literals $\mathbb{E}$
$\forall$ formulas $\mathbb{Q}$