SMT Solver

Zhaoguo Wang

Adapted From:

ETH Zurich program verification, Lecture 2, 3
(https://www.pm.inf.ethz.ch/education/courses/program-verification.html)
https://theory.stanford.edu/~nikolaj/programmingz3.html
<<Solving SAT and SAT Modulo Theories: From an Abstract Davis–Putnam–Logemann–Loveland Procedure to DPLL(T)>>

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 order logic formula.

Step 2. Ask the computer to solve the formula.

1.3 Auto-active Proof

Step 1. Axiom system

Step 2. Ask the computer to check the invariants

Outline – This Lecture

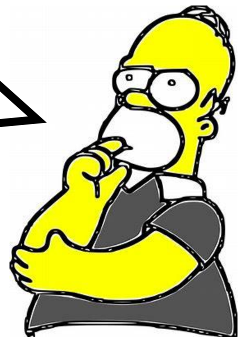
- SMT Solver
  - SMT
  - From DPLL to DPLL(T)
  - EUF

SMT

DEFINITION

The SMT problem (abbreviated satisfiability modulo theories) is the problem of determining if there exists an interpretation that satisfies a given predicate logic formula.

we can use SAT solvers to solve SAT problems automatically. Can we solve SMT problems automatically?



MiniSmt Computational Logic

MathSAT 5

An SMT Solver for Formal Verification & More

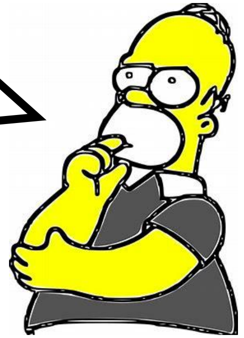
SMT solver can do it!

Z3

CVC3

CVC4

There are some powerful SAT solvers based on DPLL. Can we build SMT solvers based on these SAT solvers?



First, we consider the case of *quantifier-free formulas*.

A Quick Recap

$$\emptyset \parallel \overline{1} \vee \overline{2},\ \ 2 \vee 3,\ \ \overline{1} \vee \overline{3} \vee 4,\ \ 2 \vee \overline{3} \vee \overline{4},\ \ 1 \vee 4 \quad \Longrightarrow \quad \text{(Decide)}$$

A Quick Recap

$$\emptyset \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 \quad \Longrightarrow \quad \text{(Decide)}$$
$$1^{\mathrm{d}} \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 \quad \Longrightarrow \quad \text{(UnitPropagate)}$$

A Quick Recap

$$
\begin{array}{rcl}
\emptyset \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
1^{\mathsf{d}} \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2} \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)}
\end{array}
$$

A Quick Recap

$$
\begin{array}{rcl}
\emptyset \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
1^{\mathsf{d}} \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2} \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2}\ 3 \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)}
\end{array}
$$

A Quick Recap

$$
\begin{array}{rcl}
\emptyset \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
1^{\mathsf{d}} \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2} \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3 \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\,\overline{2}\,3\,4 \;\|\; \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow & \text{(Backtrack)}
\end{array}
$$

A Quick Recap

$$
\begin{array}{rcll}
\emptyset & \| & \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow \quad \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\;\overline{2} & \| & \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\;\overline{2}\;3 & \| & \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow \quad \text{(UnitPropagate)} \\
1^{\mathsf{d}}\;\overline{2}\;3\;4 & \| & \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow \quad \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2},\;\; 2\vee 3,\;\; \overline{1}\vee\overline{3}\vee 4,\;\; 2\vee\overline{3}\vee\overline{4},\;\; 1\vee 4 & \Longrightarrow \quad \text{(UnitPropagate)}
\end{array}
$$

A Quick Recap

$$
\begin{array}{rlllllll}
\emptyset & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & \Longrightarrow \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & \Longrightarrow \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & \Longrightarrow \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2}\ 3 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & \Longrightarrow \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2}\ 3\ 4 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & \Longrightarrow \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & \Longrightarrow \text{(UnitPropagate)} \\
\overline{1}\ 4 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & \Longrightarrow \text{(Decide)}
\end{array}
$$

A Quick Recap

$$
\begin{array}{rcllllllcl}
\emptyset & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(Decide)} \\
1^{\mathsf{d}} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2}\ 3 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2}\ 3\ 4 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(UnitPropagate)} \\
\overline{1}\ 4 & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(Decide)} \\
\overline{1}\ 4\ \overline{3}^{\mathsf{d}} & \| & \overline{1}\vee\overline{2}, & 2\vee 3, & \overline{1}\vee\overline{3}\vee 4, & 2\vee\overline{3}\vee\overline{4}, & 1\vee 4 & & \Longrightarrow & \text{(UnitPropagate)}
\end{array}
$$

A Quick Recap

$$\begin{array}{rcl}
\emptyset \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
1^{\mathsf{d}} \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2} \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2}\ 3 \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{\mathsf{d}}\ \overline{2}\ 3\ 4 \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(Backtrack)} \\
\overline{1} \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
\overline{1}\ 4 \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
\overline{1}\ 4\ \overline{3}^{\mathsf{d}} \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
\overline{1}\ 4\ \overline{3}^{\mathsf{d}}\ 2 \parallel \overline{1}\vee\overline{2},\ 2\vee 3,\ \overline{1}\vee\overline{3}\vee 4,\ 2\vee\overline{3}\vee\overline{4},\ 1\vee 4 & &
\end{array}$$

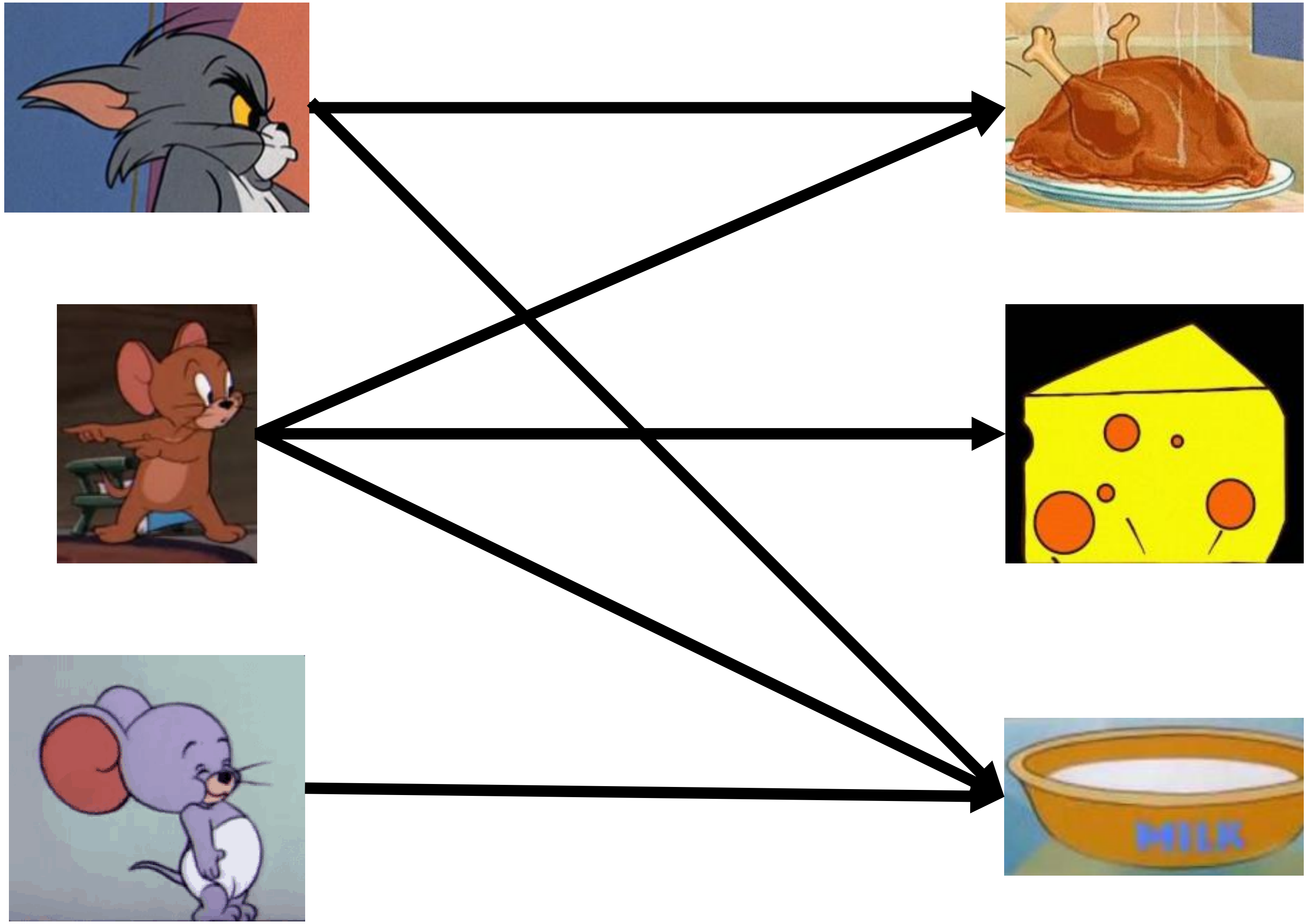
A Quick Recap

$$
\begin{array}{rcl@{\quad}c@{\quad}l}
\emptyset & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
1^{d} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{d}\ \overline{2} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{d}\ \overline{2}\ 3 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
1^{d}\ \overline{2}\ 3\ 4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(Backtrack)} \\
\overline{1} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
\overline{1}\ 4 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(Decide)} \\
\overline{1}\ 4\ \overline{3}^{d} & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & \Longrightarrow & \text{(UnitPropagate)} \\
\overline{1}\ 4\ \overline{3}^{d}\ 2 & \| & \overline{1}\vee\overline{2},\ \ 2\vee 3,\ \ \overline{1}\vee\overline{3}\vee 4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee 4 & &
\end{array}
$$

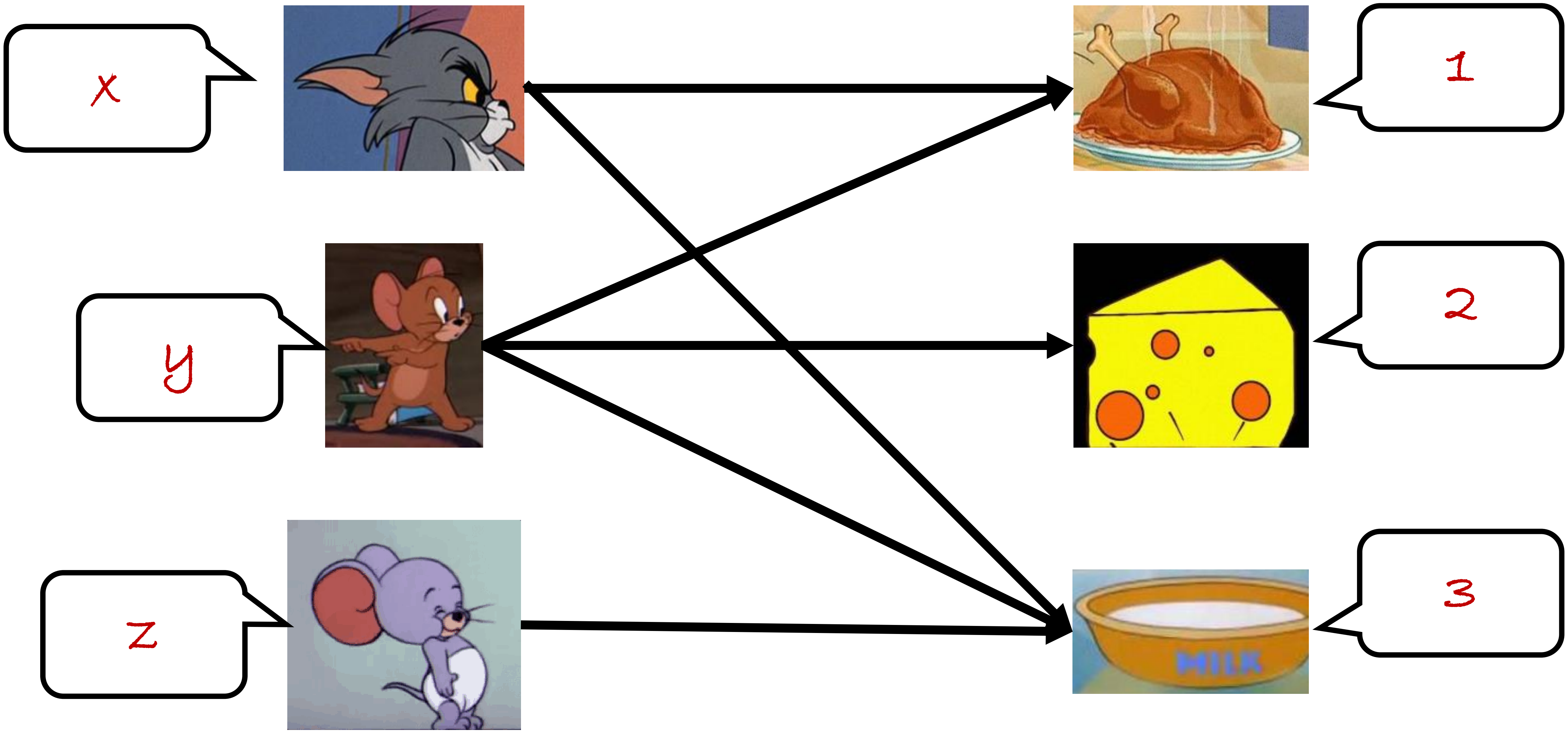
The input of a SAT solver is CNF and it uses DPLL to solve the problem.

Eager SMT converts a theory into a SAT problem

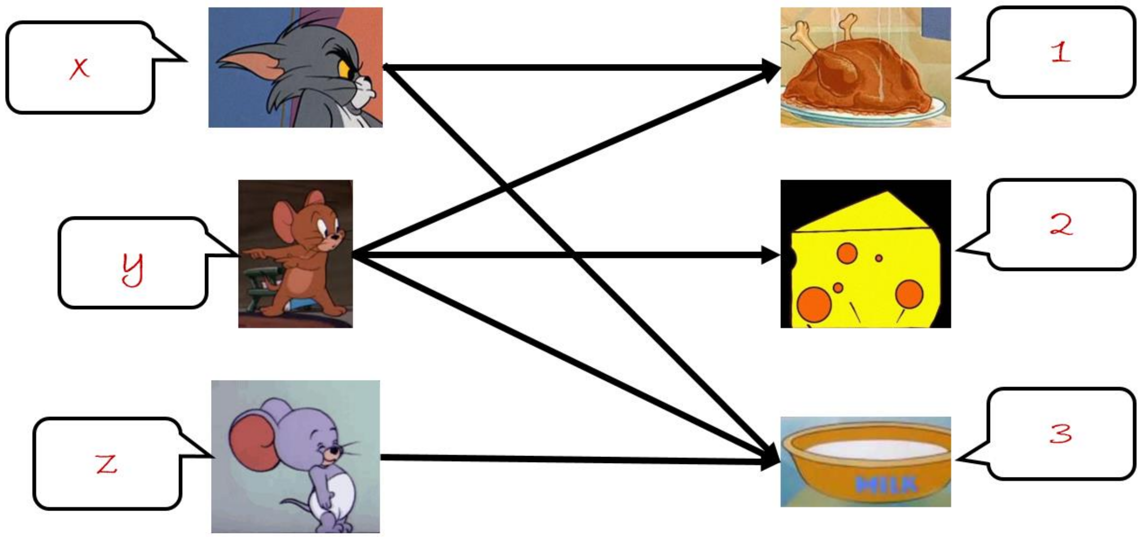
Eager SMT Techniques



Eager SMT Techniques



Eager SMT Techniques



$$(x = 1 \vee x = 3)\wedge$$

$$(y = 1 \vee y = 2 \vee y = 3)\wedge$$

$$(z = 3)\wedge$$

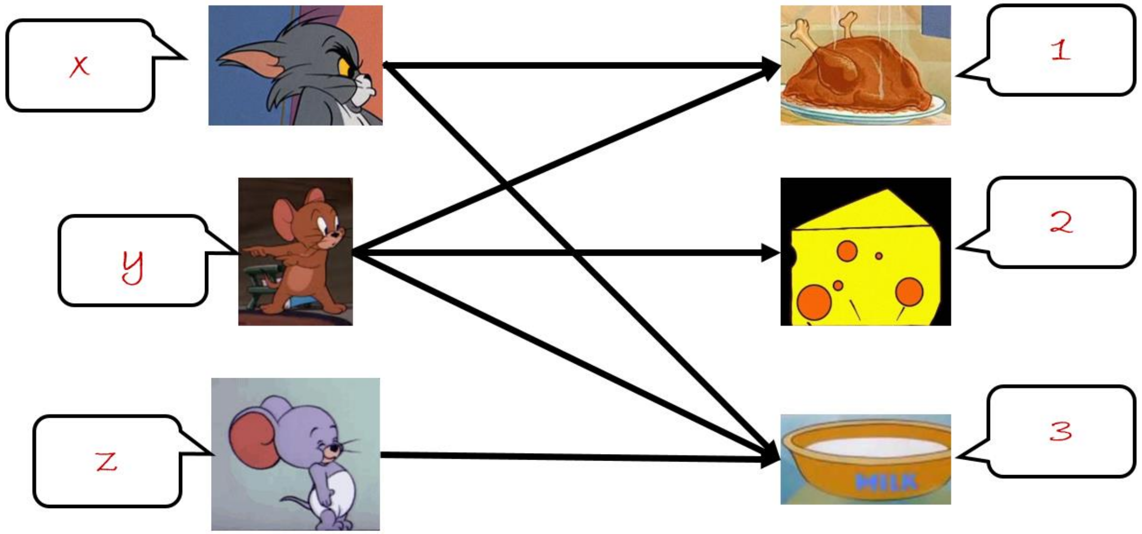
$$(x \neq y)\wedge$$

$$(x \neq z)\wedge$$

$$(y \neq z)$$

predicate logic

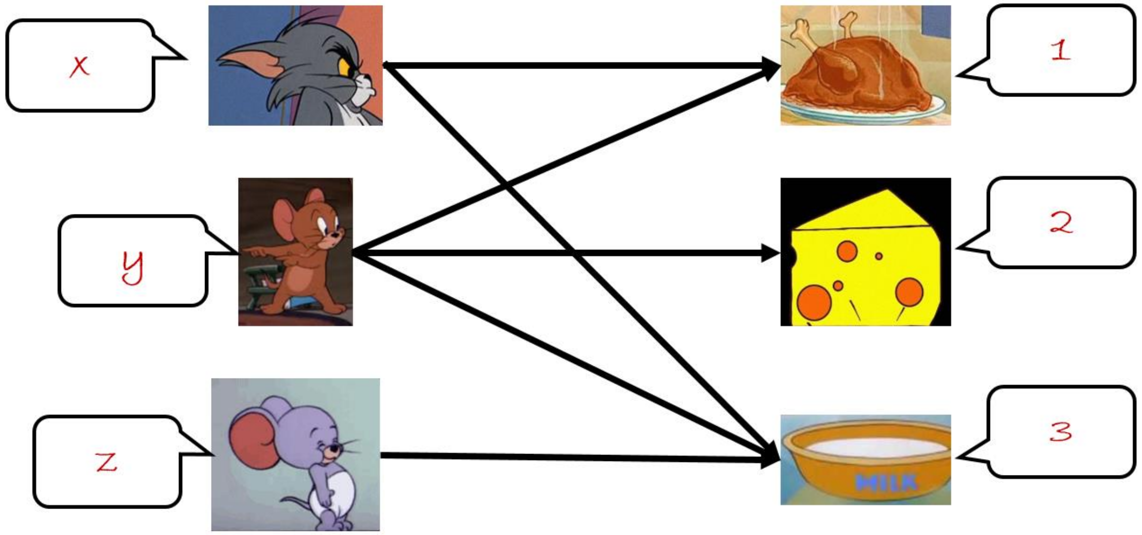
Eager SMT Techniques



$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$(x \neq y) \wedge$$
$$(x \neq z) \wedge$$
$$(y \neq z)$$

The input formula can be translated using a satisfiability-preserving transformation into a propositional CNF formula.

Eager SMT Techniques

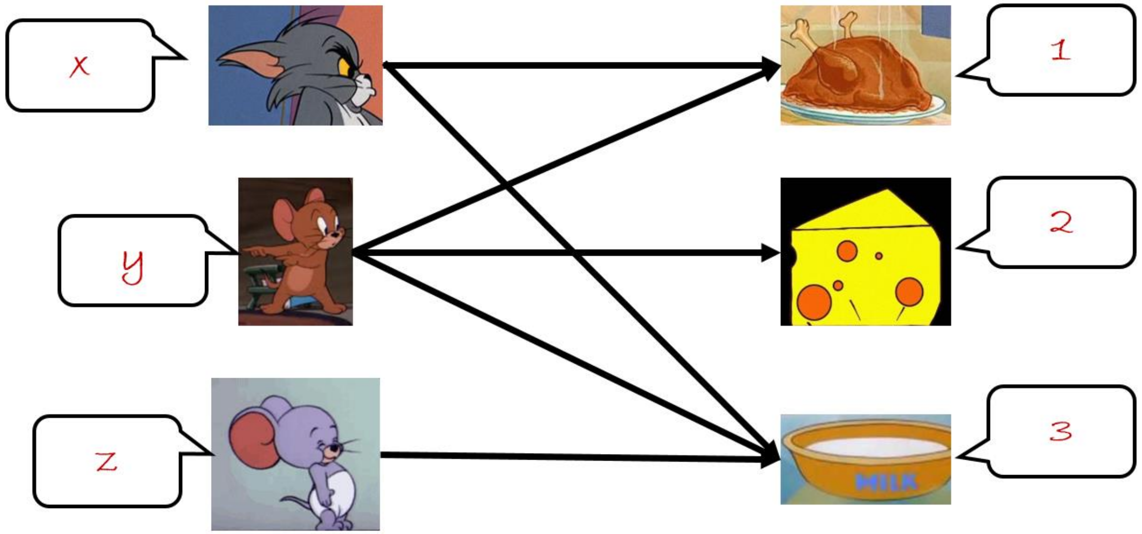


$$
\begin{aligned}
&(x = 1 \lor x = 3)\land \\
&(y = 1 \lor y = 2 \lor y = 3)\land \\
&(z = 3)\land \\
&(x \neq y)\land \\
&(x \neq z)\land \\
&(y \neq z)
\end{aligned}
$$

Represent x, y and z by two variables each. They are similar to bits.

x →	p1	p0
y →	q1	q0
z →	r1	r0

Eager SMT Techniques

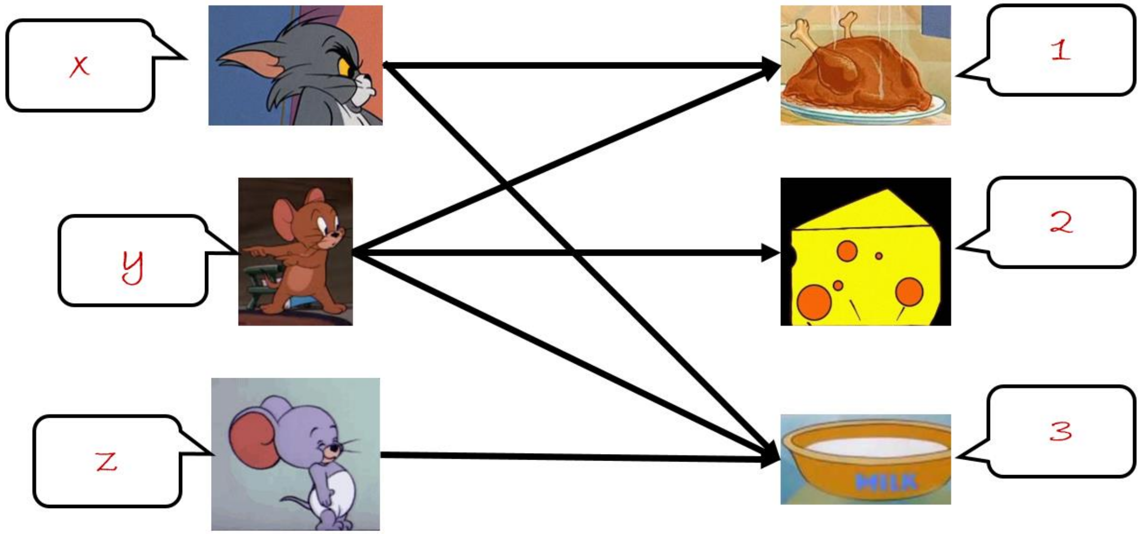


$$(x = 1 \lor x = 3) \land$$
$$(y = 1 \lor y = 2 \lor y = 3) \land$$
$$(z = 3) \land$$
$$(x \neq y) \land$$
$$(x \neq z) \land$$
$$(y \neq z)$$

$$((\neg p1 \land p0) \lor (p1 \land p0)) \land$$
$$((\neg q1 \land q0) \lor (q1 \land \neg q0) \lor (q1 \land q0)) \land$$
$$(r1 \land r0) \land$$
$$\neg((p1 \leftrightarrow q1) \land (p0 \leftrightarrow q0)) \land$$
$$\neg((q1 \leftrightarrow r1) \land (q0 \leftrightarrow r0)) \land$$
$$\neg((p1 \leftrightarrow r1) \land (p0 \leftrightarrow r0))$$

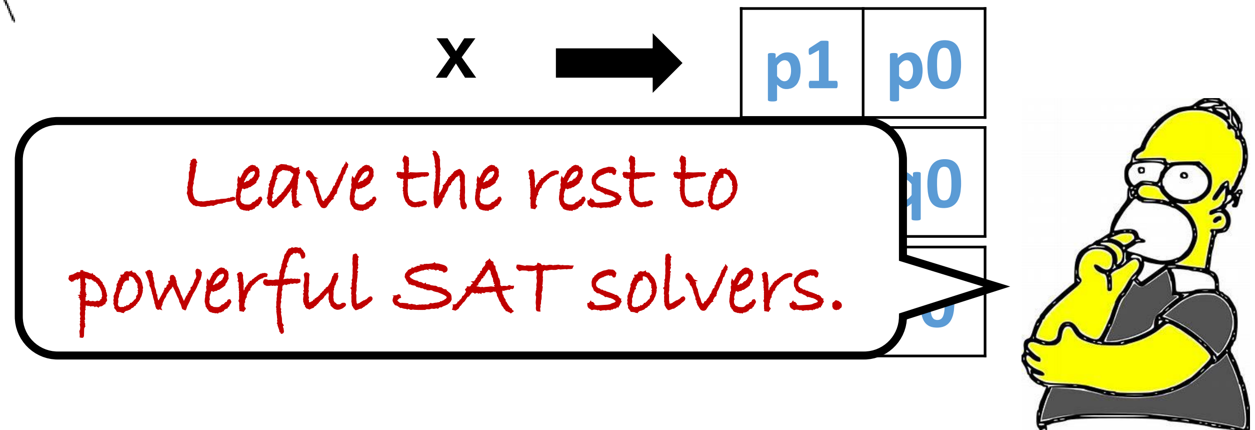
x →	p1	p0
y →	q1	q0
z →	r1	r0

Eager SMT Techniques



$$
\begin{aligned}
&(x = 1 \lor x = 3) \land \\
&(y = 1 \lor y = 2 \lor y = 3) \land \\
&(z = 3) \land \\
&(x \neq y) \land \\
&(x \neq z) \land \\
&(y \neq z)
\end{aligned}
$$

$$
\begin{aligned}
&((\neg p1 \land p0) \lor (p1 \land p0)) \land \\
&((\neg q1 \land q0) \lor (q1 \land \neg q0) \lor (q1 \land q0)) \land \\
&(r1 \land r0) \land \\
&\neg((p1 \leftrightarrow q1) \land (p0 \leftrightarrow q0)) \land \\
&\neg((q1 \leftrightarrow r1) \land (q0 \leftrightarrow r0)) \land \\
&\neg((p1 \leftrightarrow r1) \land (p0 \leftrightarrow r0))
\end{aligned}
$$



Eager SMT Techniques

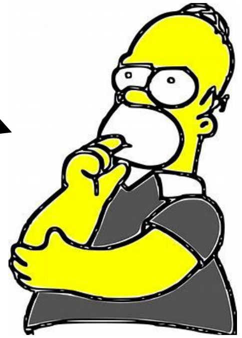
Eager SMT techniques require to encode problems to SAT:

1) Represent the state space

2) Convert the problem into the new representation

3) (Optional) Reduce redundancy

What if there are uninterpreted functions?



Eager SMT Techniques

X, y and z are bool type.

$$y = f(z) \wedge x = f(f(z)) \wedge \neg(x = f(y))$$

Function f is uninterpreted.

We want to encode it to SAT, including f.

Eager SMT Techniques

$$y = f(z) \wedge x = f(f(z)) \wedge \neg(x = f(y))$$

$$\Downarrow$$

Replace functions with fresh variables. Fresh variables are variables that never exist in original formula.

$$(y \leftrightarrow f_z) \wedge$$

$$(x \leftrightarrow f_{fz}) \wedge$$

$$\neg(x \leftrightarrow f_y)$$

Is the result equisatisfiable with the original formula?

Eager SMT Techniques

x=f(f(z))=f(y)

$$y = f(z) \wedge x = f(f(z)) \wedge \neg(x = f(y))$$

UNSAT

⬇

$$(y \leftrightarrow f_z)\wedge$$

$$(x \leftrightarrow f_{fz})\wedge$$

$$\neg(x \leftrightarrow f_y)$$

Eager SMT Techniques

x=f(f(z))=f(y)

$$y = f(z) \wedge x = f(f(z)) \wedge \neg(x = f(y))$$

UNSAT

$$\Downarrow$$

$$(y \leftrightarrow f_z)\wedge$$

$$(x \leftrightarrow f_{fz})\wedge$$

$$\neg(x \leftrightarrow f_y)$$

SAT

$$y = T, f_z = T, x = T$$

$$f_{fz} = T, f_y = F$$

Eager SMT Techniques

x=f(f(z))=f(y)

$$y = f(z) \wedge x = f(f(z)) \wedge \neg(x = f(y))$$

$$\Downarrow$$

UNSAT

SAT

$$(y \leftrightarrow f_z)\wedge$$
$$(x \leftrightarrow f_{fz})\wedge$$
$$\neg(x \leftrightarrow f_y)$$

$y = T, f_z = T, x = T$

$f_{fz} = T, f_y = F$

Without the relation between f(y) and f(f(z)): f(y)=f(f(z)).

Eager SMT Techniques

$$y = f(z) \wedge x = f(f(z)) \wedge \neg(x = f(y))$$

$$\Downarrow$$

$$\boxed{\begin{array}{l} ((y \leftrightarrow z) \rightarrow (f_y \leftrightarrow f_z))\wedge \\ ((y \leftrightarrow f_z) \rightarrow (f_y \leftrightarrow f_{fz})\wedge \\ ((z \leftrightarrow f_z) \rightarrow (f_z \leftrightarrow f_{fz}))\wedge \end{array}}$$

$$(y \leftrightarrow f_z)\wedge$$

$$(x \leftrightarrow f_{fz})\wedge$$

$$\neg(x \leftrightarrow f_y)$$

Property of function f:

$$a = b \rightarrow f(a) = f(b)$$

UNSAT

Eager SMT Techniques

We can always use the best available SAT solver off the shelf.

Eager SMT Techniques

We can always use the best available SAT solver off the shelf.

Issues:

Not very flexible to make them efficient,

Sophisticated translations are required for each theory.

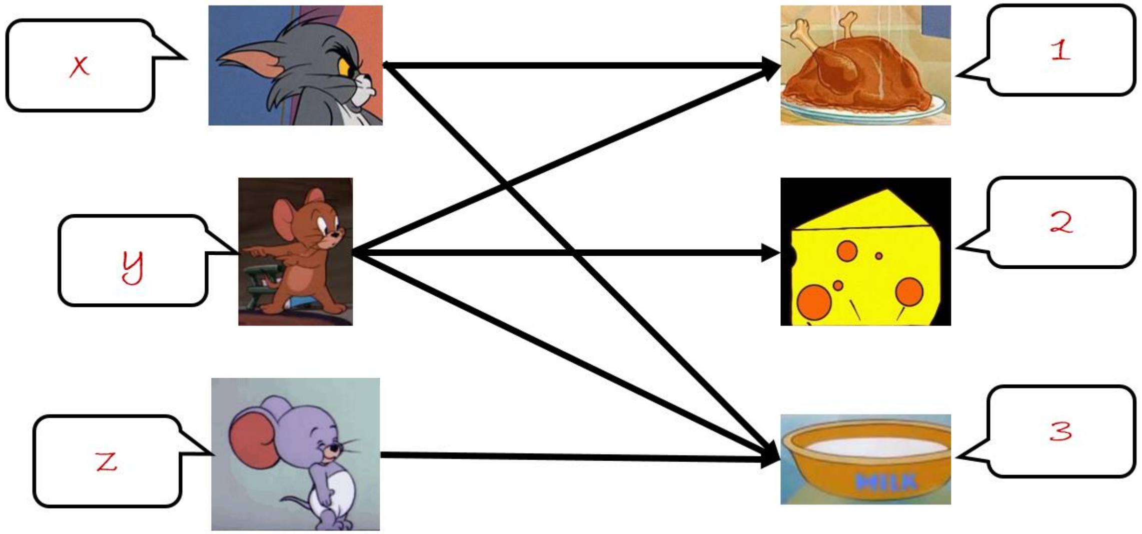
On many practical problems the translation process or the SAT solver run out of time or memory.

Lazy SMT integrates theory support into the propositional search

Lazy SMT Techniques

DEFINITION

An atomic formula in predicate logic is a formula without propositional connectives or quantifiers.



$$
\begin{aligned}
&(\boxed{x = 1} \vee \boxed{x = 3}) \wedge \\
&(\boxed{y = 1} \vee \boxed{y = 2} \vee \boxed{y = 3}) \wedge \\
&(\boxed{z = 3}) \wedge \\
&\neg(\boxed{x = y}) \wedge \\
&\neg(\boxed{x = z}) \wedge \\
&\neg(\boxed{y = z})
\end{aligned}
$$

atom formulas

Lazy SMT Techniques

$(x = 1 \lor x = 3) \land$
$(y = 1 \lor y = 2 \lor y = 3) \land$
$(z = 3) \land$
$\neg(x = y) \land$
$\neg(x = z) \land$
$\neg(y = z)$

$\rightarrow$

$(p1 \lor p2) \land$
$(p3 \lor p4 \lor p5) \land$
$(p6) \land$
$\neg p7 \land$
$\neg p8 \land$
$\neg p9$

Step1: replace atomic formulas with fresh propositional variables.

Lazy SMT Techniques

$(x = 1 \vee x = 3) \wedge$

$(y = 1 \vee y = 2 \vee y = 3) \wedge$

$(z = 3) \wedge$

$\neg(x = y) \wedge$

$\neg(x = z) \wedge$

$\neg(y = z)$

$\longrightarrow$

$(p1 \vee p2) \wedge$

$(p3 \vee p4 \vee p5) \wedge$

$(p6) \wedge$

$\neg p7 \wedge$

$\neg p8 \wedge$

$\neg p9$

$\xrightarrow{\text{DPLL}}$ UNSAT

If it is unsatisfiable, the original formula is also unsatisfiable.

Step2: use SAT solvers to get assignments.

Lazy SMT Techniques

$(x = 1 \vee x = 3) \wedge$
$(y = 1 \vee y = 2 \vee y = 3) \wedge$
$(z = 3) \wedge$
$\neg(x = y) \wedge$
$\neg(x = z) \wedge$
$\neg(y = z)$

$(p1 \vee p2) \wedge$
$(p3 \vee p4 \vee p5) \wedge$
$(p6) \wedge$
$\neg p7 \wedge$
$\neg p8 \wedge$
$\neg p9$

DPLL

$p1 = T, p2 = F, p3 = T, p4 = F,$
$p5 = F, p6 = T, p7 = F, p8 = F,$
$p9 = F$

Step2: use SAT solvers to get assignments.

Lazy SMT Techniques

$(x = 1 \vee x = 3)\wedge$

$(y = 1 \vee y = 2 \vee y = 3)\wedge$

$(z = 3)\wedge$

$\neg(x = y)\wedge$

$\neg(x = z)\wedge$

$\neg(y = z)$

$(p1 \vee p2)\wedge$

$(p3 \vee p4 \vee p5)\wedge$

$(p6)\wedge$

$\neg p7\wedge$

$\neg p8\wedge$

$\neg p9$

DPLL

$p1 = T, p2 = F, p3 = T, p4 = F,$

$p5 = F, p6 = T, p7 = F, p8 = F,$

$p9 = F$

$x = 1, x \neq 3, y = 1, y \neq 2,$

$y \neq 3, z = 3, x \neq y, x \neq z,$

$y \neq z$

Step3: use T-solvers to check the satisfiability.

Lazy SMT Techniques

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$\neg(x = y) \wedge$$
$$\neg(x = z) \wedge$$
$$\neg(y = z)$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

DPLL

$$p1 = T, p2 = F, p3 = T, p4 = F,$$
$$p5 = F, p6 = T, p7 = F, p8 = F,$$
$$p9 = F$$

$$\boxed{x = 1}, x \neq 3, \boxed{y = 1}, y \neq 2,$$
$$y \neq 3, z = 3, \boxed{x \neq y}, x \neq z,$$
$$y \neq z$$

Step3: use T-solvers to check the satisfiability.

Lazy SMT Techniques

$$(x = 1 \vee x = 3)\wedge$$
$$(y = 1 \vee y = 2 \vee y = 3)\wedge$$
$$(z = 3)\wedge$$
$$\neg(x = y)\wedge$$
$$\neg(x = z)\wedge$$
$$\neg(y = z)$$

$$(p1 \vee p2)\wedge$$
$$(p3 \vee p4 \vee p5)\wedge$$
$$(p6)\wedge$$
$$\neg p7\wedge$$
$$\neg p8\wedge$$
$$\neg p9$$

DPLL

$$p1 = T, p2 = F, p3 = T, p4 = F,$$
$$p5 = F, p6 = T, p7 = F, p8 = F,$$
$$p9 = F$$

$$\boxed{x = 1}, x \neq 3, \boxed{y = 1}, y \neq 2,$$
$$y \neq 3, z = 3, \boxed{x \neq y}, x \neq z,$$
$$y \neq z$$

FALSE

If it is unsatisfiable, tell SAT solver to give another assignment.

Lazy SMT Techniques

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$\neg(x = y) \wedge$$
$$\neg(x = z) \wedge$$
$$\neg(y = z)$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

DPLL

$$p1 = T, p2 = F, p3 = T, p4 = F,$$
$$p5 = F, p6 = T, p7 = F, p8 = F,$$
$$p9 = F$$

$$\boxed{x = 1}, x \neq 3, \boxed{y = 1}, y \neq 2,$$
$$y \neq 3, z = 3, \boxed{x \neq y}, x \neq z,$$
$$y \neq z$$

FALSE

But how to tell the SAT solver this assignment doesn't work?

Lazy SMT Techniques

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$\neg(x = y) \wedge$$
$$\neg(x = z) \wedge$$
$$\neg(y = z)$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

DPLL

$$p1 = T, p2 = F, p3 = T, p4 = F,$$
$$p5 = F, p6 = T, p7 = F, p8 = F,$$
$$p9 = F$$

$$\boxed{x = 1}, x \neq 3, \boxed{y = 1}, y \neq 2,$$
$$y \neq 3, z = 3, \boxed{x \neq y}, x \neq z,$$
$$y \neq z$$

FALSE

$$\neg(p1 \wedge \neg p2 \wedge p3 \wedge \neg p4 \wedge$$
$$\neg p5 \wedge p6 \wedge \neg p7 \wedge \neg p8 \wedge$$
$$\neg p9)$$

It may be redundant.
Clever T-solvers can
give simpler formulas.

Lazy SMT Techniques

$(x = 1 \lor x = 3) \land$
$(y = 1 \lor y = 2 \lor y = 3) \land$
$(z = 3) \land$
$\neg(x = y) \land$
$\neg(x = z) \land$
$\neg(y = z)$

$\longrightarrow$

$(p1 \lor p2) \land$
$(p3 \lor p4 \lor p5) \land$
$(p6) \land$
$\neg p7 \land$
$\neg p8 \land$
$\neg p9$

$\xrightarrow{\text{DPLL}}$

$p1 = T, p2 = F, p3 = T, p4 = F,$
$p5 = F, p6 = T, p7 = F, p8 = F,$
$p9 = F$

$\boxed{x = 1}, x \neq 3, \boxed{y = 1}, y \neq 2,$
$y \neq 3, z = 3, \boxed{x \neq y}, x \neq z,$
$y \neq z$

FALSE

$\neg(p1 \land p3 \land \neg p7)$

It may be redundant.
Clever T-solvers can
give simpler formulas.

Lazy SMT Techniques

$(x = 1 \vee x = 3) \wedge$

$(y = 1 \vee y = 2 \vee y = 3) \wedge$

$(z = 3) \wedge$

$\neg(x = y) \wedge$

$\neg(x = z) \wedge$

$\neg(y = z)$

$(p1 \vee p2) \wedge$

$(p3 \vee p4 \vee p5) \wedge$

$(p6) \wedge$

$\neg p7 \wedge$

$\neg p8 \wedge$

$\neg p9$

DPLL

$p1 = T, p2 = F, \boxed{p3 = F}\ \boxed{p4 = T}$

$p5 = F, p6 = T, p7 = F, p8 = F,$

$p9 = F$

$x = 1, x \neq 3, y = 1, y \neq 2,$

$y \neq 3, z = 3, x \neq y, x \neq z,$

$y \neq z$

FALSE

$\neg(p1 \wedge p3 \wedge \neg p7)$

Lazy SMT Techniques

$$(x = 1 \lor x = 3) \land$$
$$(y = 1 \lor y = 2 \lor y = 3) \land$$
$$(z = 3) \land$$
$$\neg(x = y) \land$$
$$\neg(x = z) \land$$
$$\neg(y = z)$$

$$(p1 \lor p2) \land$$
$$(p3 \lor p4 \lor p5) \land$$
$$(p6) \land$$
$$\neg p7 \land$$
$$\neg p8 \land$$
$$\neg p9$$

DPLL

$$p1 = T, p2 = F, \boxed{p3 = F}\, \boxed{p4 = T}$$
$$p5 = F, p6 = T, p7 = F, p8 = F,$$
$$p9 = F$$

$$\neg(p1 \land p3 \land \neg p7)$$

$$x = 1, x \neq 3, \boxed{y \neq 1}\, \boxed{y = 2,}$$
$$y \neq 3, z = 3, x \neq y, x \neq z,$$
$$y \neq z$$

SAT

Lazy SMT Techniques

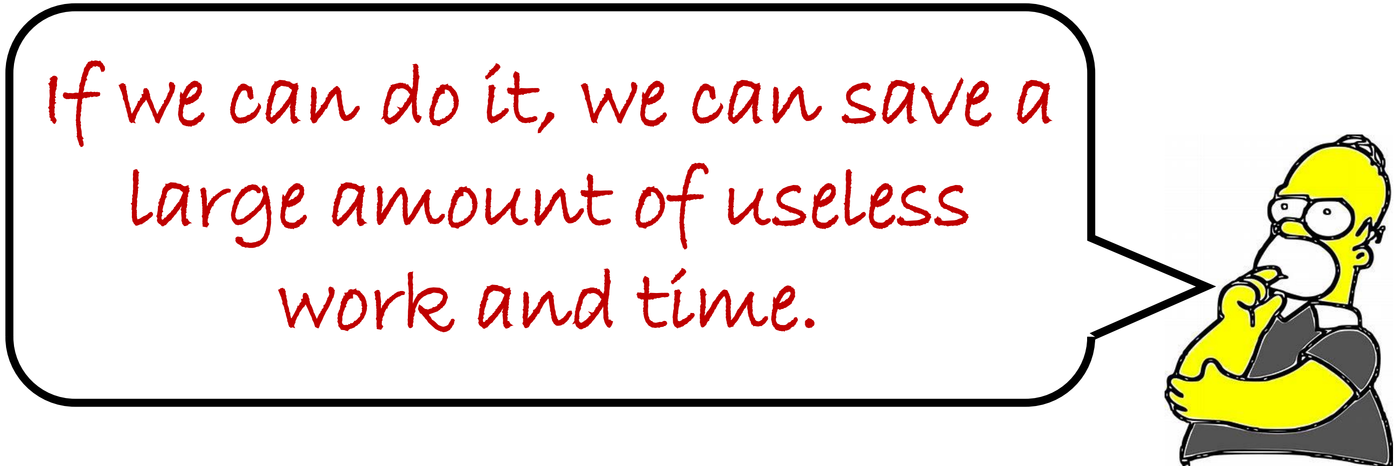
The main advantage of the lazy approach is its flexibility, since it can easily combine any SAT solver with any T-solver.

Several refinements exist that make the SMT procedure much more efficient when the SAT solver uses DPLL.

Incremental T-solver

In the original algorithm, T-solver checks unsatisfiability after DPLL finishes.

Can we discover errors earlier and notice SAT solver?



Incremental T-solver

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$\neg(x = y) \wedge$$
$$\neg(x = z) \wedge$$
$$\neg(y = z)$$

$$\Rightarrow$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

Incremental T-solver

$$
\begin{array}{lcl}
(x = 1 \lor x = 3) \land & & (p1 \lor p2) \land \\
(y = 1 \lor y = 2 \lor y = 3) \land & & (p3 \lor p4 \lor p5) \land \\
(z = 3) \land & \Rightarrow & (p6) \land \\
\neg(x = y) \land & & \neg p7 \land \\
\neg(x = z) \land & & \neg p8 \land \\
\neg(y = z) & & \neg p9
\end{array}
$$

$$\emptyset \parallel \quad 1 \lor 2,\ \ 3 \lor 4 \lor 5,\ \ 6,\ \ \overline{7},\ \ \overline{8},\ \ \overline{9} \quad \Longrightarrow (UnitProp)$$

Ignore pure literal rule here.

Incremental T-solver

$$
\begin{array}{lcl}
(x = 1 \lor x = 3)\land & & (p1 \lor p2)\land \\
(y = 1 \lor y = 2 \lor y = 3)\land & & (p3 \lor p4 \lor p5)\land \\
(z = 3)\land & & (p6)\land \\
\neg(x = y)\land & \Rightarrow & \neg p7\land \\
\neg(x = z)\land & & \neg p8\land \\
\neg(y = z) & & \neg p9
\end{array}
$$

$$\emptyset \parallel \; 1 \lor 2,\;\; 3 \lor 4 \lor 5,\;\; 6,\;\; \overline{7},\;\; \overline{8},\;\; \overline{9} \;\Longrightarrow (UnitProp)$$

T-solver $\Leftarrow$ $$6\;\; \overline{7}\;\; \overline{8}\;\; \overline{9} \parallel \; 1 \lor 2,\;\; 3 \lor 4 \lor 5,\;\; 6,\;\; \overline{7},\;\; \overline{8},\;\; \overline{9} \;\Longrightarrow (Decide)$$

✓

Incremental T-solver

$$
\begin{array}{ll}
(x = 1 \lor x = 3)\land & (p1 \lor p2)\land \\
(y = 1 \lor y = 2 \lor y = 3)\land & (p3 \lor p4 \lor p5)\land \\
(z = 3)\land & (p6)\land \\
\neg(x = y)\land & \neg p7\land \\
\neg(x = z)\land & \neg p8\land \\
\neg(y = z) & \neg p9
\end{array}
$$

$$
\begin{array}{rl}
\emptyset \;\| & 1 \lor 2,\;\; 3 \lor 4 \lor 5,\;\; 6,\;\; \overline{7},\;\; \overline{8},\;\; \overline{9} \implies (UnitProp) \\
6\;\; \overline{7}\;\; \overline{8}\;\; \overline{9} \;\| & 1 \lor 2,\;\; 3 \lor 4 \lor 5,\;\; 6,\;\; \overline{7},\;\; \overline{8},\;\; \overline{9} \implies (Decide) \\
6\;\; \overline{7}\;\; \overline{8}\;\; \overline{9}\;\; \overline{1}^{d} \;\| & 1 \lor 2,\;\; 3 \lor 4 \lor 5,\;\; 6,\;\; \overline{7},\;\; \overline{8},\;\; \overline{9} \implies (UnitProp)
\end{array}
$$

T-solver

Incremental T-solver

$(x = 1 \vee \boxed{x = 3})\wedge$ $\quad\quad$ $(p1 \vee p2)\wedge$

$(y = 1 \vee y = 2 \vee y = 3)\wedge$ $\quad\quad$ $(p3 \vee p4 \vee p5)\wedge$

$\boxed{(z = 3)}\wedge$ $\quad\quad$ $(p6)\wedge$

$\neg(x = y)\wedge$ $\quad\quad$ $\neg p7\wedge$

$\boxed{\neg(x = z)}\wedge$ $\quad\quad$ $\neg p8\wedge$

$\neg(y = z)$ $\quad\quad$ $\neg p9$

$$\emptyset \;\|\; 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (UnitProp)$$

$$6\; \overline{7}\; \overline{8}\; \overline{9} \;\|\; 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (Decide)$$

$$6\; \overline{7}\; \overline{8}\; \overline{9}\; \overline{1}^{d} \;\|\; 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (UnitProp)$$

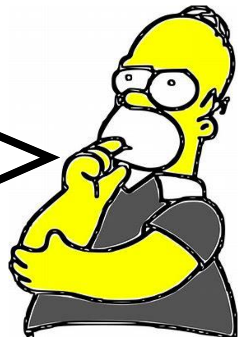
T-solver $\Leftarrow$ $\boxed{6}\; \overline{7}\; \boxed{\overline{8}}\; \overline{9}\; \overline{1}^{d}\; \boxed{2} \;\|\; 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9}$

Restart the SAT solver.

Incremental T-solver

It can be done fully eagerly, detecting unsatisfiability as soon as they are generated. But it may be much too expensive.

How to make it more efficient?



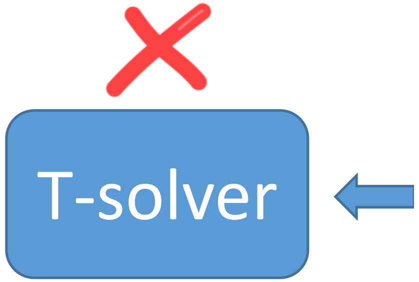
Incremental T-solver

It can be done fully eagerly, detecting unsatisfiability as soon as they are generated. But it may be much too expensive.

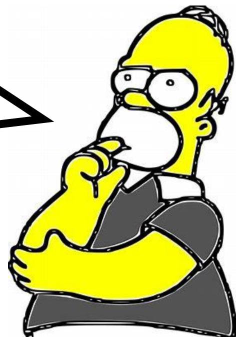
Detection can be done at regular intervals, for example, once every k literals added to the assignment.

Theory Propagation

$$\begin{aligned}
\emptyset \;\|\;& 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (UnitProp)\\
6\; \overline{7}\; \overline{8}\; \overline{9} \;\|\;& 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (Decide)\\
6\; \overline{7}\; \overline{8}\; \overline{9}\; \overline{1}^d \;\|\;& 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (UnitProp)\\
6\; \overline{7}\; \overline{8}\; \overline{9}\; \overline{1}^d\; 2 \;\|\;& 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (UnitProp)
\end{aligned}$$



T-solver just checks satisfiability. Can T-solver provide more information to guide SAT solver to produce assignment?



Theory Propagation

$$
\begin{aligned}
&(x = 1 \vee x = 3)\wedge && (p1 \vee p2)\wedge \\
&(y = 1 \vee y = 2 \vee y = 3)\wedge && (p3 \vee p4 \vee p5)\wedge \\
&(z = 3)\wedge && (p6)\wedge \\
&\neg(x = y)\wedge \quad\Rightarrow && \neg p7\wedge \\
&\neg(x = z)\wedge && \neg p8\wedge \\
&\neg(y = z) && \neg p9
\end{aligned}
$$

$$\emptyset \parallel \ 1 \vee 2,\ \ 3 \vee 4 \vee 5,\ \ 6,\ \ \overline{7},\ \ \overline{8},\ \ \overline{9} \implies (UnitProp)$$

T-solver

$$6\ \overline{7}\ \overline{8}\ \overline{9} \parallel \ 1 \vee 2,\ \ 3 \vee 4 \vee 5,\ \ 6,\ \ \overline{7},\ \ \overline{8},\ \ \overline{9}$$

Theory Propagation

$$x \neq 3$$
$$y \neq 3$$

$$\begin{array}{ll}
(x = 1 \vee x = 3) \wedge & (p1 \vee p2) \wedge \\
(y = 1 \vee y = 2 \vee y = 3) \wedge & (p3 \vee p4 \vee p5) \wedge \\
\boxed{\begin{array}{l} (z = 3) \wedge \\ \neg(x = y) \wedge \\ \neg(x = z) \wedge \\ \neg(y = z) \end{array}} \quad \Rightarrow & \begin{array}{l} (p6) \wedge \\ \neg p7 \wedge \\ \neg p8 \wedge \\ \neg p9 \end{array}
\end{array}$$

$$\emptyset \parallel \; 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9} \implies (UnitProp)$$

$$6\ \overline{7}\ \overline{8}\ \overline{9} \parallel \; 1 \vee 2,\; 3 \vee 4 \vee 5,\; 6,\; \overline{7},\; \overline{8},\; \overline{9}$$

T-solver

Theory Propagation

$x \neq 3$

$y \neq 3$

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$\boxed{\begin{array}{l}(z = 3) \wedge \\ \neg(x = y) \wedge \\ \neg(x = z) \wedge \\ \neg(y = z)\end{array}}$$

$$\Rightarrow$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

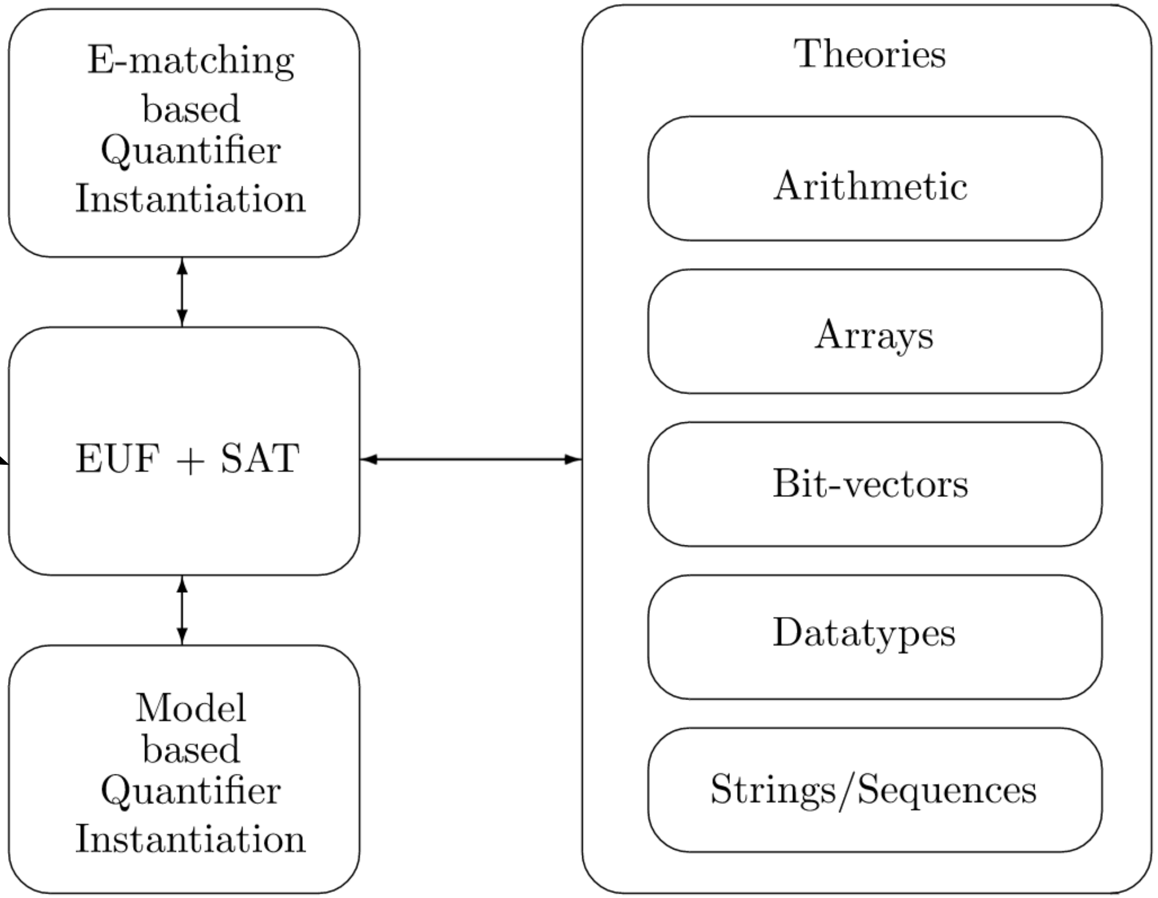
$$\emptyset \parallel \quad 1 \vee 2, \ \ 3 \vee 4 \vee 5, \ \ 6, \ \ \bar{7}, \ \ \bar{8}, \ \ \bar{9} \quad \Longrightarrow (UnitProp)$$

T-solver

$$6 \ \ \bar{7} \ \ \bar{8} \ \ \bar{9} \parallel \quad 1 \vee 2, \ \ 3 \vee 4 \vee 5, \ \ 6, \ \ \bar{7}, \ \ \bar{8}, \ \ \bar{9}$$

$$6 \ \ \bar{7} \ \ \bar{8} \ \ \bar{9} \ \ \boxed{\bar{2} \ \ \bar{5}} \mid \quad 1 \vee 2, \ \ 3 \vee 4 \vee 5, \ \ 6, \ \ \bar{7}, \ \ \bar{8}, \ \ \bar{9}$$

EUF is a T-Solver.



In addition to EUF, there are many other theory solvers.

Architecture of Z3's SMT Core solver

Theory Solver

What does T-Solver looks Like?

EUF

The logic of **equality and uninterpreted function**, EUF, is a basic ingredient for (first-order) predicate logic.

$$a = b, b = c, d = e, b = s, d = t$$

How to check if it is satisfiable?

EUF

While formulas are not empty:

remove a=b;

merge(a, b);

$$a = b, b = c, d = e$$

$$b = s, d = t$$

$$\Downarrow$$

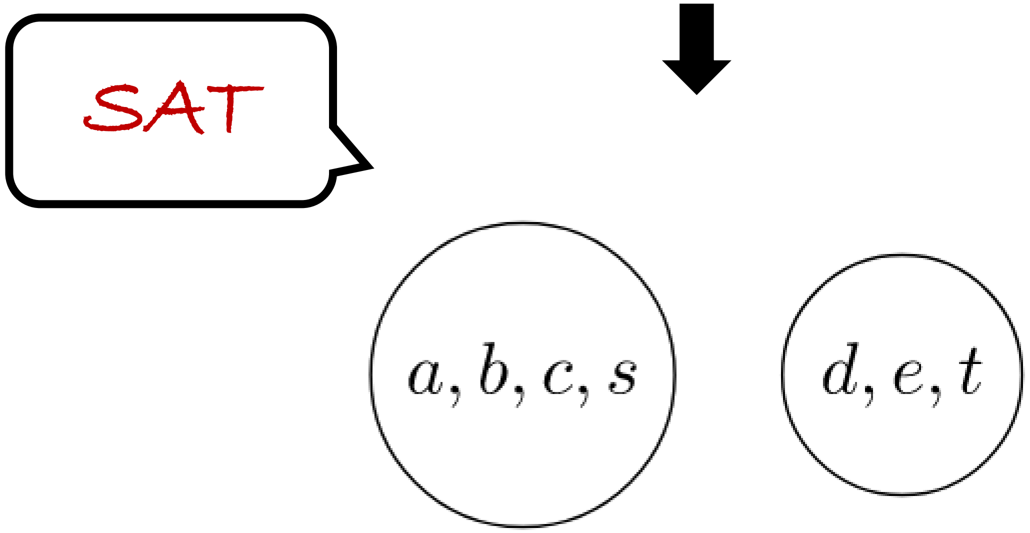
$$a, b, c, s \qquad d, e, t$$

EUF

```
While formulas are not empty:
        remove a=b;
        merge(a, b);
If find(a) = find(b) for some a≠b
        return false;
else
        return true;
```

$$a = b, b = c, d = e$$

$$b = s, d = t$$

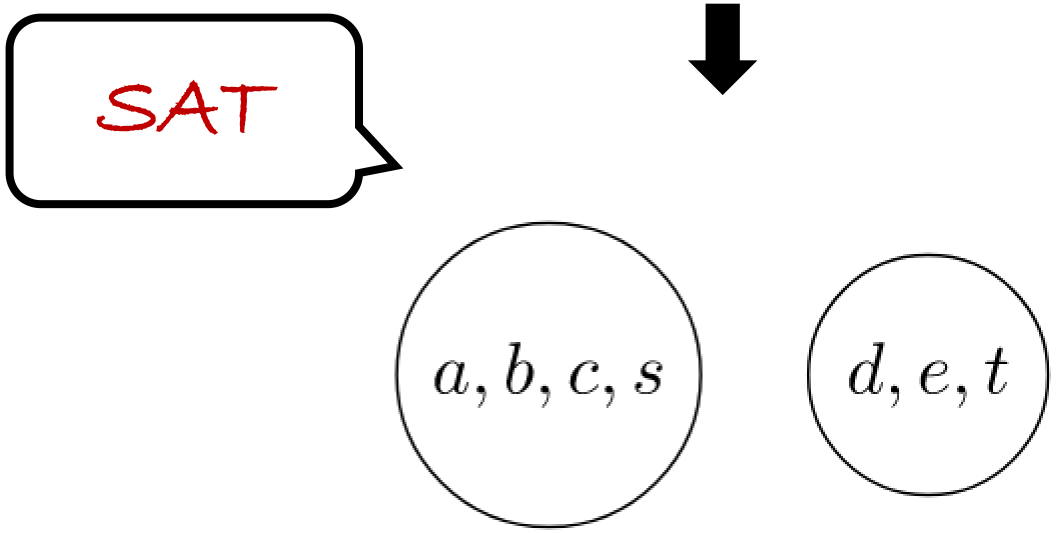


EUF

```
While formulas are not empty:
        remove a=b;
        merge(a, b);
If find(a) = find(b) for some a≠b
        return false;
else
        return true;
```

$$a = b, b = c, d = e$$

$$b = s, d = t, a \neq d$$



EUF

While formulas are not empty:

  remove a=b;

  merge(a, b);

If find(a) = find(b) for some a≠b

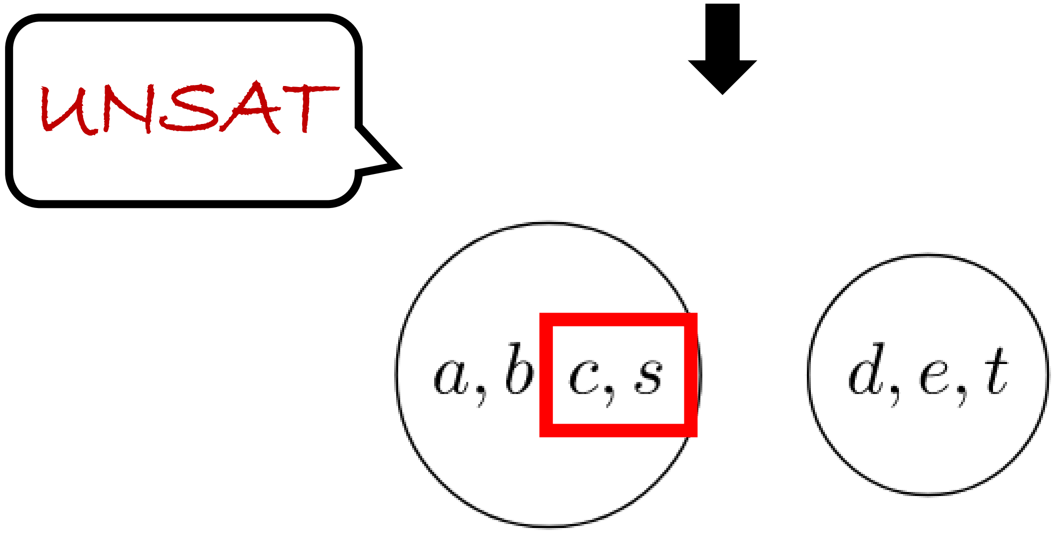
  return false;

else

  return true;

$$a = b, b = c, d = e$$

$$b = s, d = t, \boxed{c \neq s}$$

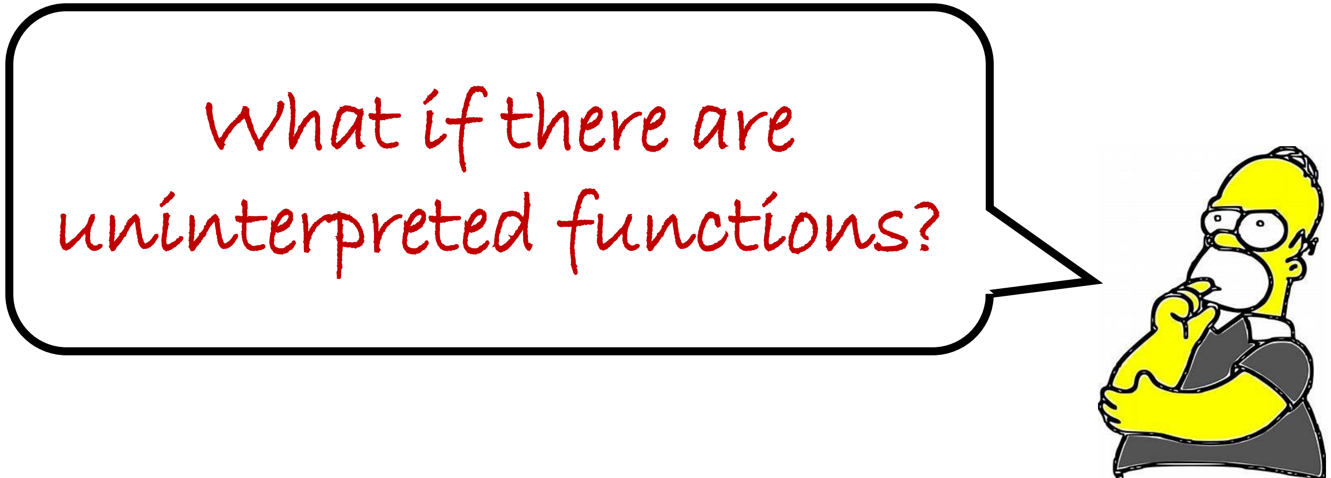


EUF

$$a = b, b = c, d = e$$

$$b = s, d = t$$

$$f(a, g(d)) \neq f(b, g(e))$$



EUF

THEOREM

Congruence rule:

$$x_1 = y_1, \ldots, x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

$$a = b, b = c, d = e$$

$$b = s, d = t$$

$$f(a, g(d)) \neq f(b, g(e))$$

We can construct the graph with congruence rule.

EUF

1) introduce constants that can be used as shorthands for sub-terms.

$$a = b, b = c, d = e$$
$$b = s, d = t$$
$$f(a, g(d)) \neq f(b, g(e))$$

$$\Rightarrow$$

$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$

$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

EUF

2) Working bottom-up, the congruence rule dictates how to merge groups

$$a = b, b = c, d = e$$

$$b = s, d = t, v3 \neq v4$$

$$v1 := g(e), v2 := g(d)$$

$$v3 := f(a, v2), v4 := f(b, v1)$$

a, b, c, s d, e, t v_1 v_2 v_3 v_4

EUF

2) Working bottom-up, the congruence rule dictates how to merge groups

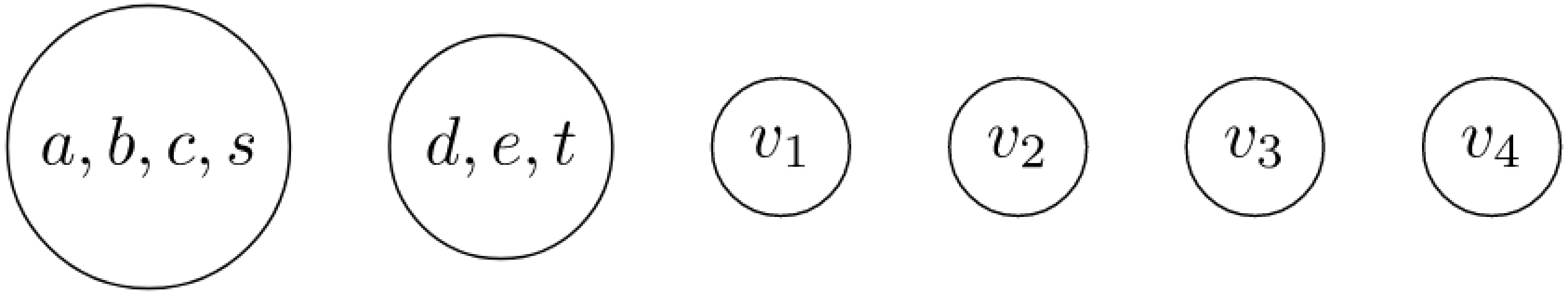
$$a = b, b = c, d = e$$

$$b = s, d = t, v3 \neq v4$$

$$v1 := g(e), v2 := g(d)$$

$$v3 := f(a, v2), v4 := f(b, v1)$$

$$e = d \Rightarrow g(e) = g(d)$$

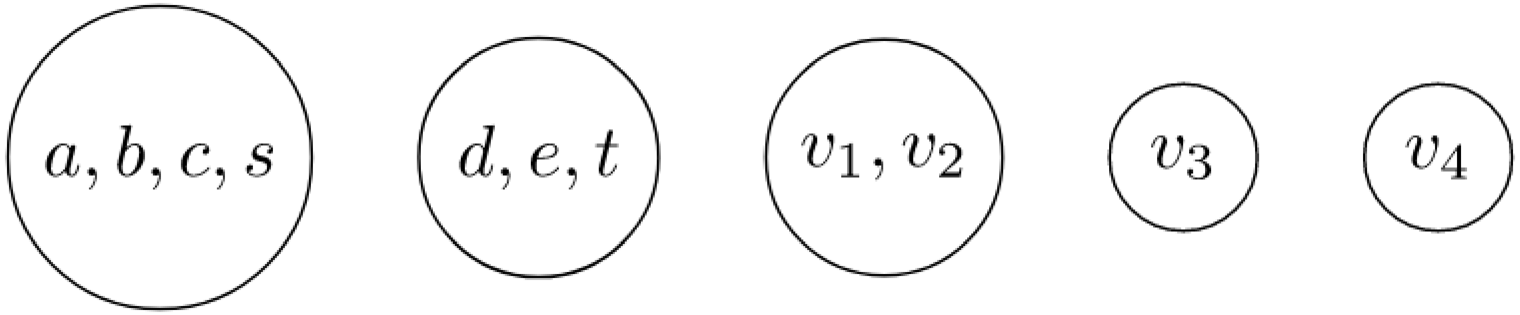


EUF

2) Working bottom-up, the congruence rule dictates how to merge groups

$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$
$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

$$e = d \Rightarrow g(e) = g(d)$$

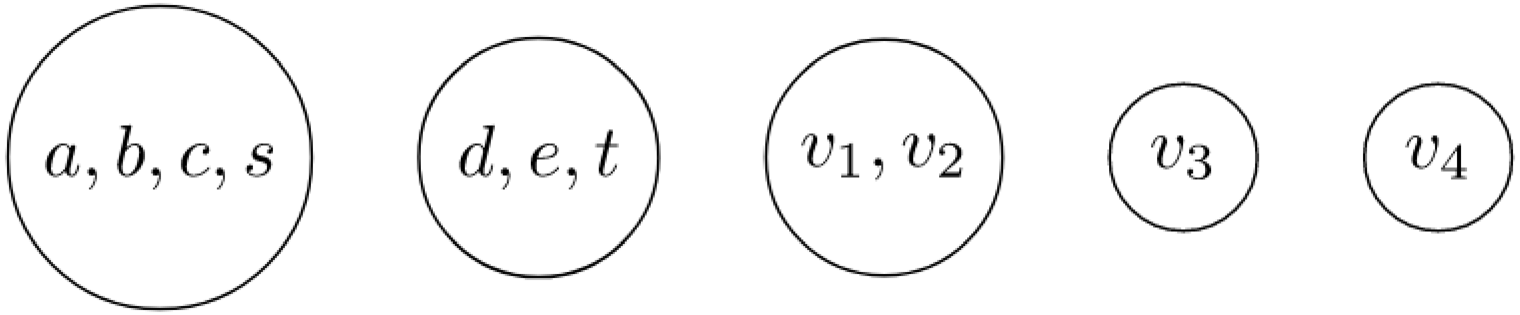


EUF

2) Working bottom-up, the congruence rule dictates how to merge groups

$$a = b, v2 = v1 \Rightarrow$$
$$f(a, v2) = f(b, v1)$$

$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$
$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$



EUF

2) Working bottom-up, the congruence rule dictates how to merge groups

$$a = b, v2 = v1 \Rightarrow$$
$$f(a, v2) = f(b, v1)$$

$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$
$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

a, b, c, s d, e, t v_1, v_2 v_3, v_4

EUF

3) Check satisfiability like before

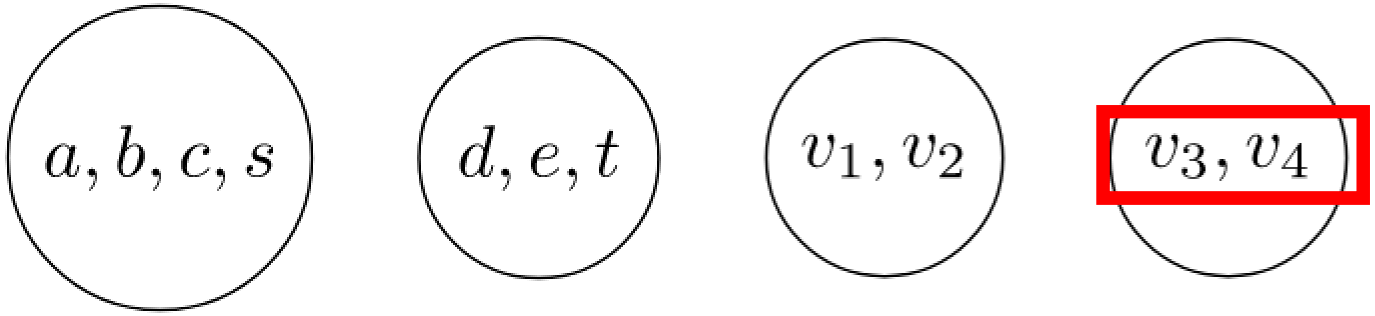
$$a = b, b = c, d = e$$

$$b = s, d = t, \boxed{v3 \neq v4}$$

$$v1 := g(e), v2 := g(d)$$

$$v3 := f(a, v2), v4 := f(b, v1)$$

UNSAT



Z3

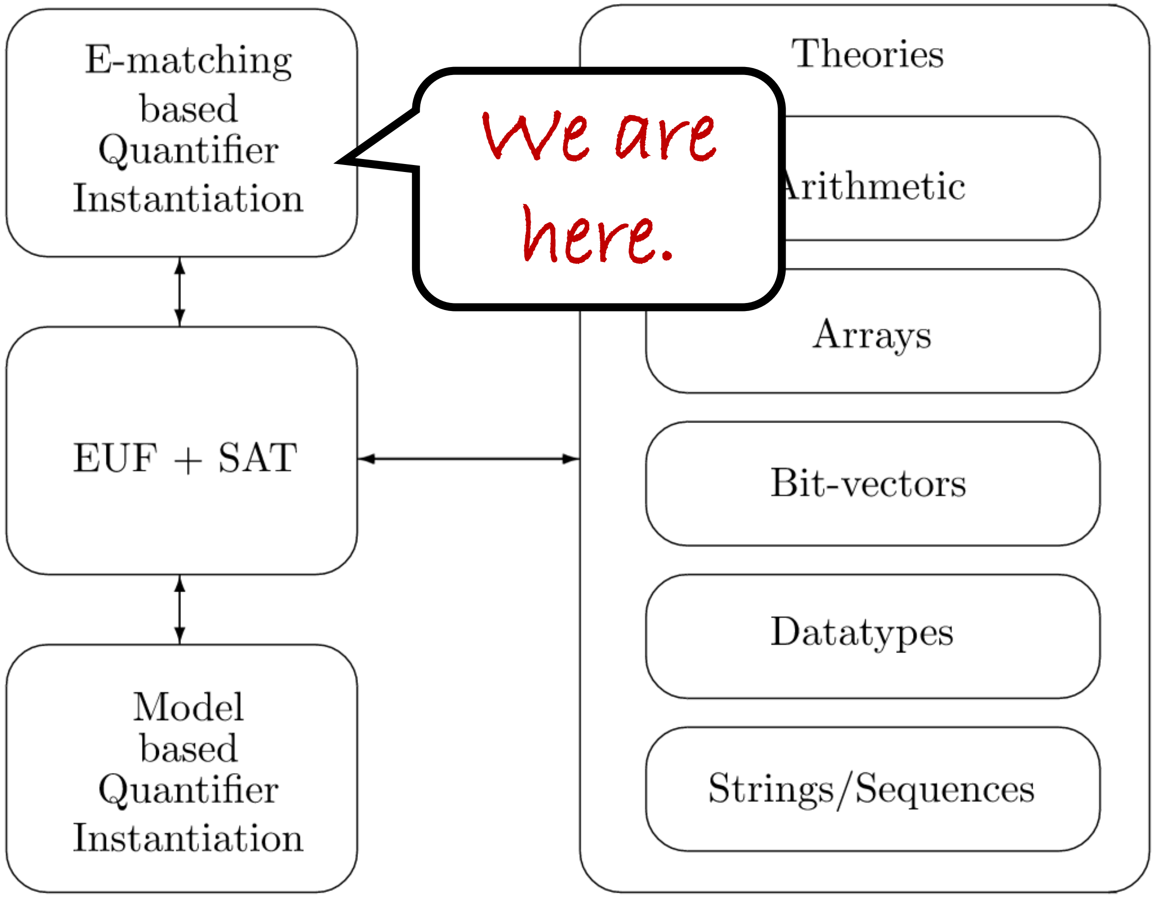
$$(x = 1 \lor x = 3) \land$$
$$(y = 1 \lor y = 2 \lor y = 3) \land$$
$$(z = 3) \land$$
$$(x \neq y) \land$$
$$(x \neq z) \land$$
$$(y \neq z)$$

individuals

Predicate logic formulas

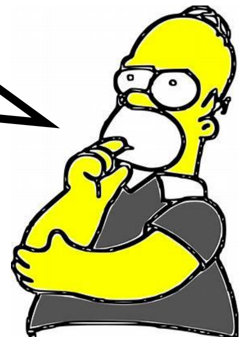
Solve it and return sat

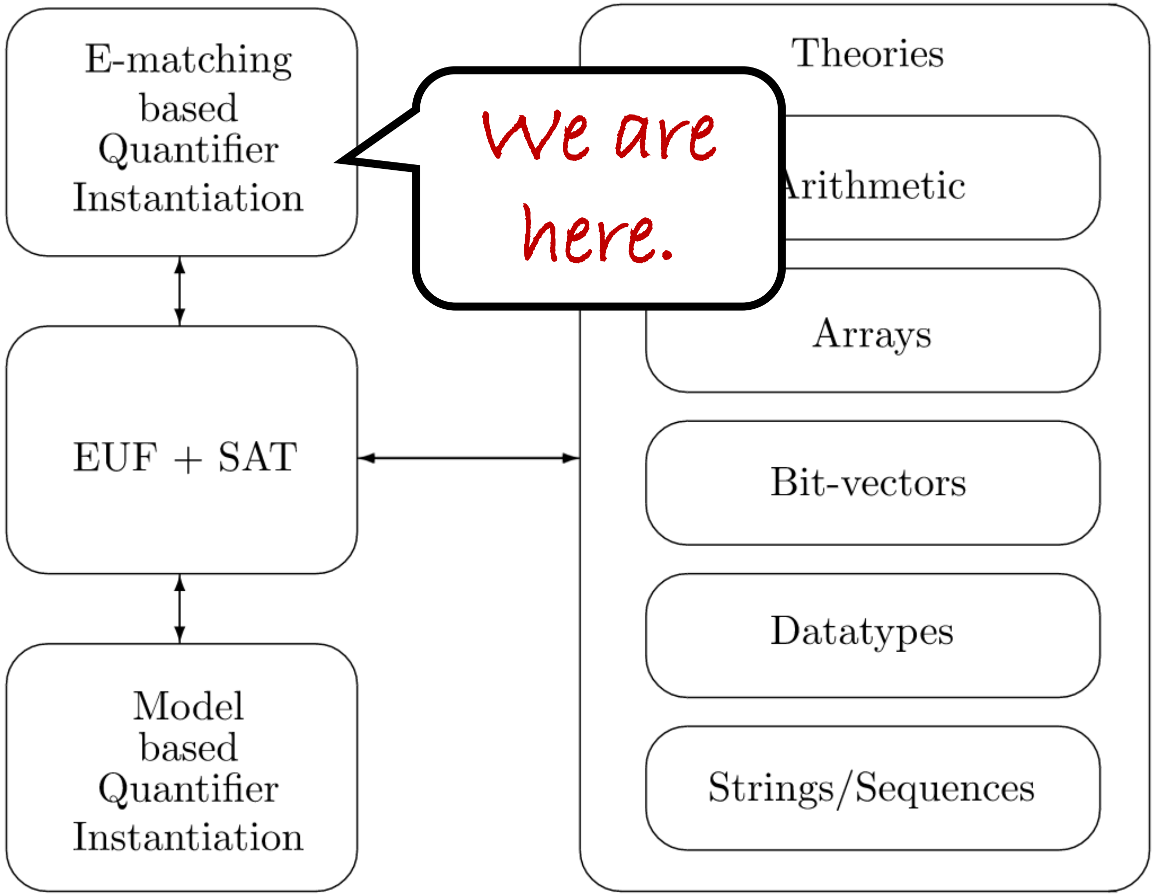
```
context ctx;
expr x = ctx.int_const("x");
expr y = ctx.int_const("y");
expr z = ctx.int_const("z");

solver s(ctx);
s.add(x==1 || x==3);
s.add(y==1 || y==2 || y==3);
s.add(z==3);
s.add(x != y);
s.add(x != z);
s.add(y != z);

if(s.check() == sat) cout << "sat";
else cout << "unsat";
```



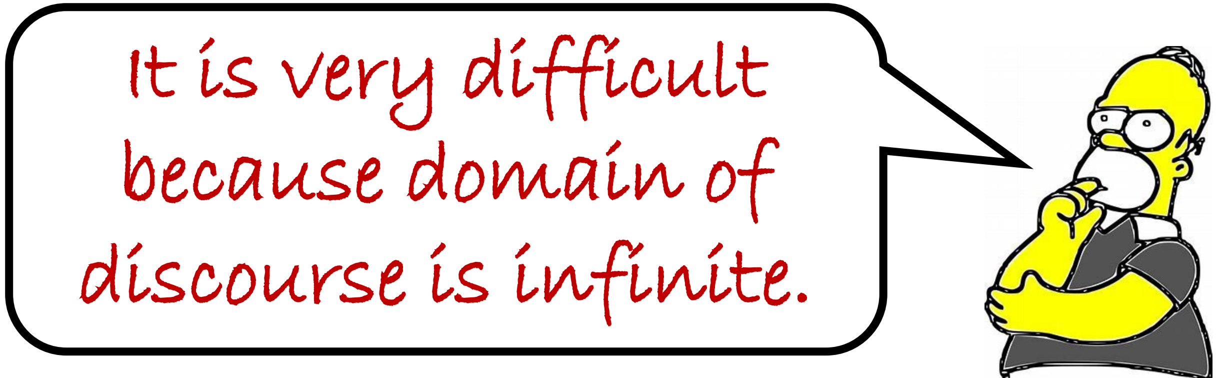
Architecture of Z3's SMT Core solver

We have been ignoring quantifiers so far.



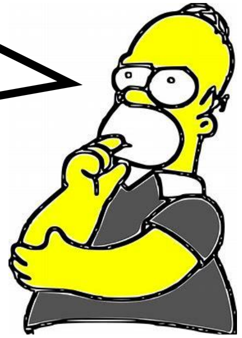


Architecture of Z3's SMT Core solver



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

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

$$\Downarrow$$

y depends on x

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

$$\Downarrow$$

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

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

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

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a

$$g(f(a)) = 0 \land \boxed{(\forall x)(g(f(x)) = 1)}$$

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

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a

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

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

Theory Solvers

UNSAT

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a.

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

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

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.

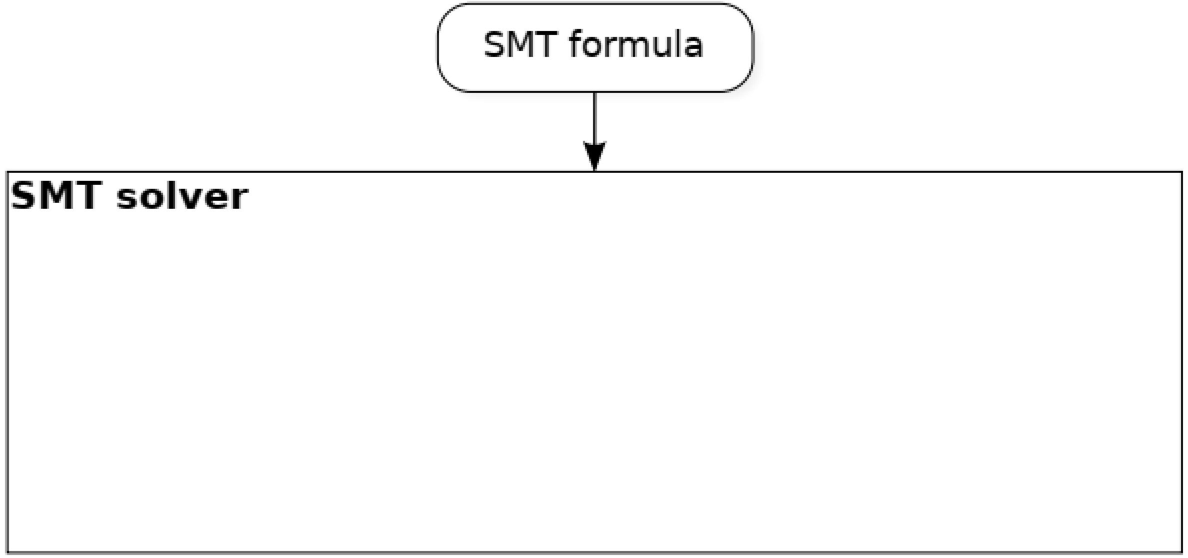
DEFINITION

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

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



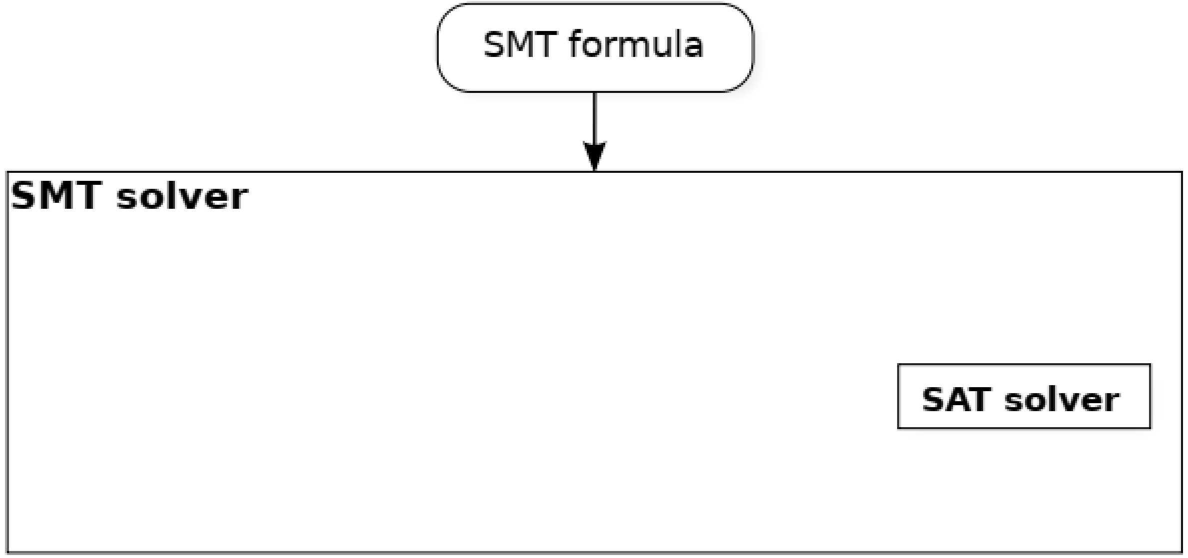
Quantifier

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

$$\Downarrow$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



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

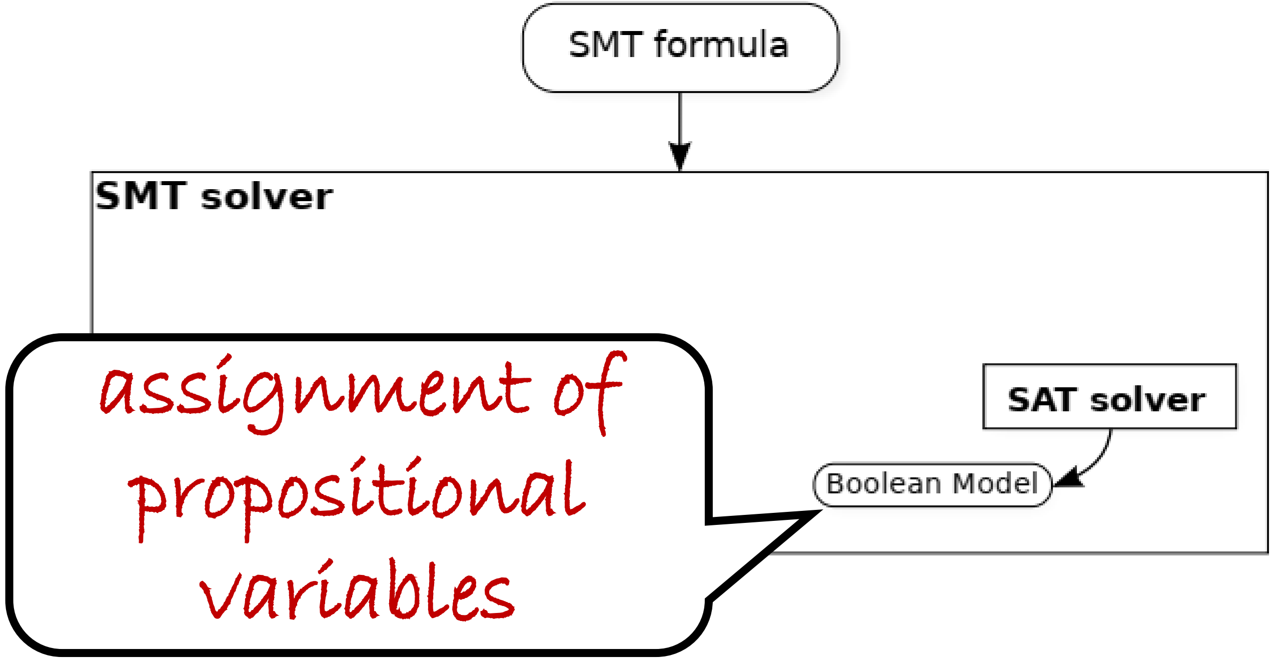
Quantifier

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

$$\Downarrow$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



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

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

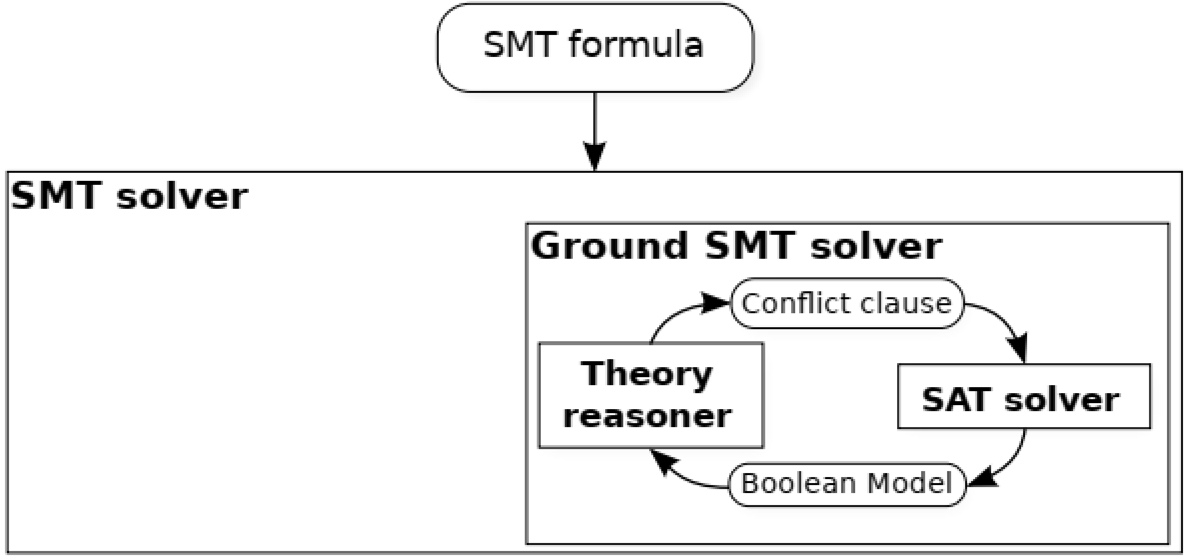
Quantifier

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

$$\Downarrow$$

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



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

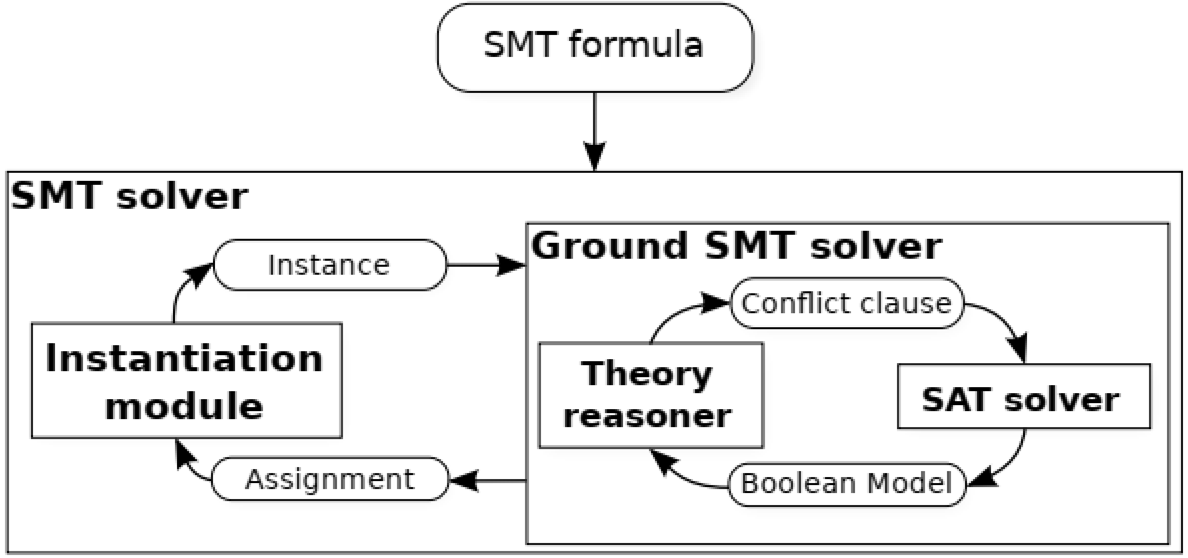
$$l_1 \;\; l_2 \;\; l_3 \parallel \; l_1,\; l_2,\; l_3$$

Quantifier

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

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



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

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

$$l_1 \;\; l_2 \;\; l_3 \parallel \quad l_1, \quad l_2, \quad l_3, \; \boxed{\neg l_2 \vee b \not\geq 0 \vee f(b) = a}$$

preserve the satisfiability of formula

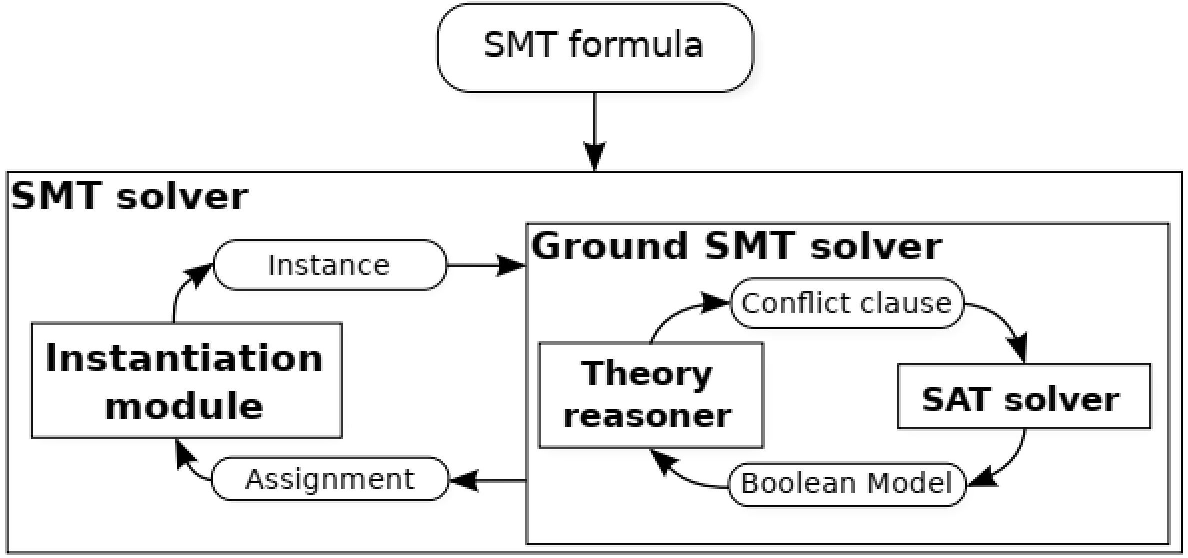
Quantifier

$$\boxed{0 \le b} \quad \land \quad \forall x.\,(x \not\geq 0 \lor f(x) = a) \quad \land \quad f(b) \neq a$$

$$\Downarrow$$

$$l_1 \land l_2 \land l_3$$

Abstract CNF



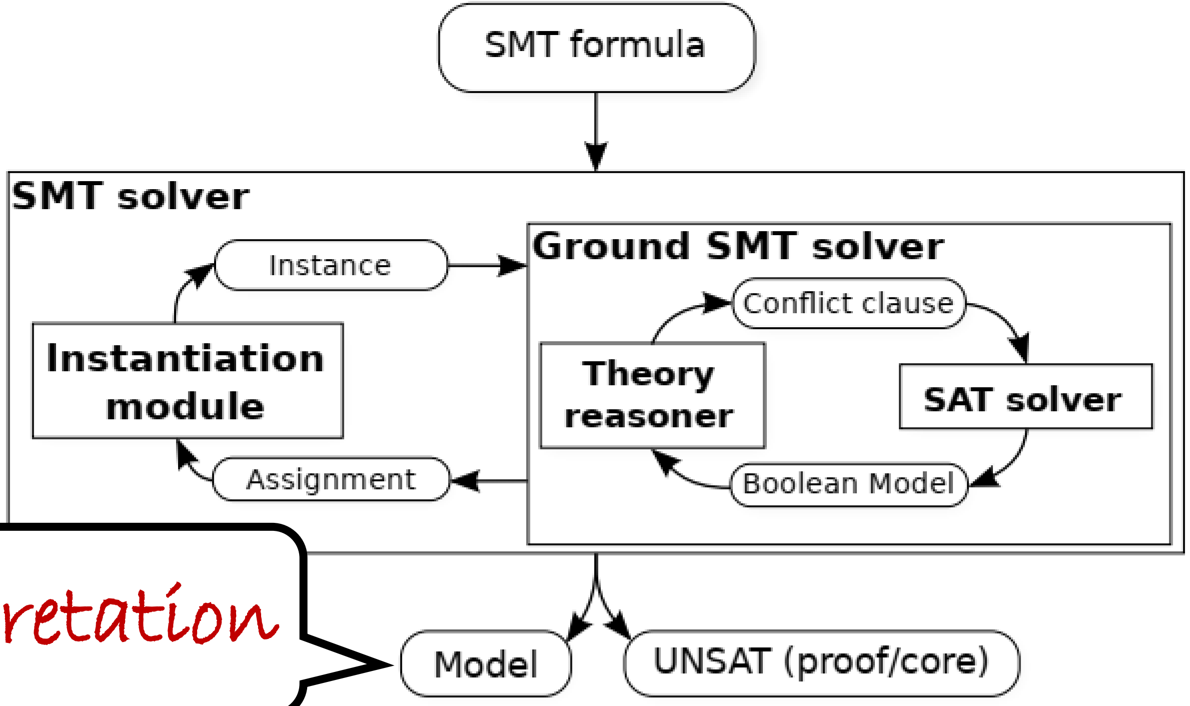
$$\begin{array}{rll}
\emptyset \parallel & l_1,\ l_2,\ l_3 & \Longrightarrow (UnitProp) \\
l_1\ \ l_2\ \ l_3 \parallel & l_1,\ l_2,\ l_3 & \Longrightarrow (\forall - Inst) \\
l_1\ \ l_2\ \ l_3 \parallel & l_1,\ l_2,\ l_3,\ \neg l_2 \lor \boxed{b \not\geq 0} \lor f(b) = a & \Longrightarrow (T - Prop)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

Quantifier

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

$$l_1 \wedge l_2 \wedge l_3$$

Abstract CNF



interpretation

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

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

$$l_1 \; l_2 \; l_3 \parallel \; l_1, \; l_2, \; l_3, \; \neg l_2 \vee b \not\geq 0 \vee f(b) = a \qquad \Longrightarrow (T - Prop)$$

$$l_1 \; \boxed{l_2 \; l_3 \; b \geq 0} \parallel \; l_1, \; l_2, \; l_3, \; \boxed{\neg l_2} \vee \boxed{b \not\geq 0} \vee \boxed{f(b) = a} \qquad \Longrightarrow (Fail)$$

$$fail$$

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.

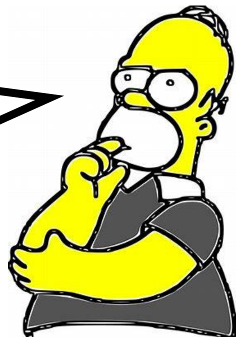
$$(\forall x.\{f(x)\})(f(x) > 0)$$

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

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

$$\Downarrow$$

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

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.

Thanks & Questions