Quantifier

Zhaoguo Wang

Adapted From:

NYU G22.2390-001, Lecture 8, 9
ETH Zurich program verification
(https://www.pm.inf.ethz.ch/education/courses/program-verification.html Encoding to SAT)
https://theory.stanford.edu/~nikolaj/programmingz3.html
<<Quantifiers in Satisfiability Modulo Theories>> SAT/SMT/AR Summer School 2018

Predicate Logic（谓词逻辑）

1.1 Propositional Logic

Step 0. Proof.

Step 1. Convert program into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

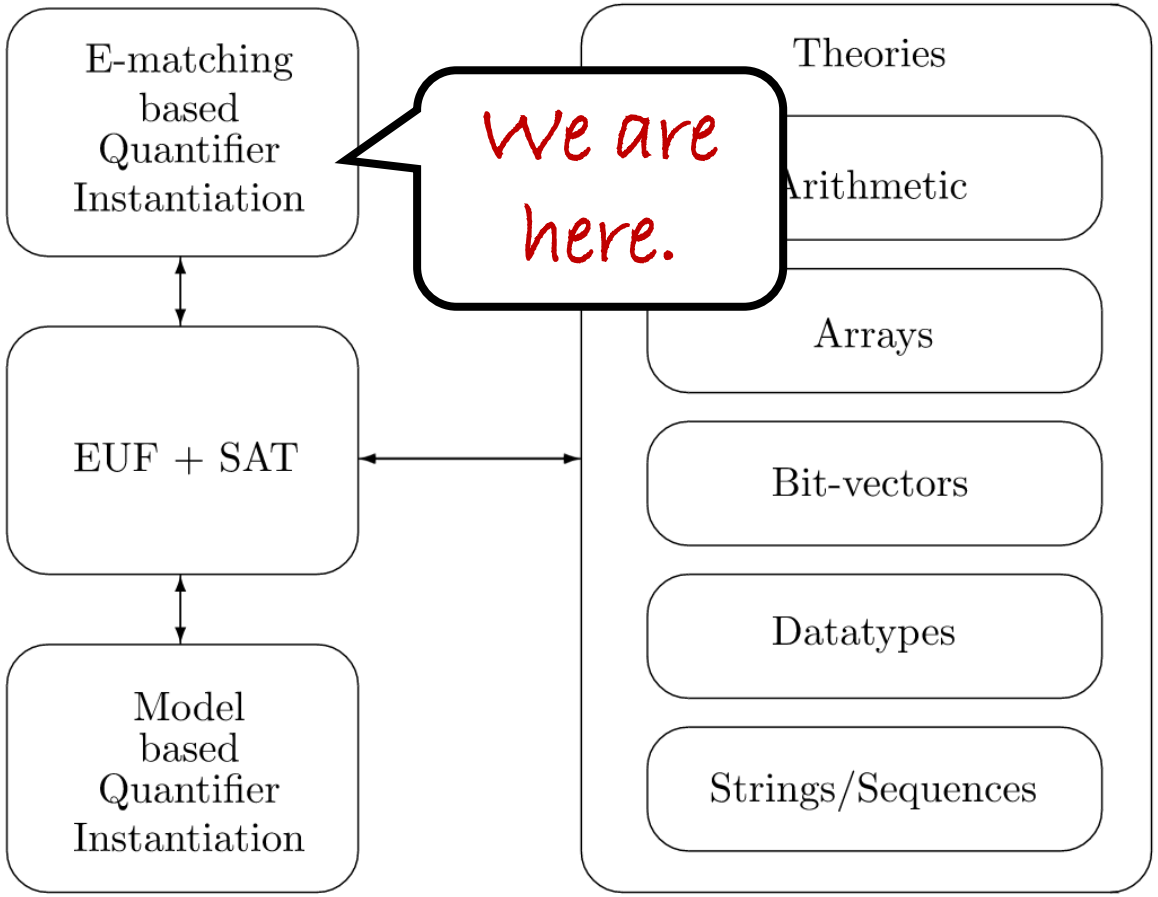
Step 1. Convert it into first logic formula.

Step 2. Ask the computer to solve the formula.

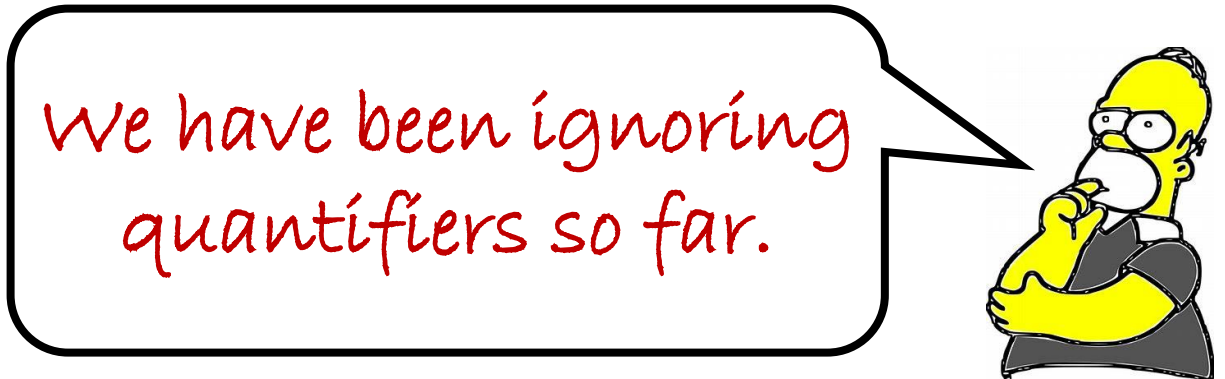
1.3 Interactive Proof

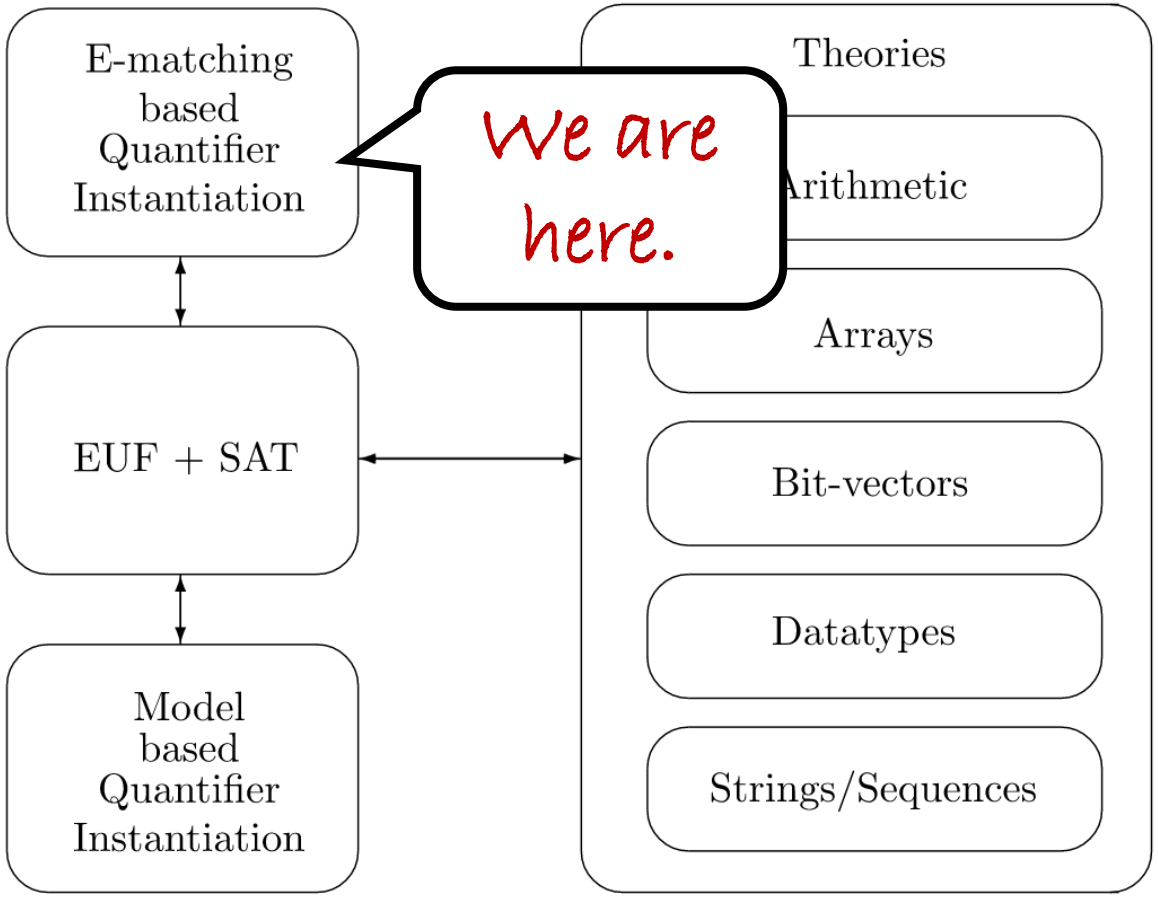
Step 1. Axiom system

Step 2. Ask the computer to check the invariants



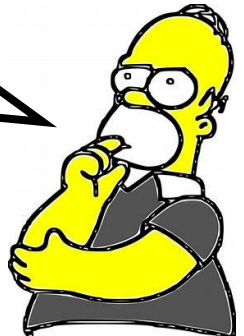
Architecture of Z3's SMT Core solver

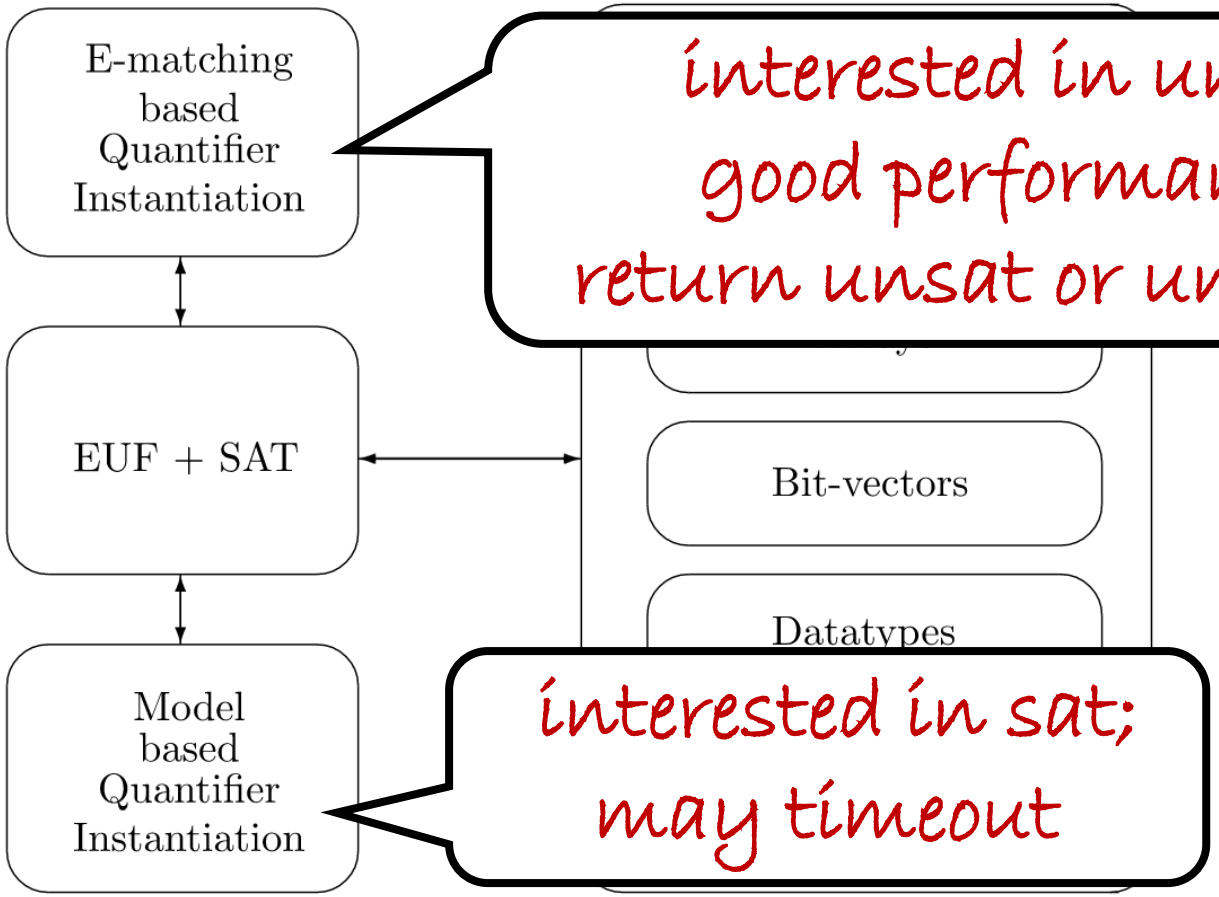




Architecture of Z3's SMT Core solver

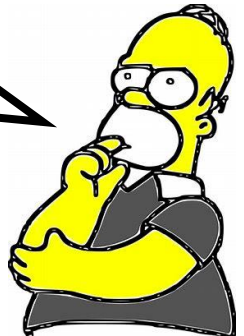
It is very difficult because domain of discourse is infinite.





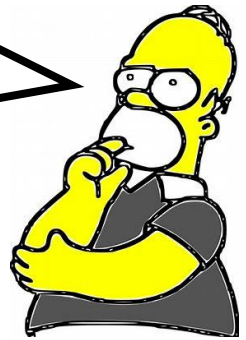
Architecture of Z3's SMT Core solver

It is very difficult because domain of discourse is infinite.



Quantifier

To make things easier, we can eliminate all existential quantifiers.



Quantifier

$$\neg(\forall x)(P(x)) = (\exists x)(\neg P(x))$$
$$\neg(\exists x)(P(x)) = (\forall x)(\neg P(x))$$

1) Move “not” inwards.
2) Use skolemization to eliminate existential quantifiers.
3) Now we obtain a formula with only positive universally-quantified literals.

There is no “not” before quantifiers.

A Quick Recap

$$(\forall x)(\exists y)(\exists z)S(a,b,x,y,z)$$

y depends on x

$$(\forall x)(\exists z)S(a,b,x,f(x),z)$$

$$(\forall x)S(a,b,x,f(x),g(x))$$

z depends on x

We don't need to move all the quantifiers to the front of all formulas. We directly do skolemization on quantifiers.

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a

$$g(f(a)) = 0 \wedge (\forall x)(g(f(x)) = 1)$$

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a

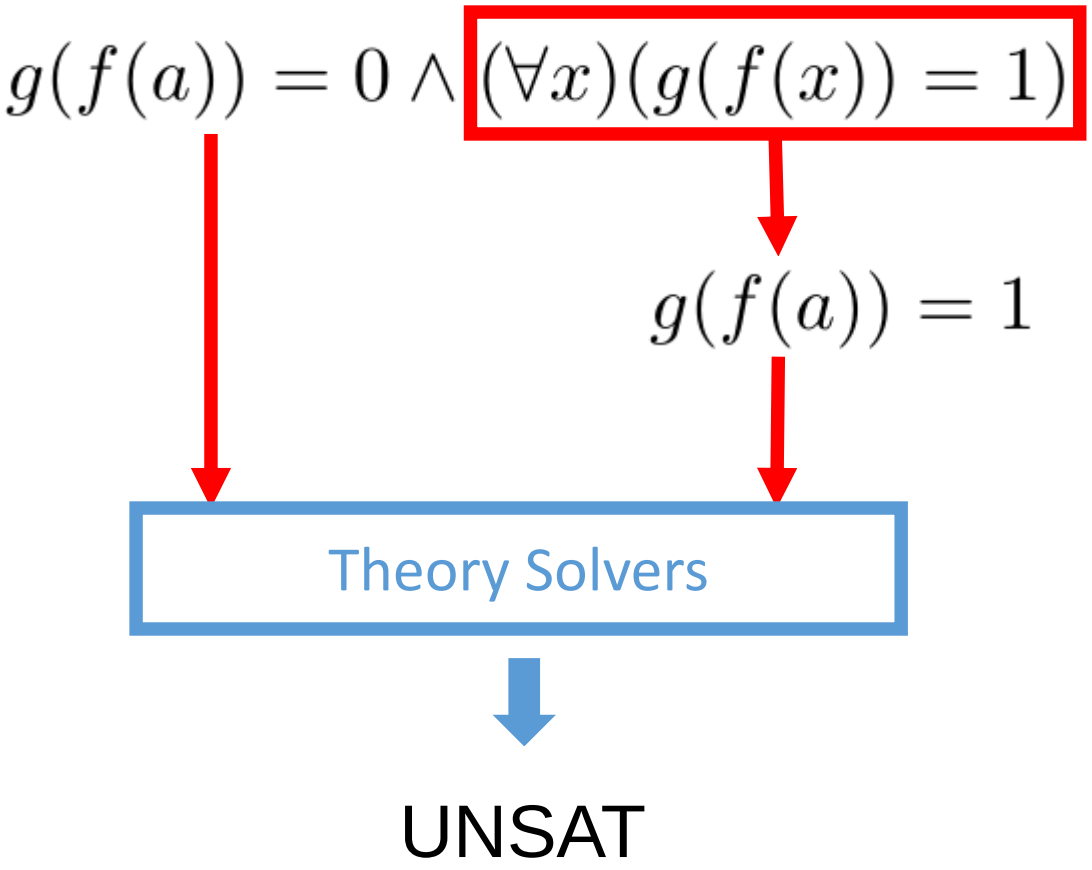
$$g(f(a)) = 0 \wedge (\forall x)(g(f(x)) = 1)$$

$$g(f(a)) = 1$$

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a



Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a.

$$g(f(a)) = 0 \wedge (\forall x)(g(f(x)) = 1)$$

$$g(f(a)) = 1$$

Theory Solvers

How to combine instantiation with SMT?

Quantifier

DEFINITION

A quantified literal is a formula $(\forall x)P(x)$ or its negation $\neg(\forall x)P(x)$.

Now, there are only positive universally-quantified literals.

$$(\forall x)(x > 0) \qquad (\forall x)(f(x) = g(x))$$

$$(\forall x)(f(x) = 3 \vee g(x) = 5)$$

Quantifier

DEFINITION

A quantified literal is a formula $(\forall x)P(x)$ or its negation $\neg(\forall x)P(x)$.

DEFINITION

An extended clause is a disjunction of literals and quantified literals.

$$(\forall x)(x > 0) \vee a = 5 \vee b > 3$$

Quantifier

DEFINITION

A quantified literal is a formula $(\forall x)P(x)$ or its negation $\neg(\forall x)P(x)$.

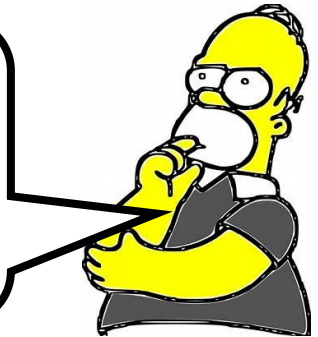
DEFINITION

An extended clause is a disjunction of literals and quantified literals.

DEFINITION

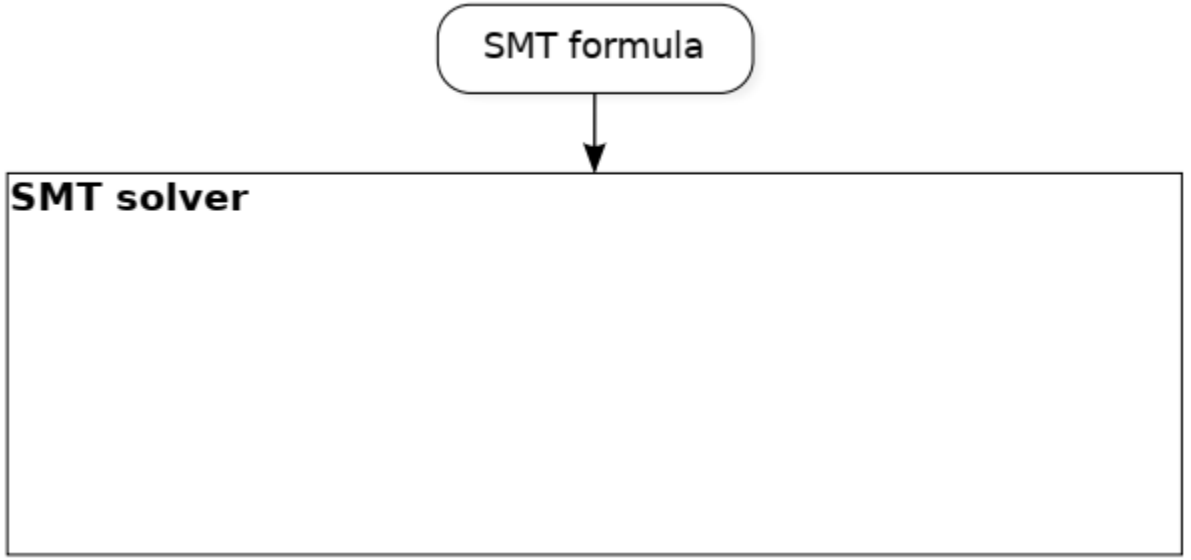
A formula is in extended CNF iff it is a conjunction of extended clauses.

We can convert a formula to extended CNF with method in propositional logic.



Quantifier

$$0 \leq b \quad \wedge \quad \forall x.\, (x \not\geq 0 \vee f(x) = a) \quad \wedge \quad f(b) \neq a$$

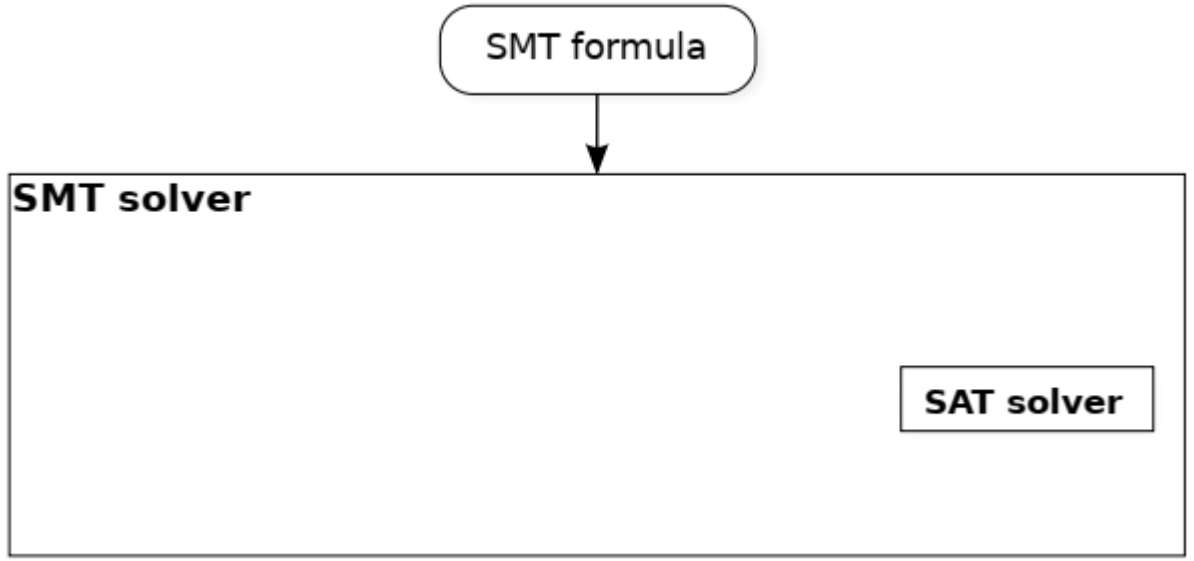


Quantifier

$$0 \leq b \quad \wedge \quad \forall x.\,(x \not\geq 0 \vee f(x) = a) \quad \wedge \quad f(b) \neq a$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



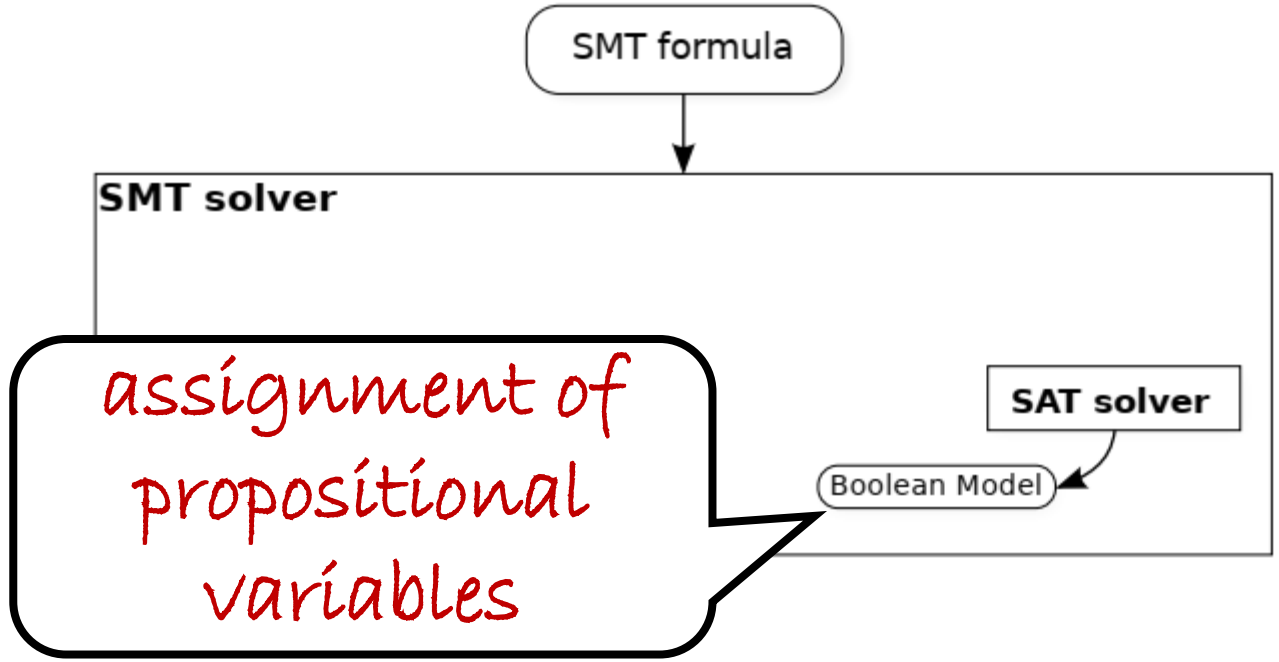
$$\emptyset \parallel \; l_1,\; l_2,\; l_3 \qquad \Longrightarrow (UnitProp)$$

Quantifier

$$0 \leq b \quad \wedge \quad \forall x.\,(x \not\geq 0 \vee f(x) = a) \quad \wedge \quad f(b) \neq a$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



$$\emptyset \parallel \; l_1,\; l_2,\; l_3$$

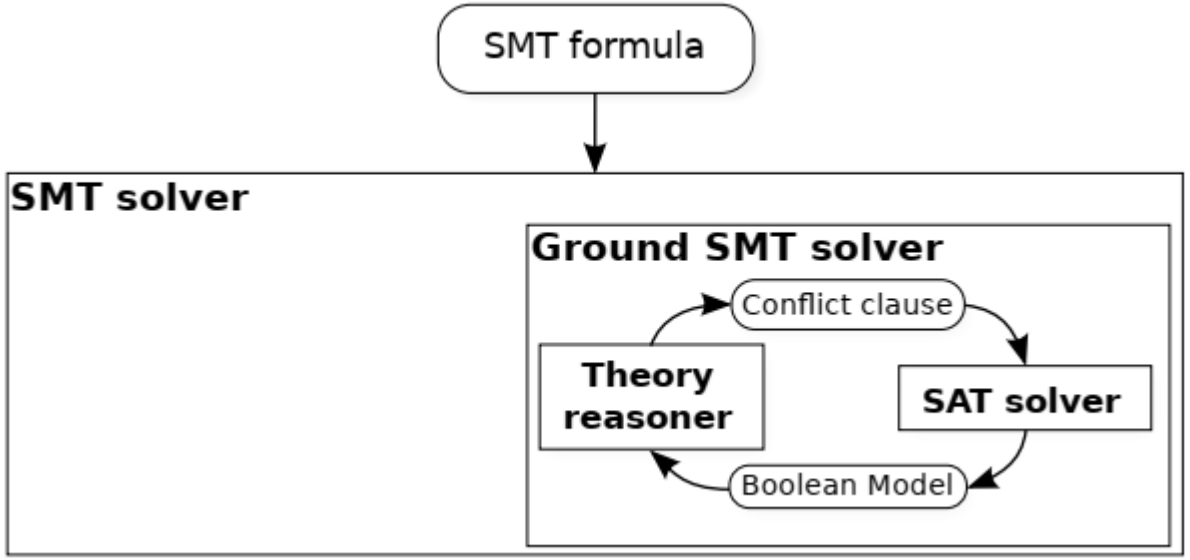
$$l_1 \;\; l_2 \;\; l_3 \parallel \; l_1,\; l_2,\; l_3 \qquad \Longrightarrow (UnitProp)$$

Quantifier

$$0 \leq b \quad \wedge \quad \forall x.\,(x \not\geq 0 \vee f(x) = a) \quad \wedge \quad f(b) \neq a$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



$$\emptyset \parallel \; l_1,\; l_2,\; l_3$$

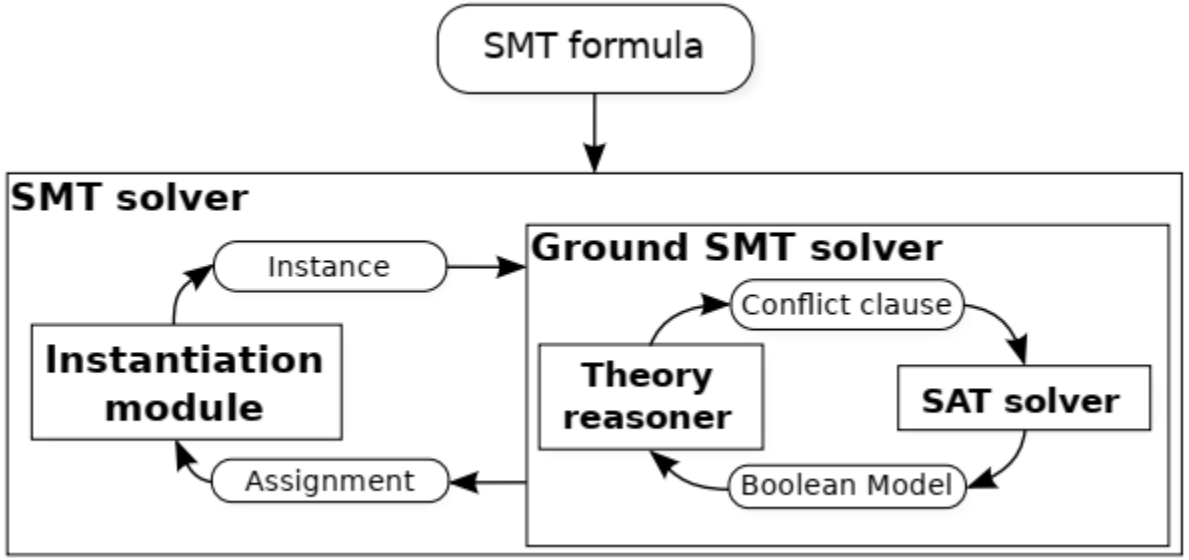
$$l_1 \;\; l_2 \;\; l_3 \parallel \; l_1,\; l_2,\; l_3 \qquad \Longrightarrow (UnitProp)$$

Quantifier

$$0 \leq b \quad \wedge \quad \forall x.(x \not\geq 0 \vee f(x) = a) \quad \wedge \quad f(b) \neq a$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



$$\emptyset \parallel \ l_1, \ l_2, \ l_3 \qquad \Longrightarrow (UnitProp)$$

$$l_1 \ l_2 \ l_3 \parallel \ l_1, \ l_2, \ l_3 \qquad \Longrightarrow (\forall - Inst)$$

$$l_1 \ l_2 \ l_3 \parallel \ l_1, \ l_2, \ l_3, \ \neg l_2 \vee b \not\geq 0 \vee f(b) = a$$

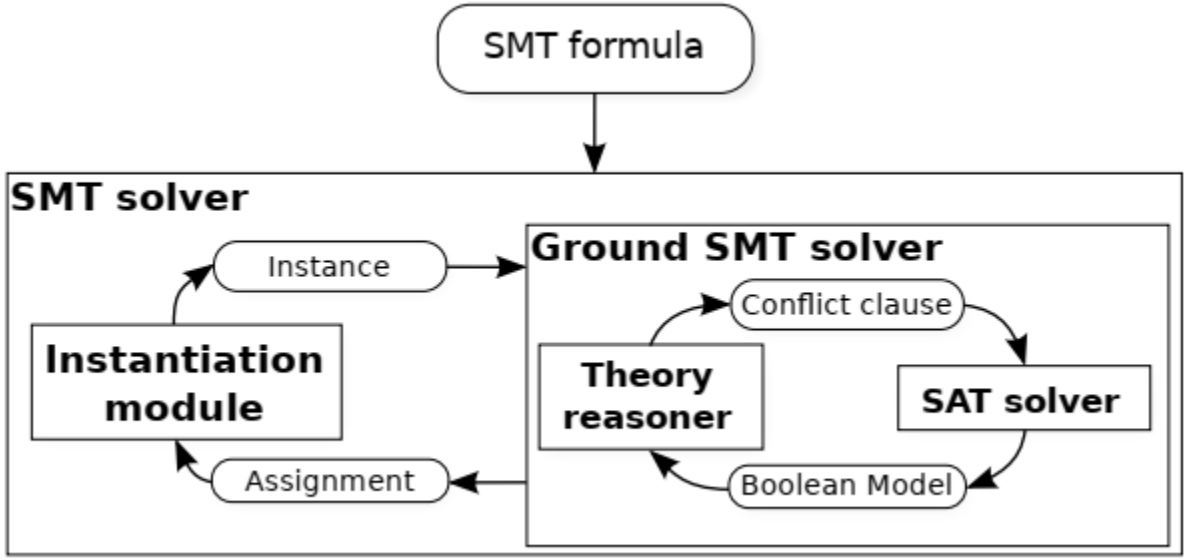
preserve the satisfiability of formula

Quantifier

$$0 \leq b \quad \wedge \quad \forall x.\,(x \not\geq 0 \vee f(x) = a) \quad \wedge \quad f(b) \neq a$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



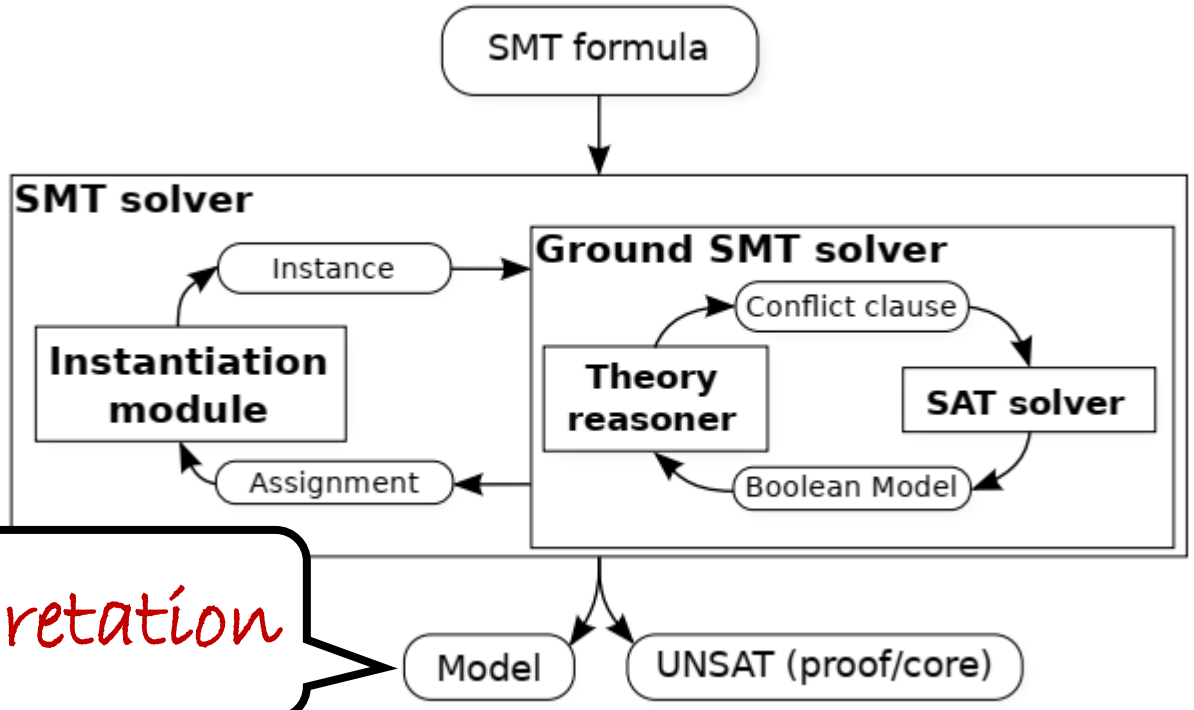
$$\begin{aligned}
\emptyset \parallel \; & l_1, \; l_2, \; l_3 && \Longrightarrow (UnitProp)\\
l_1 \; l_2 \; l_3 \parallel \; & l_1, \; l_2, \; l_3 && \Longrightarrow (\forall - Inst)\\
l_1 \; l_2 \; l_3 \parallel \; & l_1, \; l_2, \; l_3, \; \neg l_2 \vee b \not\geq 0 \vee f(b) = a && \Longrightarrow (T - Prop)\\
l_1 \; l_2 \; l_3 \; b \geq 0 \parallel \; & l_1, \; l_2, \; l_3, \; \neg l_2 \vee b \not\geq 0 \vee f(b) = a &&
\end{aligned}$$

Quantifier

$$0 \leq b \quad \wedge \quad \forall x.\,(x \not\geq 0 \vee f(x) = a) \quad \wedge \quad f(b) \neq a$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



$$\begin{array}{rll}
\emptyset \parallel & l_1,\ l_2,\ l_3 & \Longrightarrow (UnitProp) \\
l_1\ l_2\ l_3 \parallel & l_1,\ l_2,\ l_3 & \Longrightarrow (\forall - Inst) \\
l_1\ l_2\ l_3 \parallel & l_1,\ l_2,\ l_3,\ \neg l_2 \vee b \not\geq 0 \vee f(b) = a & \Longrightarrow (T - Prop) \\
l_1\ l_2\ l_3\ b \geq 0 \parallel & l_1,\ l_2,\ l_3,\ \neg l_2 \vee b \not\geq 0 \vee f(b) = a & \Longrightarrow (Fail) \\
fail & &
\end{array}$$

The remaining problem is how to generate instances.

Trigger Matching

DEFINITION

A trigger is a term t, satisfying the following criteria:

1) t must contain all of the variables quantified by the quantifier

2) t must contain at least one non-constant function symbol

$$(\forall x.\{f(x)\})(f(x) > 0)$$

trigger

$$(\forall x.\{g(f(x))\})(g(f(x)) = 1)$$

trigger

Trigger Matching

DEFINITION

A ground term is a term containing no quantified variables.

$(\forall x.\{f(x)\})(f(x) > 0)$

Not ground term

$f(3)$

ground term

We can compare the form of triggers and ground terms to generate instances

Trigger Matching

DEFINITION

A ground term is a term containing no quantified variables.

$(\forall x.\{f(x)\})(f(x) > 0)$

Not ground term

$f(3)$

ground term

$g(f(a)) = 0 \wedge (\forall x.\{g(f(x))\})(g(f(x)) = 1)$

$g(f(a)) = 1$

Term g(f(a)) matches the trigger, causing the instantiation.

Trigger Matching

Without a matching ground term, no information will be deduced from the quantifier body.

$$(\forall x.p(x))(p(x) = 1) \wedge (\forall x.p(x))(p(x) = 0)$$

With trigger matching, solver returns unknown rather than sat.

Trigger Matching

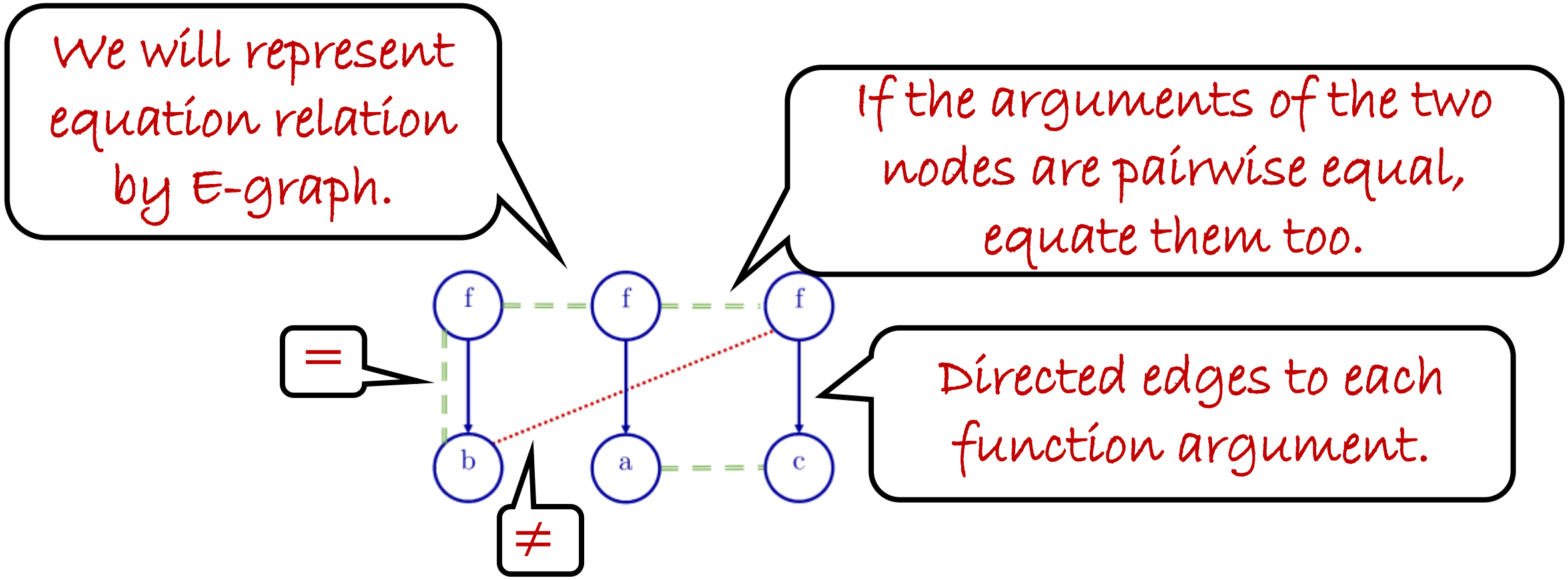
Without a matching ground term, no information will be deduced from the quantifier body.

$$g(b) = 0 \wedge b = f(a) \wedge (\forall x.\{g(f(x))\})(g(f(x)) = 1)$$

But we can address this by E-graph.

E-graph

$$f(b) = b, f(b) = f(a), f(c) \neq b, a = c$$

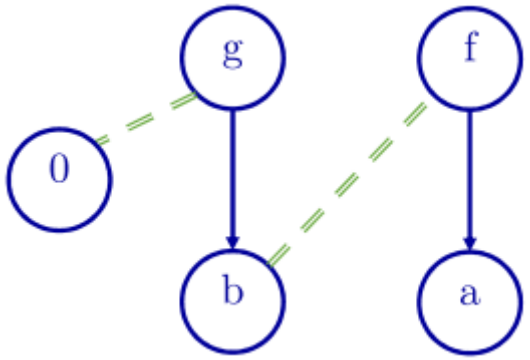


E-matching

Triggers are matched modulo equalities (E-matching):

E-matching can be efficiently implemented by pattern-matching triggers against an E-graph (representing known equivalence classes on terms)

$$g(b) = 0 \wedge b = f(a) \wedge (\forall x.\{g(f(x))\})(g(f(x)) = 1)$$



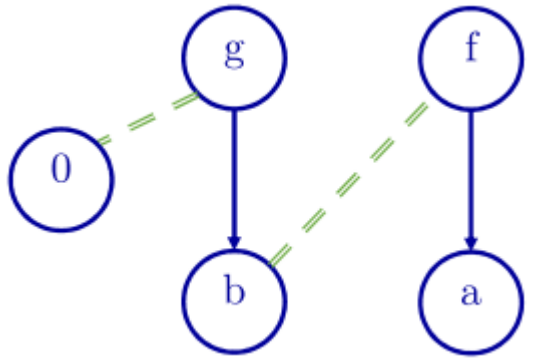
E-matching

Match trigger against E-graph:

- start from each g node
- for their (only) argument nodes, search the equivalence class for any f nodes
- use argument of (each) f for instantiation
- We get a match for g(f(a))

$$g(b) = 0 \wedge b = f(a) \wedge (\forall x.\{g(f(x))\})(g(f(x)) = 1)$$

$$g(f(a)) = 1$$

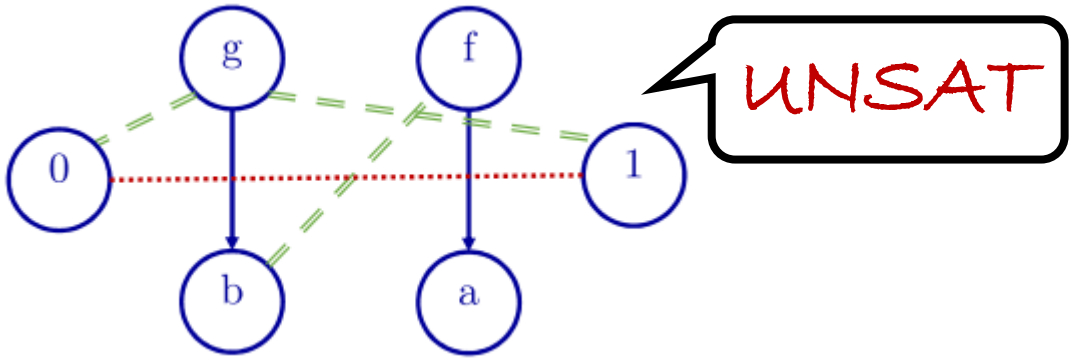


E-matching

Match trigger against E-graph:

- start from each g node
- for their (only) argument nodes, search the equivalence class for any f nodes
- use argument of (each) f for instantiation
- We get a match for g(f(a))

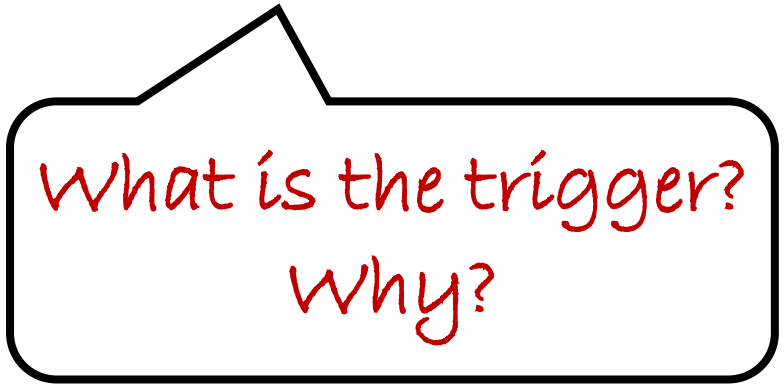
$$g(b) = 0 \wedge b = f(a) \wedge g(f(a)) = 1$$



E-matching

- E-matching depends heavily on carefully-chosen triggers.

$$\neg(a = b), f(a) = f(b), (\forall x)(g(f(x)) = x)$$



E-matching

E-matching depends heavily on carefully-chosen triggers.

$$\neg(a = b), f(a) = f(b), (\forall x)(g(f(x)) = x)$$

The trigger is f(x) instead of g(f (x)). Otherwise, you cannot prove unsat.

With only E-matching, SMT solver will return unsat or unknown, which is suitable for program verification and analysis.

Model-Based Quantifier Instantiation can return sat and give the interpretation.

Thanks & Questions