Finding Conflicting Instances of Quantified Formulas in SMT

Andrew Reynolds
Cesare Tinelli
Leonardo De Moura
July 18, 2014

Outline of Talk

- SMT solvers:
  - Efficient methods for ground constraints
  - Heuristic methods for quantified formulas

  $\Rightarrow$ *Can we reduce dependency on heuristic methods?*
- New method for quantifiers in SMT
  - Finds conflicting instances of quantified formulas
- Experimental results
- Summary and Future Work

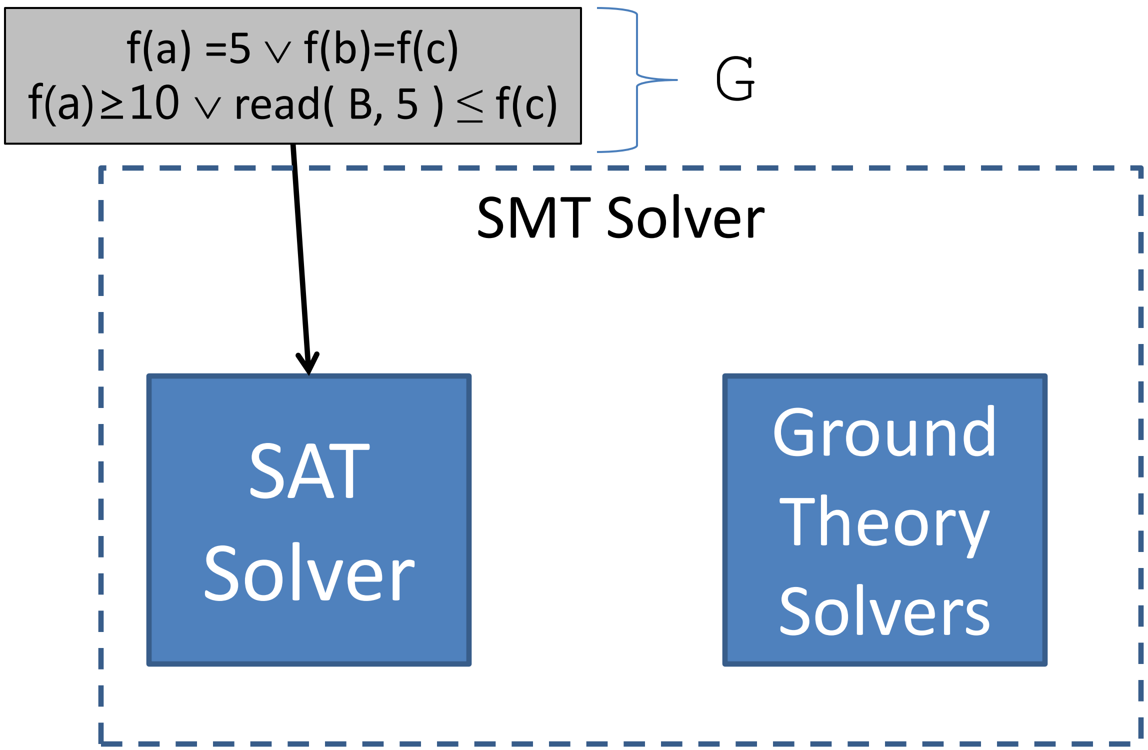
Satisfiability Modulo Theories (SMT)

- SMT solvers
  - Are efficient for problems over ground constraints $\mathbb{G}$
  - Determine the satisfiability of $\mathbb{G}$ using a combination of:
    - Off-the-shelf SAT solver
    - Efficient ground decision procedures, e.g.
      - Uninterpreted Functions
      - Linear arithmetic
      - Arrays
      - Datatypes
      - …

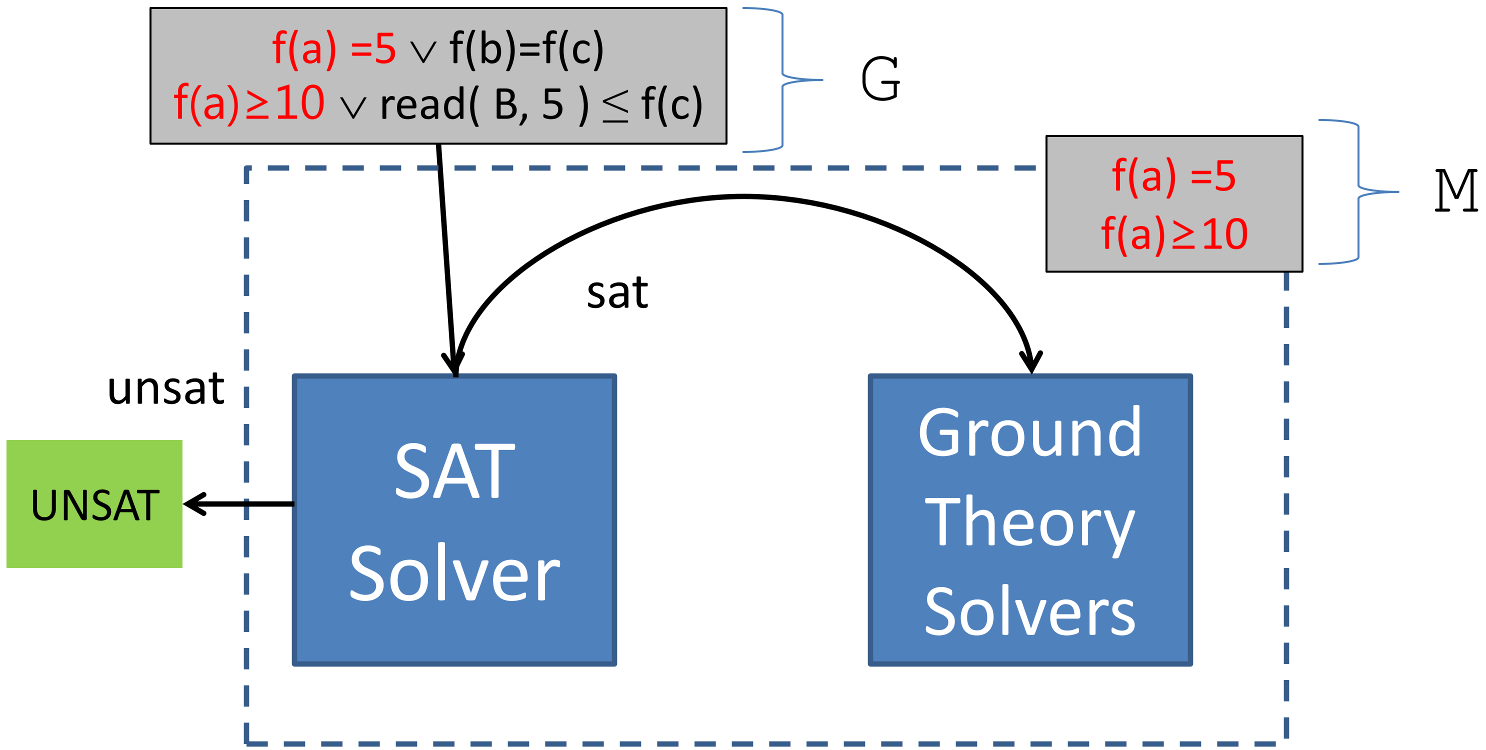
$$
\mathbb{G}\left\{
\begin{array}{c}
f(3) \neq f(c) \\
c=2 \lor c+1 \leq 0 \\
a+1 = read(A,b) \\
tail(l_1)=cons(a,l_2)
\end{array}
\right.
$$

- Used in many applications:
  - Software/hardware verification
  - Scheduling and Planning
  - Automated Theorem Proving

DPLL(T)-Based SMT Solver

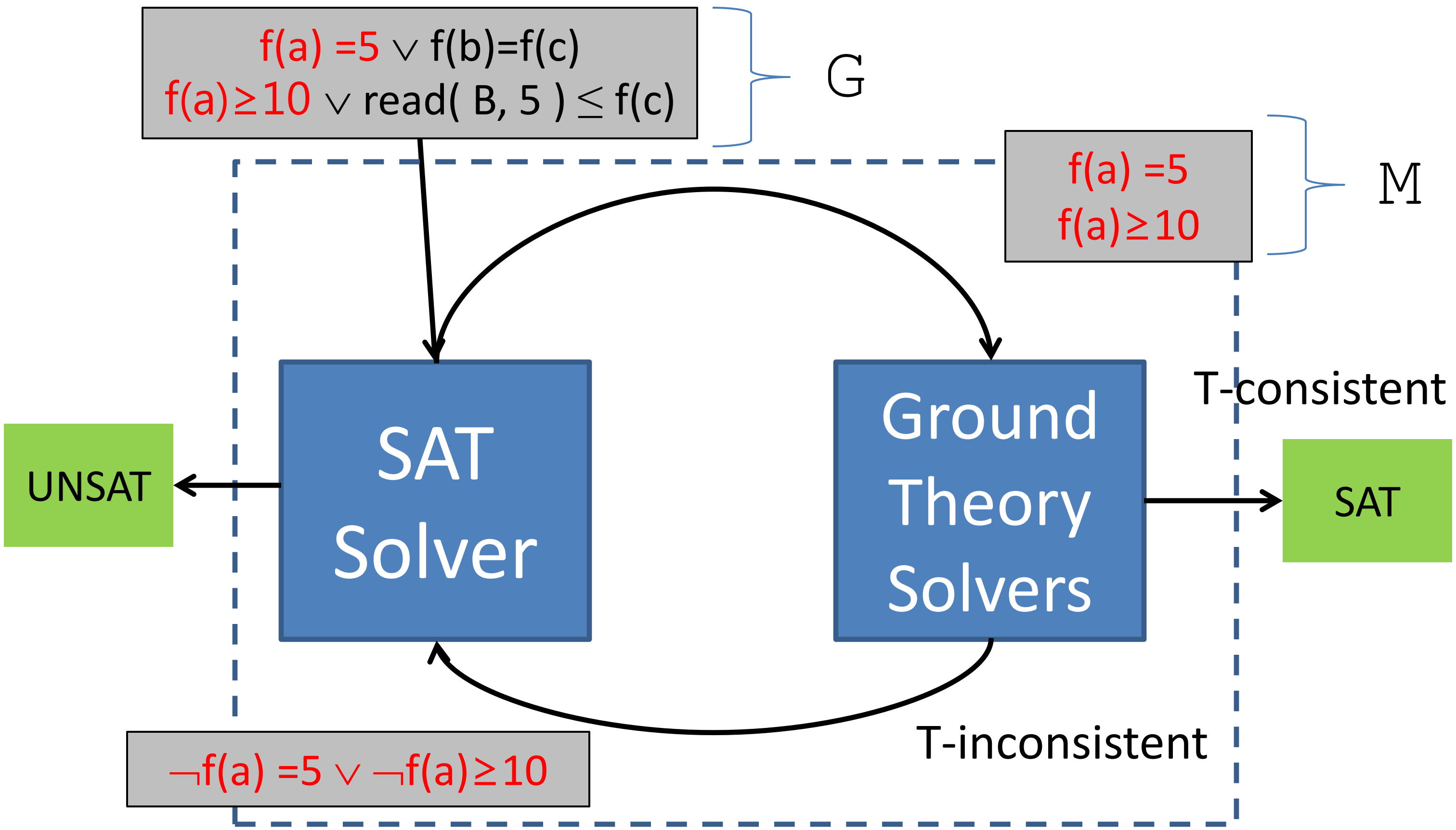


DPLL(T)-Based SMT Solver



- SAT solver either:
  - Determines G is unsatisfiable at propositional level
  - Returns a satisfying assignment M, e.g. a "context"

DPLL(T)-Based SMT Solver



- Ground theory solvers either:
  - Determines $\mathbb{M}$ is consistent according to theory
  - Add clause to $\mathbb{G}$ that explains why $\mathbb{M}$ is inconsistent

SMT + Quantified Formulas

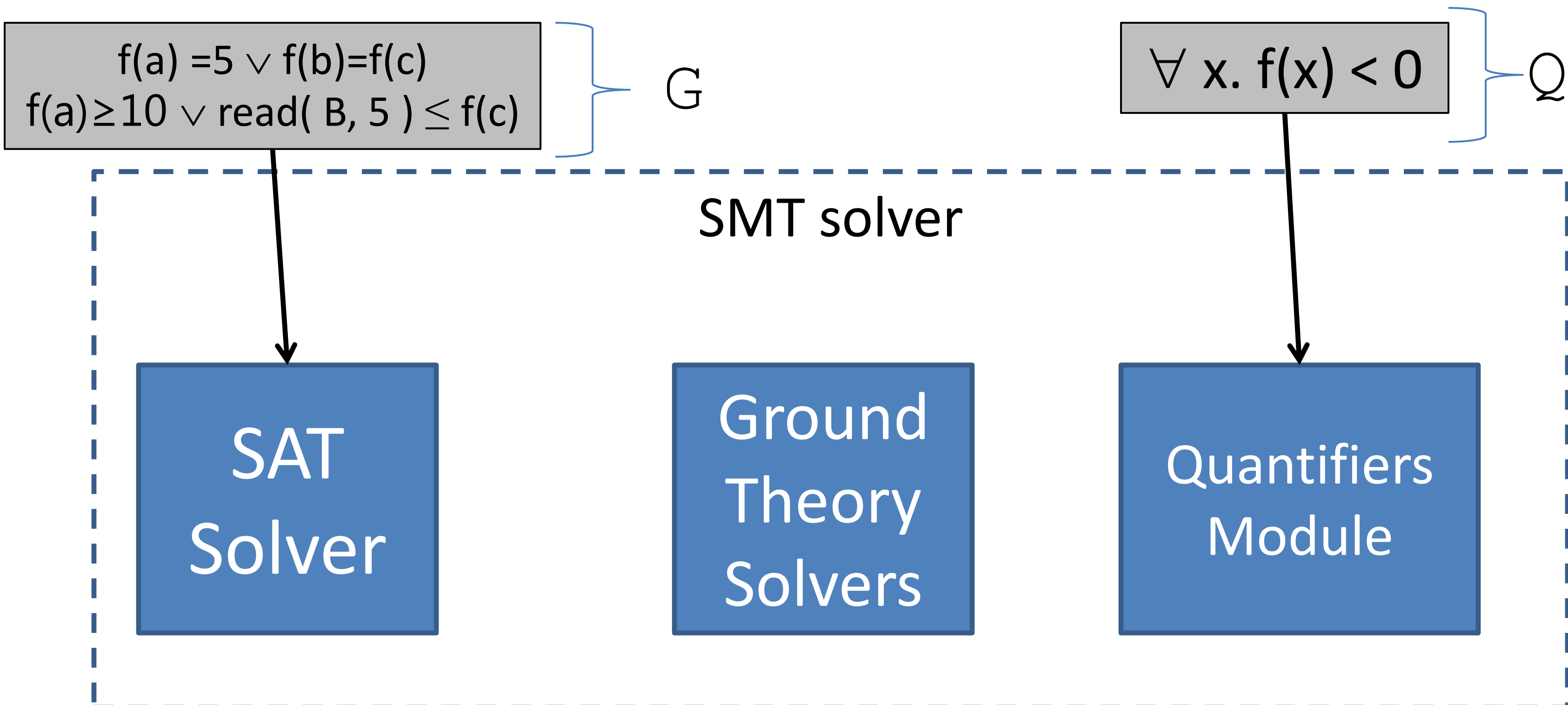
- SMT solvers have limited support for:
  - First-order universally quantified formulas Q

$$\left.\begin{array}{c} f(a) = 5 \vee f(b)=f(c) \\ f(a)\geq 10 \vee read(\ B, 5\ ) \leq f(c) \end{array}\right\} G$$

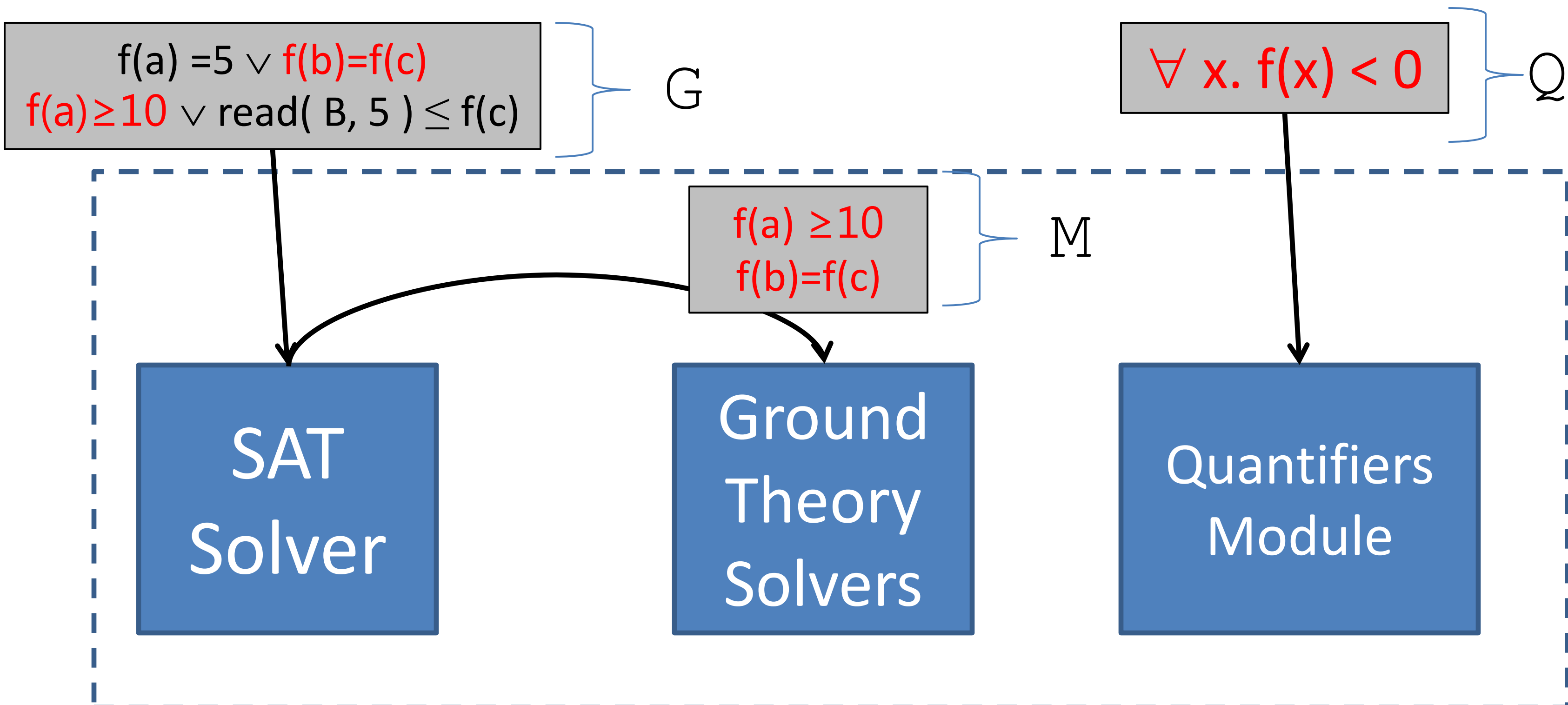
$$\left.\begin{array}{c} \forall\ x.\ f(x) < 0 \end{array}\right\} Q$$

- Used in an increasing number of applications, for:
  - Defining axioms for symbols not supported natively
  - Encoding frame axioms, transition systems, …
  - Universally quantified conjectures
- When universally quantified formulas Q are present, decision problem is generally undecidable
  - General approaches for $G \cup Q$ in SMT are heuristic

SMT Solver + Quantified Formulas

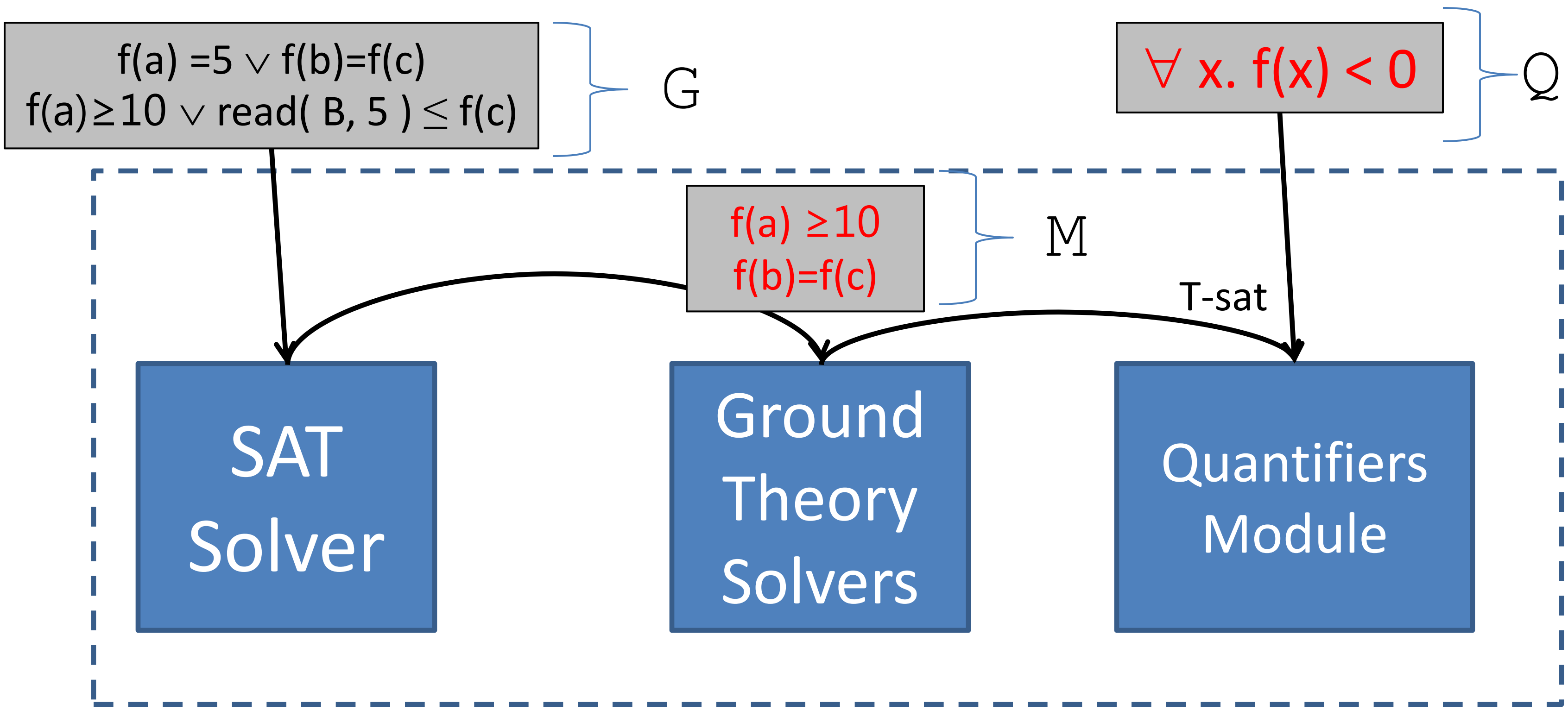


SMT Solver + Quantified Formulas



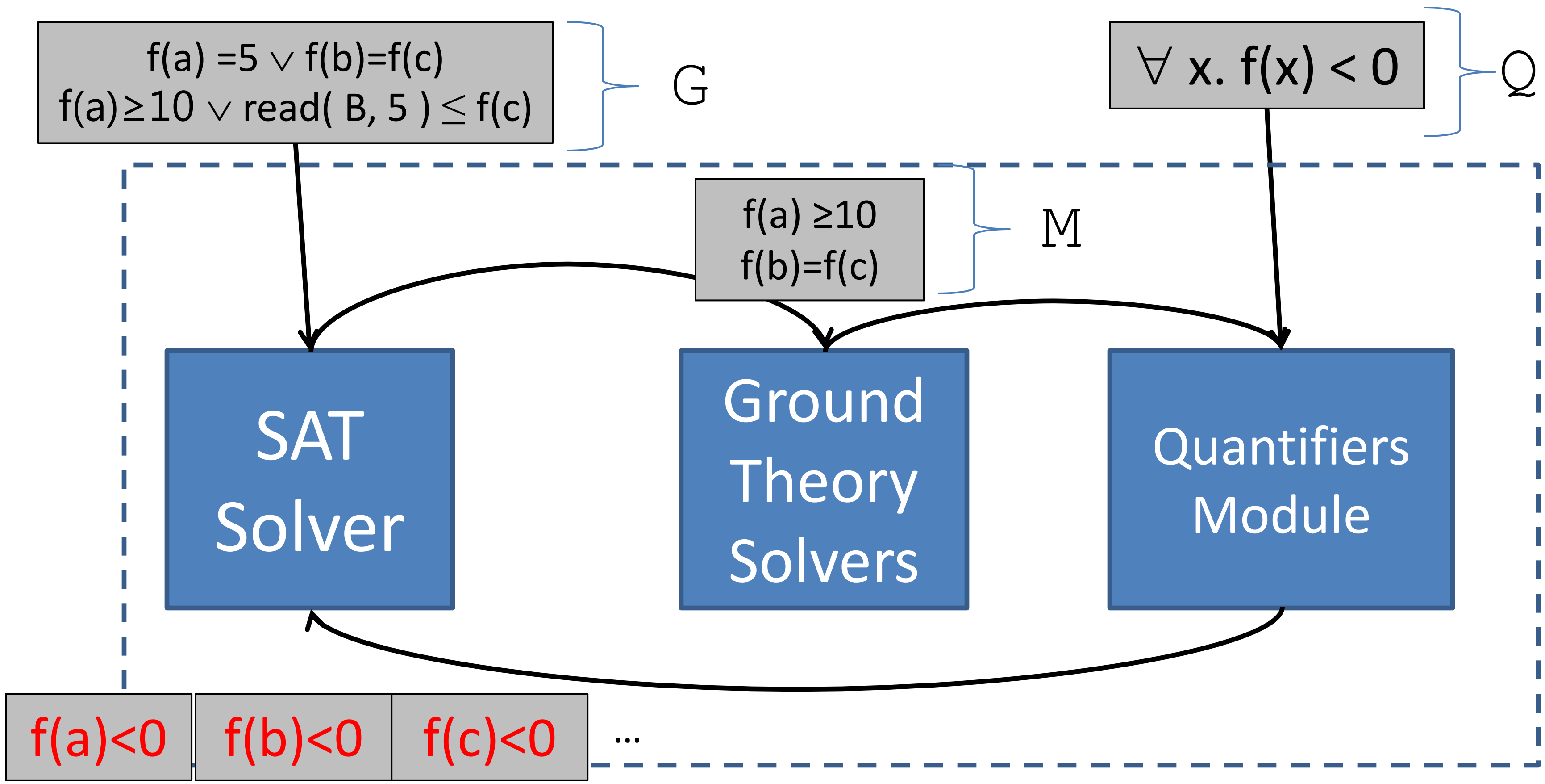
- Find satisfying assignment $\mathbb{M}$

SMT Solver + Quantified Formulas



- If M is T-consistent,
  - Then we must answer: *"is $M \cup Q$ consistent?"*
    - Problem is generally undecidable

Quantifier Instantiation



- Instantiation-based approaches:
  - Add instances of quantified formulas, based on some strategy
    - E.g. based on patterns (known as "E-matching")

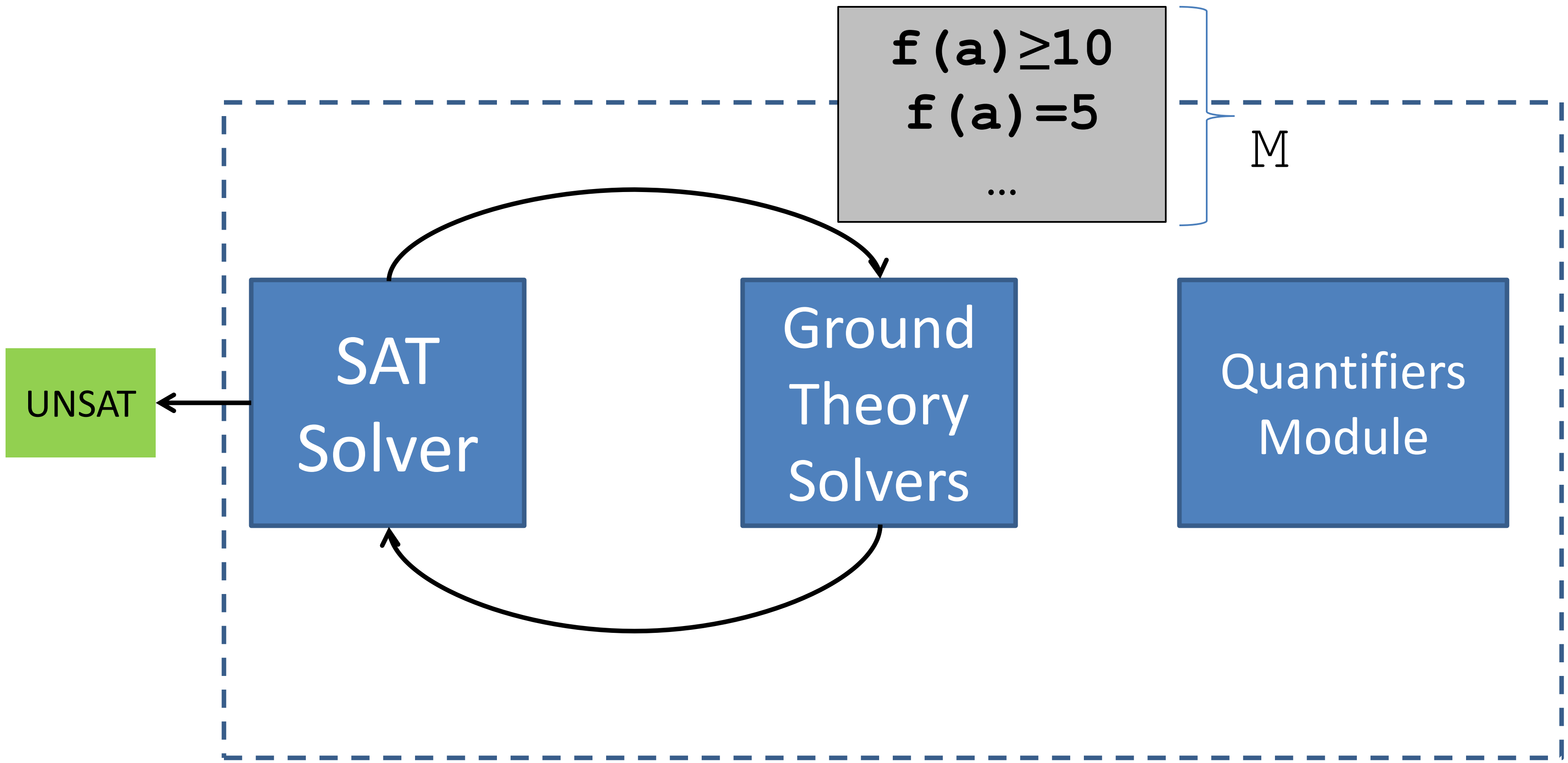
Instantiation-Based Approaches

- Complete approaches:
  - E.g. Complete instantiation, local theory extensions, finite model finding, Inst-Gen, user triggers
    - Idea: identify a finite subset of instances of $\mathcal{Q}$ to consider
    - Cons: only work for limited fragments
- General approaches: *Focus of this talk*
  - Heuristic E-matching
    - Idea: choose instances of $\mathcal{Q}$ based on pattern matching
    - Cons: only for UNSAT, highly heuristic, often inefficient

Motivation

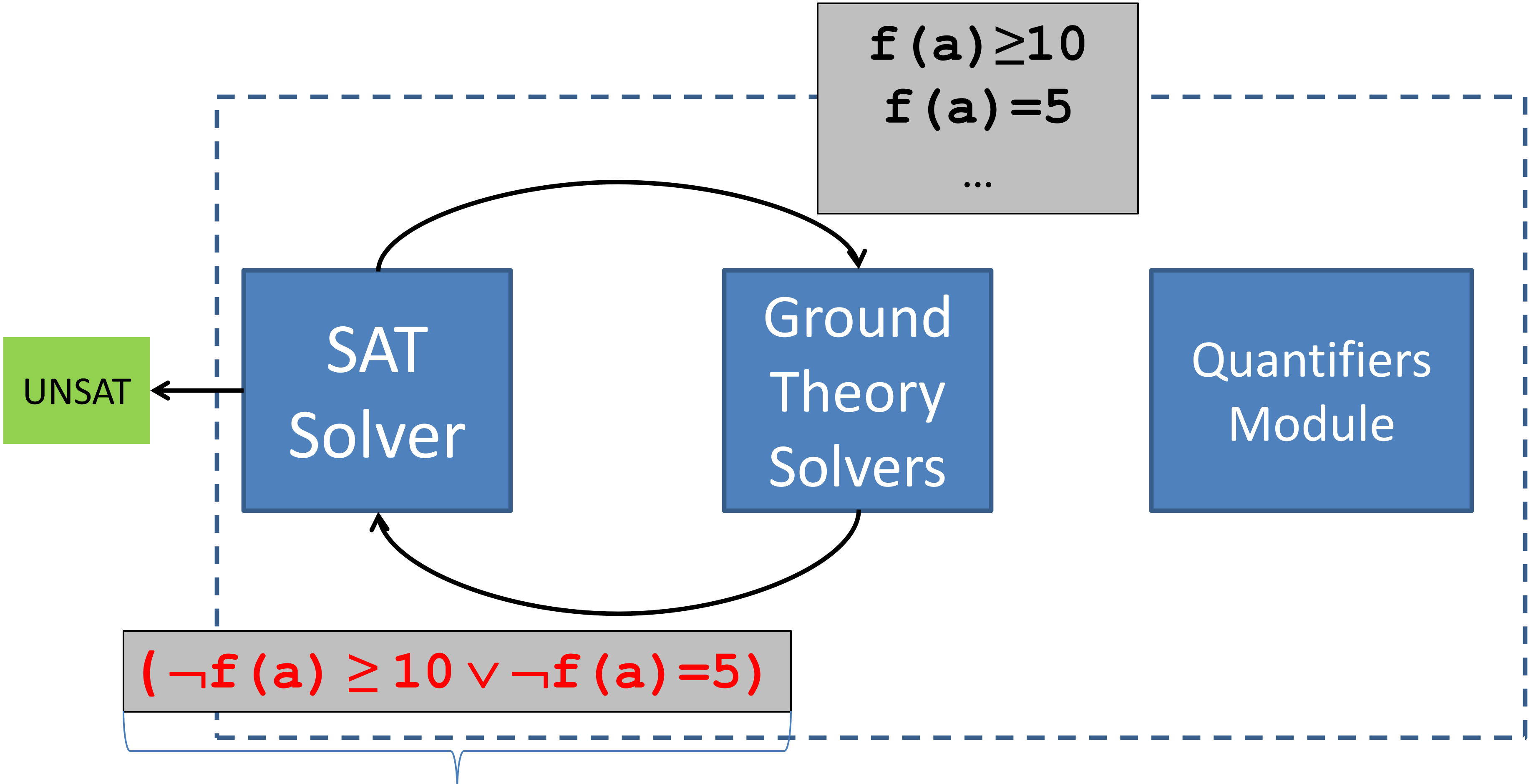
- Current SMT solvers:
  - Are highly efficient for ground constraints
    - Recognizing theory conflicts, T-propagations, …
  - Resort to heuristic instantiation for quantified formulas
    - Expensive, due to overloading the solver with instances
- In this talk: new method for handling quantified formulas
  - Goals:
    - Reduce dependency on heuristic methods
    - Applicable to arbitrary quantified formulas
  - Not goals:
    - Completeness (thus, focus only on UNSAT)

Ground Theories : Conflicts



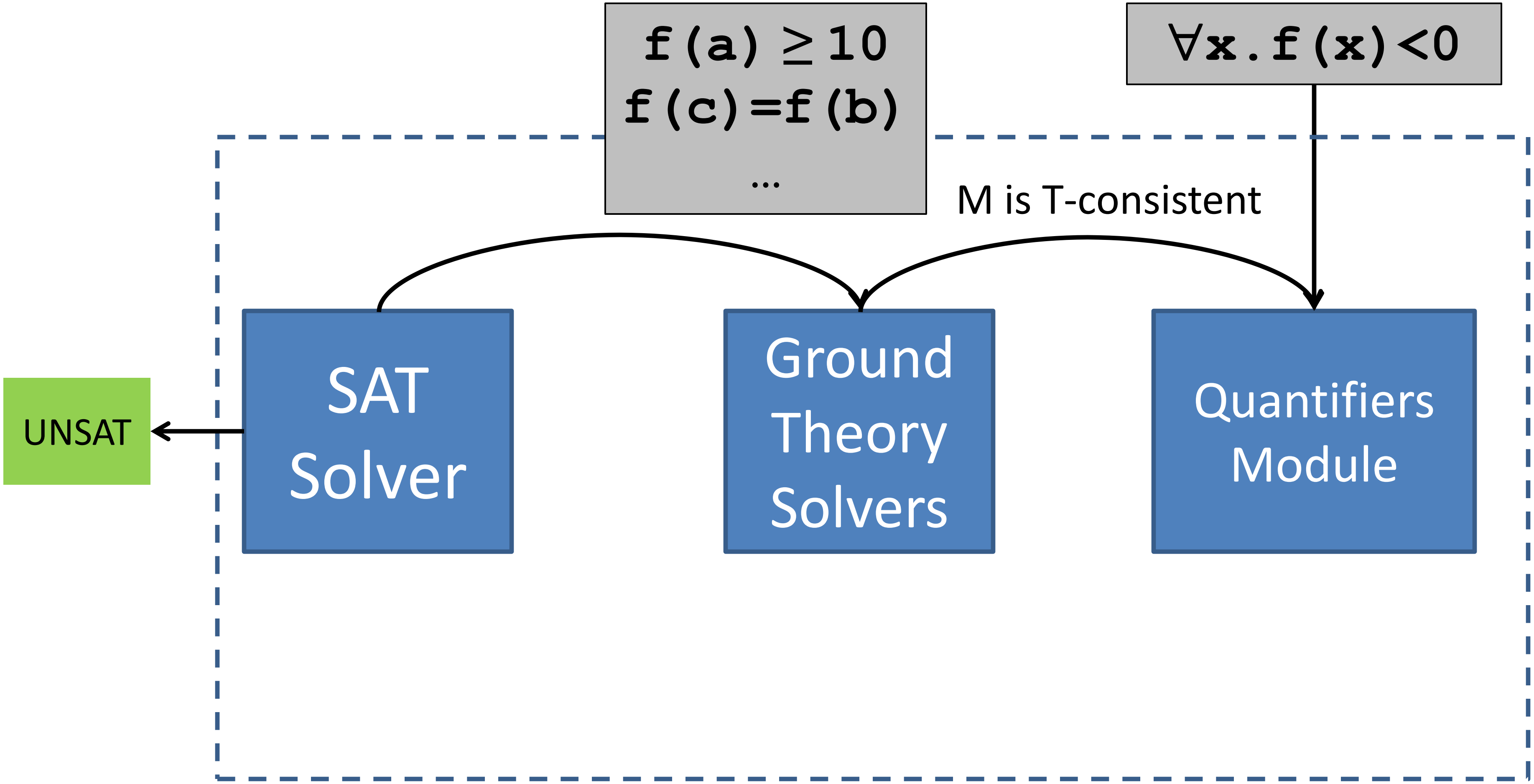
- If M is inconsistent according to ground theory,

Ground Theories : Conflicts



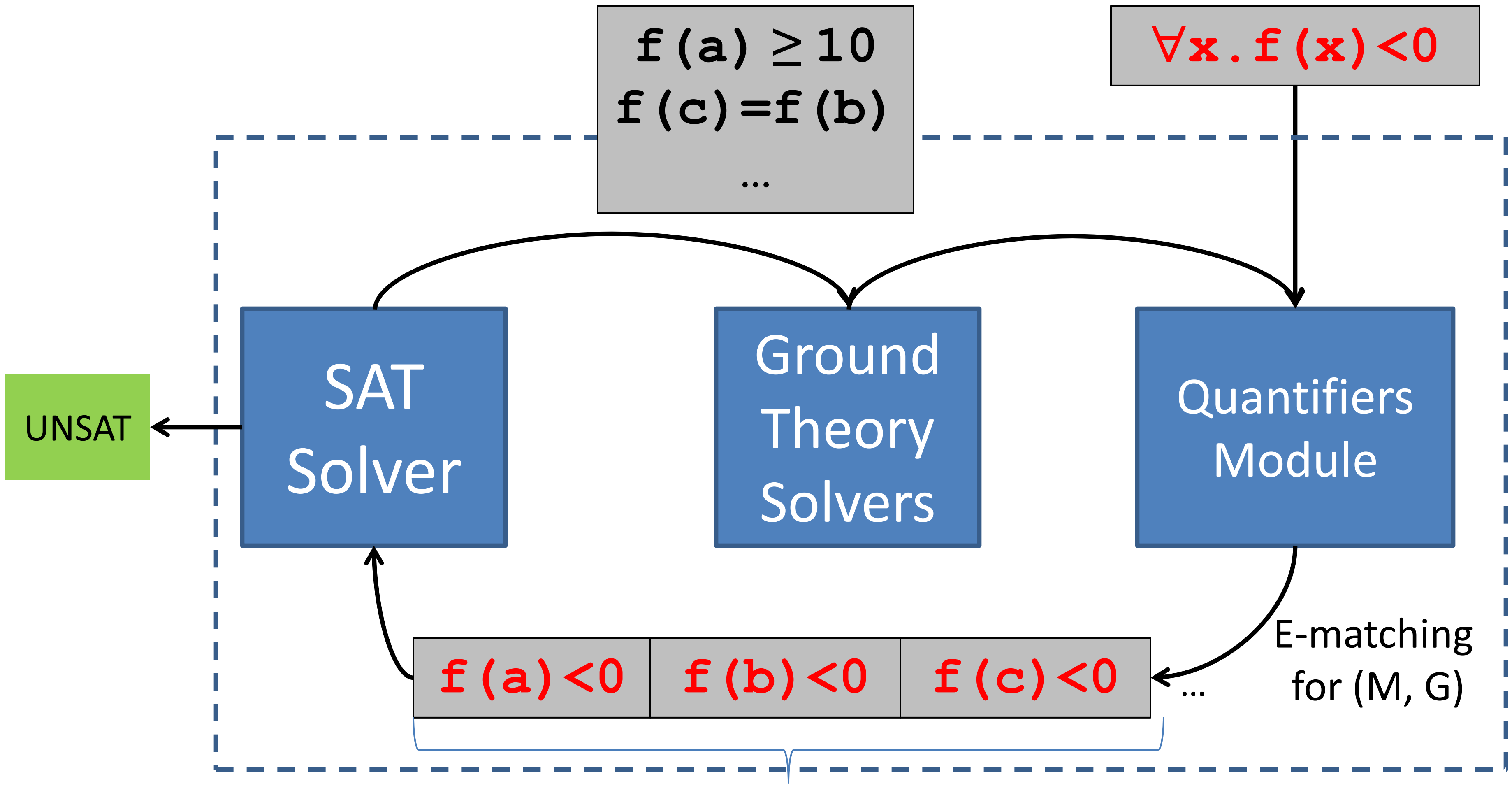
- Ground theory solver reports a single conflict clause
  - Typically, can be determined efficiently

Quantifiers : Heuristic Instantiation?



- The decision problem for $M \cup Q$ is undecidable,

Quantifiers : Heuristic Instantiation?



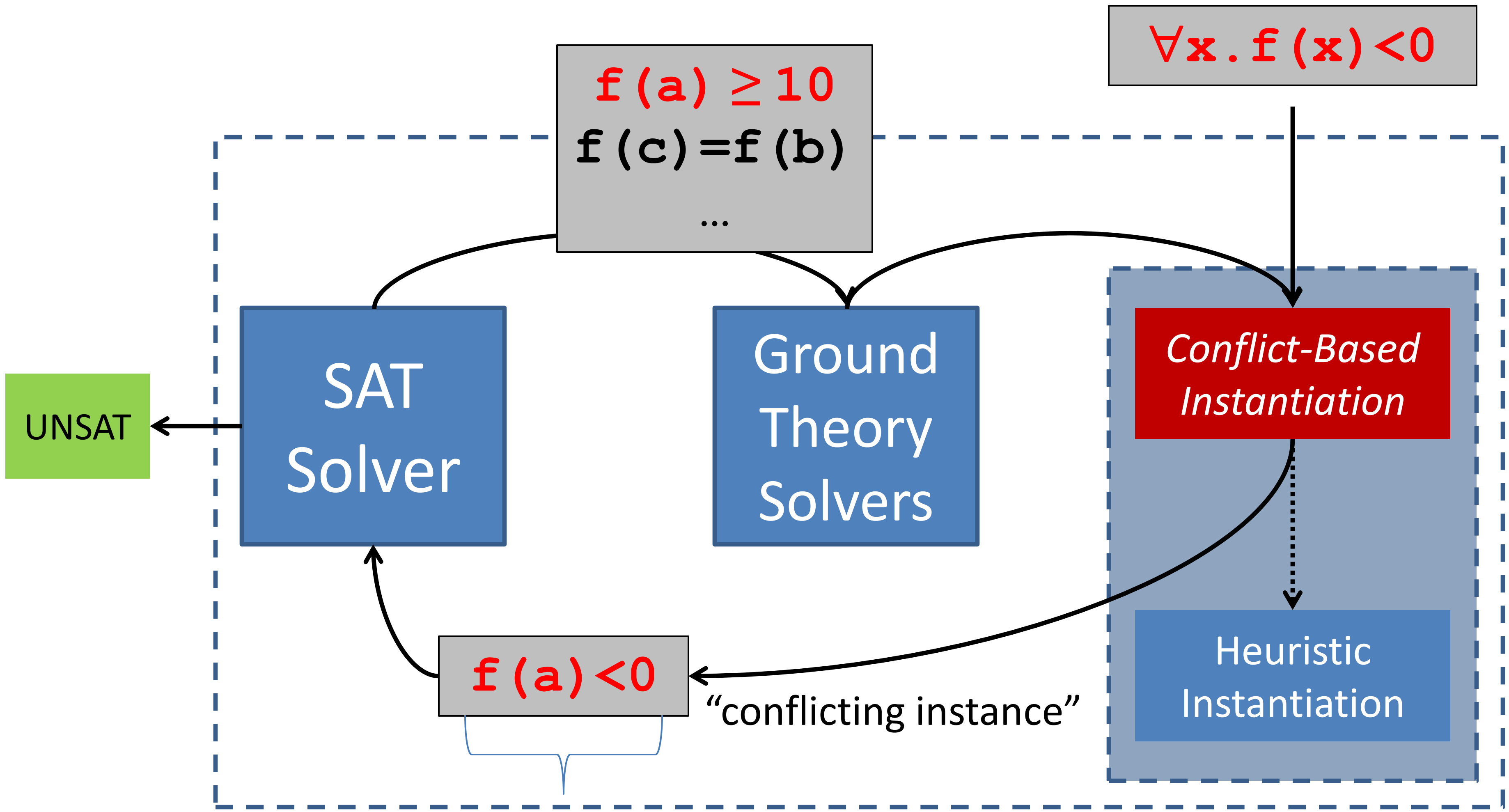
- Add a potentially large set of instances, heuristically
  - This can overload the ground solver

Conflicting Instances

$\Rightarrow$*Can we make the quantifiers module behave more like a theory solver?*

- Idea: find cases when $\mathrm{M} \cup \mathrm{Q}$ is inconsistent:
  - Quantified formula $\mathrm{Q}_1 \in \mathrm{Q}$
  - Grounding substitution σ
    - Such that $\mathrm{M} \models_T \neg \mathrm{Q}_1\sigma$
- $\mathrm{Q}_1\sigma$ is a *conflicting instance*

Conflict-Based Instantiation



- First, determine if a conflicting instance exists
  - If not, resort to heuristic instantiation

Limit of Approach

- *Caveat*: No complete method will determine whether a conflicting instance exists for $(\mathbb{M},\mathbb{Q})$
- Thus, our approach:
  1. Uses an incomplete procedure to determine a conflicting instance for $(\mathbb{M}, \mathbb{Q})$
  2. If not, resort to E-matching for $(\mathbb{M}, \mathbb{Q})$

  $\Rightarrow$ *In practice, Step 1 succeeds for a majority of* $(\mathbb{M}, \mathbb{Q})$

E-matching vs Conflicting Instances

$$\underbrace{\overbrace{g(b)}^{\text{Ground term}} \neq f(a), \quad b = h(a)}_{\mathbb{M}}$$

$$\underbrace{\forall x.\, f(x) = \underbrace{g(h(x))}_{\text{Trigger term}}}_{\mathbb{Q}}$$

- In above example,
  - $g(h(x))$ is a trigger term for $\mathbb{Q}$
    - $\mathbb{M} \models_T g(b)=g(h(x))\sigma$, for $\sigma = \{x \rightarrow a\}$

$\Rightarrow$ *E-matching for (M,Q) returns σ*

E-matching vs Conflicting Instances

$$\underbrace{\boxed{\begin{array}{c} g(b) \neq f(a) \\ b = h(a) \end{array}}}_{\mathrm{M}} \qquad \underbrace{\boxed{\forall x.\ f(x) = g(h(x))}}_{\mathrm{Q}}$$

- In this example, for $\sigma = \{ x \rightarrow a \}$:
  1. Ground terms match each sub-term from Q
     - $\mathrm{M} \models_{T} g(b) = g(h(x))\sigma$
     - $\mathrm{M} \models_{T} f(a) = f(x)\sigma$
  2. …and the body of Q is falsified:
     - $\mathrm{M} \models_{T} f(x) \neq g(h(x))\sigma$

$\Rightarrow$ *σ is a conflicting substitution*

E-matching vs Conflicting Instances

$$\boxed{\begin{array}{c} g(b) \neq f(a) \\ b = h(a) \end{array}} \Big\} \; \mathbb{M} \qquad \boxed{\forall x.\ f(x) = g(h(x))} \Big\} \; \mathbb{Q}$$

- In this example, for $\sigma = \{\ x \rightarrow a\ \}$:
  1. Ground terms match each sub-term from $\mathbb{Q}$
     - $\mathbb{M} \models_T g(b)=g(h(x))\sigma$
     - $\mathbb{M} \models_T f(a)=f(x)\sigma$
  2. ...and the body of $\mathbb{Q}$ is falsified:
     - $\mathbb{M} \underbrace{\models_T}$ $f(x) \neq g(h(x))\sigma$

For now, limit T to EUF

$\Rightarrow$ σ is a conflicting substitution

- ***Finding σ requires:*** modified version of E-matching

E-matching vs Conflicting Instances

$$\underbrace{\boxed{\begin{array}{c} g(b) \neq f(a) \\ b = h(a) \end{array}}}_{\mathbb{M}} \qquad \underbrace{\boxed{\forall x.\ f(x) = g(h(x))}}_{\mathbb{Q}}$$

- Consider *flat form* of Q:

$$\forall x\ y_1\ y_2\ y_3.$$
$$\underbrace{y_1 = f(x) \wedge y_2 = g(y_3) \wedge y_3 = h(x)}_{\text{Matching constraints } \mu} \Rightarrow \underbrace{y_1 = y_2}_{\text{Flattened body } \Psi}$$

- Conflicting substitution σ for $(\mathbb{M}, \mathbb{Q})$ is such that:
  - $\mathbb{M}$ entails $\mu\sigma$
  - $\mathbb{M}$ entails $\neg\Psi\sigma$

Equality-Inducing Instances

$$\left.\begin{array}{c} g(b)\neq c \\ d=f(a) \\ b=h(a) \end{array}\right\} \mathtt{M} \qquad \left.\boxed{\forall x.\ f(x) = g(h(x))}\right\} \mathtt{Q}$$

- Modified example, for $\sigma = \{ x \rightarrow a \}$:
  1. Ground terms match each sub-term from $\mathtt{Q}$
     - $\mathtt{M} \models_T g(b)=g(h(x))\sigma$
     - $\mathtt{M} \models_T f(a)=f(x)\sigma$
  2. …but the body of $\mathtt{Q}$ is *not* falsified:
     - $\mathtt{M} \not\models_T f(x)\neq g(h(x))\sigma$

Equality-Inducing Instances

$$\underbrace{\boxed{\begin{array}{c} g(b)\neq c \\ d=f(a) \\ b=h(a) \end{array}}}_{\mathbb{M}} \qquad \underbrace{\boxed{\forall x.\ f(x) = g(h(x))}}_{\mathbb{Q}}$$

- *Still*, it may be useful to add the instance $\mathbb{Q}\{ x \rightarrow a \}$
  - It entails equality g(b) = f(a) between known terms in $\mathbb{M}$

    $\Rightarrow \{ x \rightarrow a \}$ is an equality-inducing substitution
  - Mimics T-propagation done by theory solvers
- Such substitutions produced by relaxing criteria #2
  - $\mathbb{M}$ need not entail *disequalities* from $\neg\mathbb{Q}\{ x \rightarrow a \}$

Instantiation Strategy

InstantiationRound($\mathbb{Q}$, $\mathbb{M}$)
(1) Return a (single) conflicting instance for ($\mathbb{Q}$, $\mathbb{M}$)
(2) Return a set of equality-inducing instances for ($\mathbb{Q}$, $\mathbb{M}$)
(3) Return instances based on E-matching for ($\mathbb{Q}$, $\mathbb{M}$)

- Three configurations:
  - **cvc4** : step (3)
  - **cvc4+c** : steps (1), (3)
  - **cvc4+ci** : steps (1),(2),(3)

Experimental Results

- Implemented techniques in SMT solver CVC4
- UNSAT benchmarks from:
  - TPTP
  - Isabelle
  - SMT Lib
- Solvers:
  - **cvc3, z3**
  - 3 configurations: **cvc4, cvc4+c, cvc4+ci**

UNSAT Benchmarks Solved

	cvc3	z3	cvc4	cvc4+c	cvc4+ci
TPTP	5234	6268	6100	6413	6616
Isabelle	3827	3506	3858	3983	4082
SMTLIB	3407	3983	3680	3721	3747
Total	12468	13757	13638	14117	14445

- Configuration cvc4+ci solves the most (14,445)
  - Against cvc4 : 1,049 vs 235 (+807)
  - Against z3: 1,998 vs 1,310 (+688)
  - 359 that no implementation of E-matching (cvc3, z3, cvc4) can solve

Instantiations for Solved Benchmarks

	TPTP		Isabelle		SMT lib	
	Solved	Inst	Solved	Inst	Solved	Inst
cvc3	5245	627.0M	3827	186.9M	3407	42.3M
z3	6269	613.5M	3506	67.0M	3983	6.4M
cvc4	6100	879.0M	3858	119.M	3680	60.7M
cvc4+c	6413	190.8M	3983	54.0M	3721	41.1M
cvc4+ci	6616	150.9M	4082	28.2M	3747	32.5M

- cvc4+ci
  - Solves the most benchmarks for TPTP and Isabelle
  - Requires almost an order of magnitude fewer instantiations
- Improvements less noticeable on SMT LIB
  - Due to encodings that make heavy use of theory symbols
    - Method for finding conflicting instances is more incomplete

Instances Produced

InstantiationRound($\mathbb{Q}$, $\mathbb{M}$)
(1) conflicting instance for ($\mathbb{Q}$, $\mathbb{M}$)
(2) equality-inducing instances for ($\mathbb{Q}$, $\mathbb{M}$)
(3) E-matching for ($\mathbb{Q}$, $\mathbb{M}$)

			E-matching		Conflicting		C-Inducing	
		IR	IR	#	IR	#	IR	#
smtlib	**cvc4**	14032	100.0%	60.7M				
	cvc4+c	51696	24.3%	41.0M	75.7%	39.1K		
	cvc4+cp	58003	20.0%	32.3M	71.6%	41.5K	8.4%	51.5K
TPTP	**cvc4**	71634	100.0%	879.0M				
	cvc4+c	201990	21.7%	190.1M	78.3%	158.2K		
	cvc4+cp	208970	20.3%	150.4M	76.4%	160.0K	3.3%	41.6K
Isabelle	**cvc4**	6969	100.0%	119.0M				
	cvc4+c	18160	28.9%	54.0M	71.1%	12.9K		
	cvc4+cp	21756	22.4%	28.2M	64.0%	13.9K	13.6%	130.1K

- Conflicting instances found on ~75% of IR
- cvc4+ci :
  - Requires 3.1x more instantiation rounds w.r.t. cvc4
  - Calls E-matching 1.5x fewer times overall
    - As a result, adds 5x fewer instantiations

Details on Solved Problems

- For the 30,081 benchmarks we considered:
  - cvc4+ci solves more (14,445) than any other
  - 359 are solved *uniquely* by cvc4+c or cvc4+ci
    - Techniques increase precision of SMT solver
  - cvc4+ci does not rely on E-matching for 21% of benchmarks
    - 94 of these not solved by any E-matching implementation
    - Techniques reduce dependency on heuristic instantiation

Comparison with ATP

- Modern ATP use strategy scheduling
  - Using scheduling strategy from CASC 24:
    - E solves 9,751 unsatisfiable TPTP benchmarks
    - iProver solves 6,508
  - Using scheduling with techniques from paper:
    - CVC4 solves 7,227

      $\Rightarrow$Fairly competitive with modern ATP systems
- For more comparison, see CVC4 in CASC J7 FOF

Summary and Future Work

- Conflict-based method for quantifiers in SMT
  - Supplements existing techniques
  - Improves performance, both in:
    - Number of instantiations required for UNSAT
    - Number of UNSAT benchmarks solved
- Future work:
  - More incremental instantiation strategies
  - Specialize techniques to other theories
    - Handle quantified formulas containing (e.g.) linear arithmetic
  - Completeness
    - Identify criteria for saturation

Thank You

- Solver is publicly available:
  `http://cvc4.cs.nyu.edu/`
- Techniques enabled by option:
  “`cvc4 --quant-cf` …”