Predicate Logic

Zhaoguo Wang

Adapted From:

NYU G22.2390-001, Lecture 8, 9
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 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

A Quick Recap – Predicate Logic

- Basic Concept
- Deduction

Outline – This Lecture

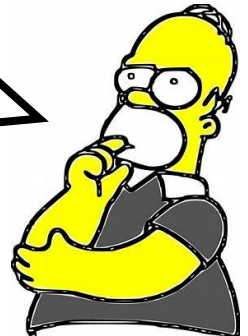
- SMT Solver
  - SMT
  - From DPLL to DPLL(T)
  - EUF
  - The Nelson-Oppen Method

SMT

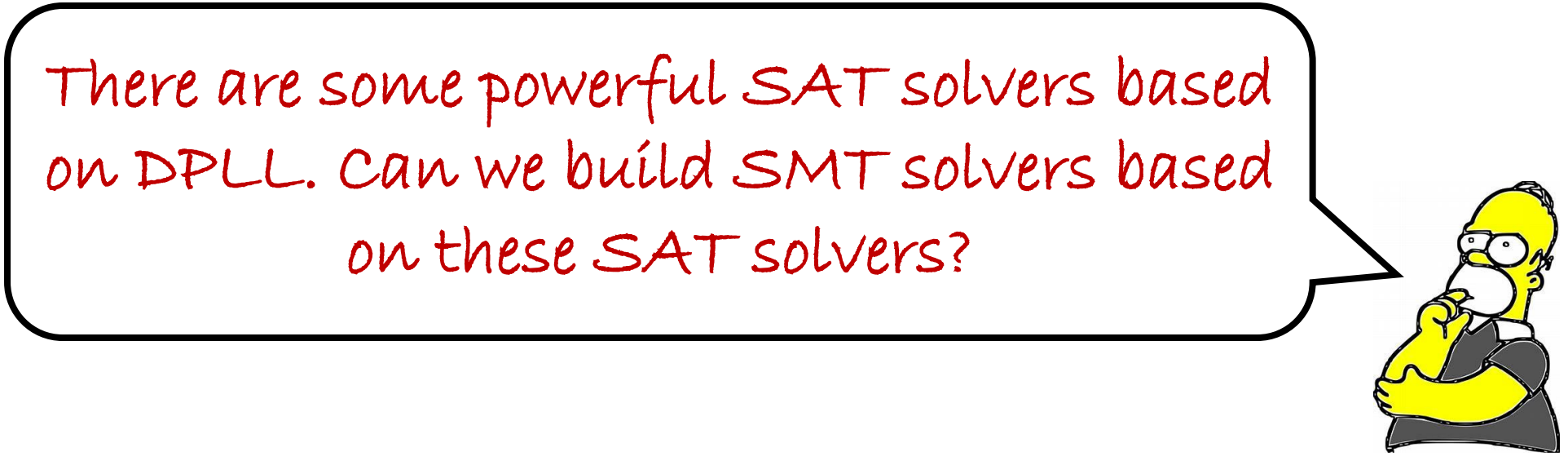
DEFINITION

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?



SMT Solvers can do it!

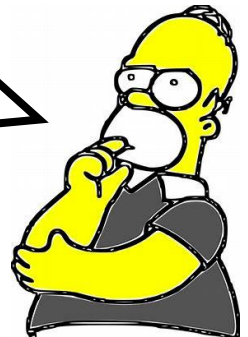


First, we consider the case of *quantifier-free formulas*.
In the textbook, interpretation doesn't include constants.
But in SMT solvers, interpretation includes them.

A Quick Recap

$$\emptyset \parallel \bar{1} \vee \bar{2},\quad 2 \vee 3,\quad \bar{1} \vee \bar{3} \vee 4,\quad 2 \vee \bar{3} \vee \bar{4},\quad 1 \vee 4$$

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



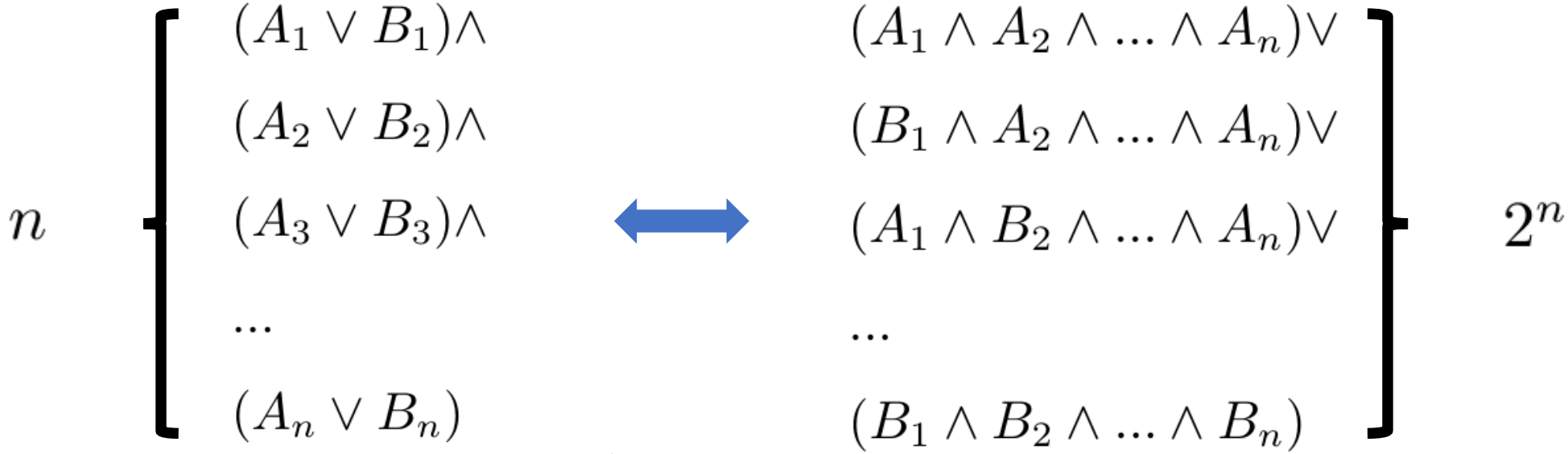
WHY CNF?

1) Convert the formula to DNF.

2) Build a specialized T-solver for deciding the satisfiability of conjunctions of literals.

3) Use the T-solver to check whether any of the DNF conjuncts is satisfiable.

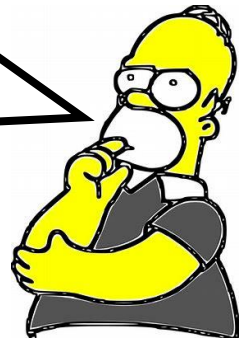
A Quick Recap



You can convert a formula into DNF, but the resulting formula might be very much larger than the original formula—in fact, exponentially so.

WHY CNF?

We may combine SAT solvers and T-solvers to determine satisfiability of a formula in CNF.



WHY CNF?

$$(A1 \wedge A2 \wedge A3) \vee (B1 \wedge B2 \wedge B3)$$

3X3

$$(A1 \vee B1) \wedge (A1 \vee B2) \wedge (A1 \vee B3) \wedge$$
$$(A2 \vee B1) \wedge (A2 \vee B2) \wedge (A2 \vee B3) \wedge$$
$$(A3 \vee B1) \wedge (A3 \vee B2) \wedge (A3 \vee B3)$$

It seems to exponentially explode, too.

Switch Variables

$$(A1 \wedge A2 \wedge A3) \vee (B1 \wedge B2 \wedge B3)$$

$$(Z \rightarrow A1 \wedge A2 \wedge A3) \wedge$$
$$(\neg Z \rightarrow B1 \wedge B2 \wedge B3)$$

The translation is equi-satisfiable.

The translation is not equivalent.

Exercise: convert it to CNF and check if there is explosion.

Switch Variables

$$(A1 \wedge A2 \wedge A3) \vee (B1 \wedge B2 \wedge B3)$$

The translation is equi-satisfiable.

The translation is not equivalent.

$$(Z \rightarrow A1 \wedge A2 \wedge A3) \wedge$$
$$(\neg Z \rightarrow B1 \wedge B2 \wedge B3)$$

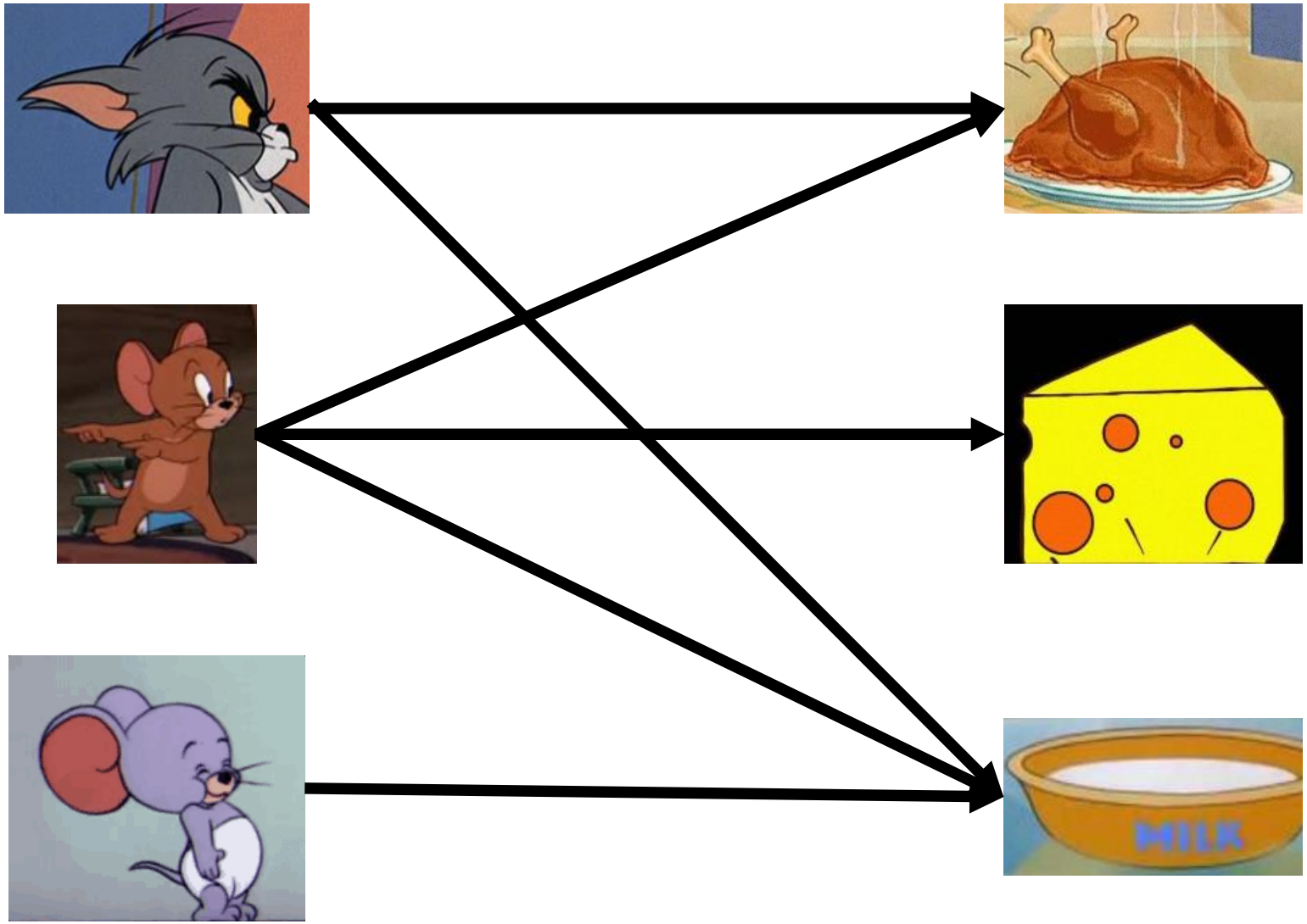
$$(Z \rightarrow (A1 \wedge A2 \wedge A3)) \wedge (\neg Z \rightarrow (B1 \wedge B2 \wedge B3))$$
$$= (\neg Z \vee (A1 \wedge A2 \wedge A3)) \wedge (Z \vee (B1 \wedge B2 \wedge B3))$$
$$= (\neg Z \vee A1) \wedge (\neg Z \vee A2) \wedge (\neg Z \vee A3) \wedge$$
$$(Z \vee B1) \wedge (Z \vee B2) \wedge (Z \vee B3)$$

3+3

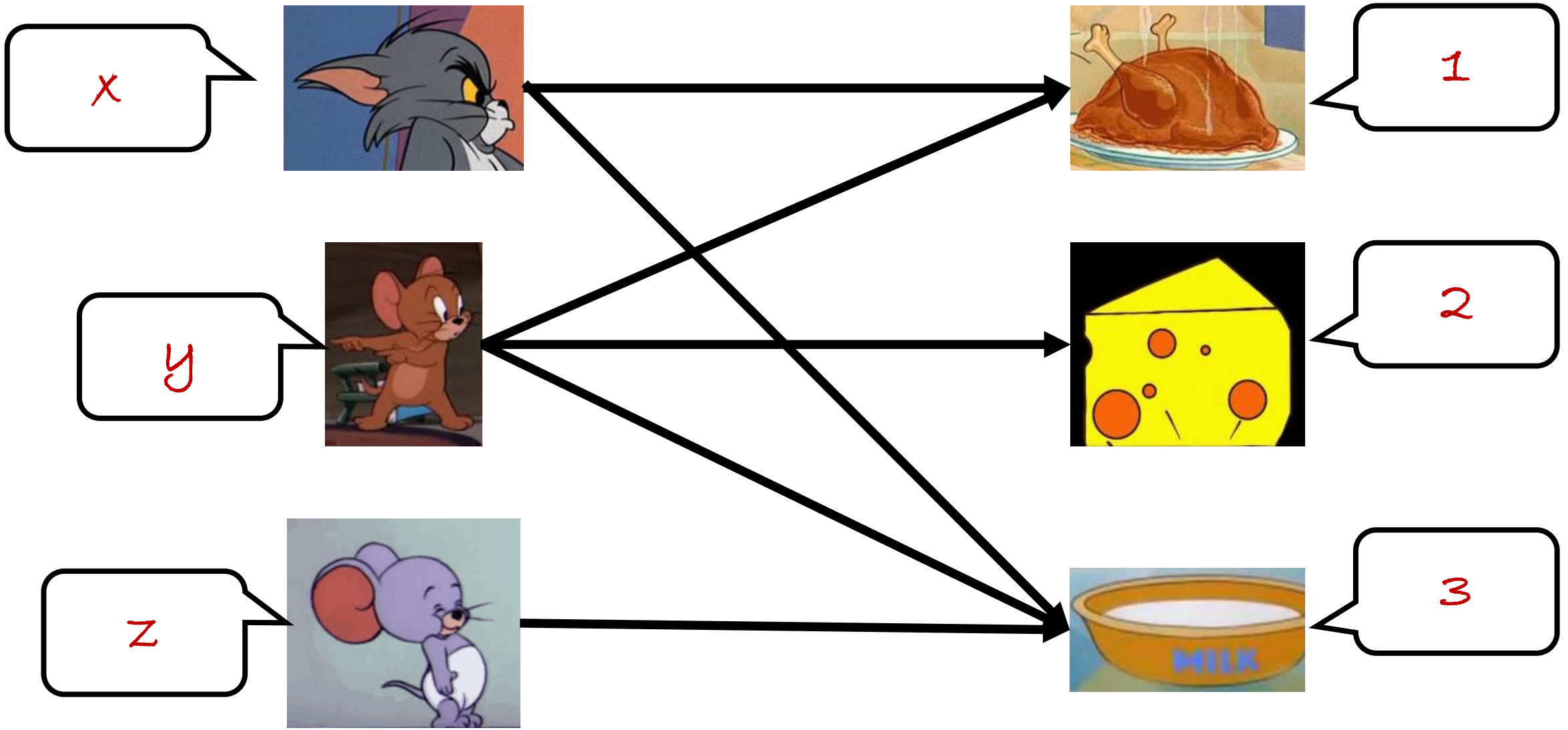
There is not explosion.

Eager SMT converts a theory into a SAT problem

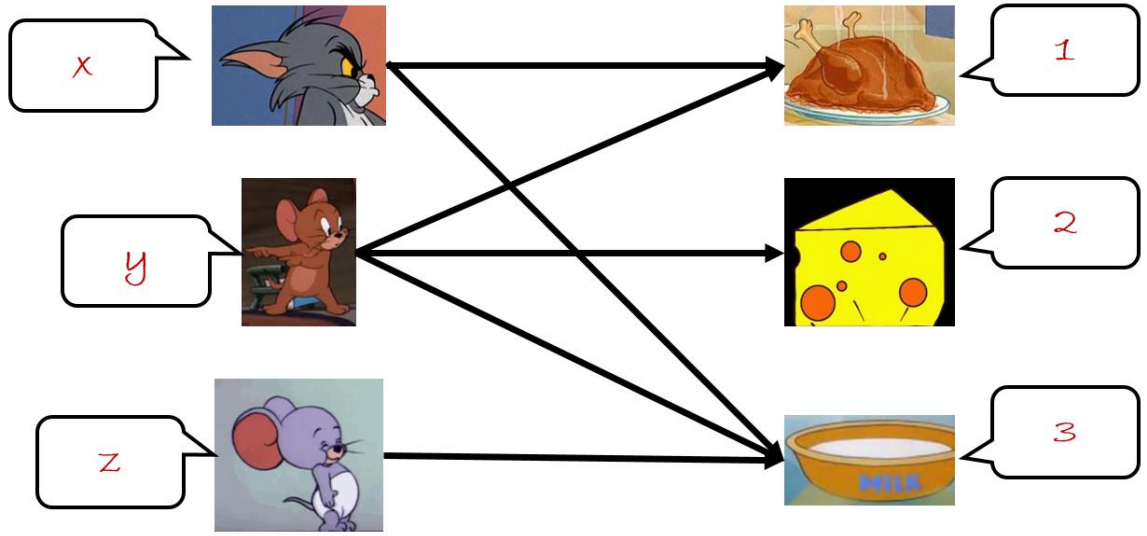
Eager SMT Techniques



Eager SMT Techniques



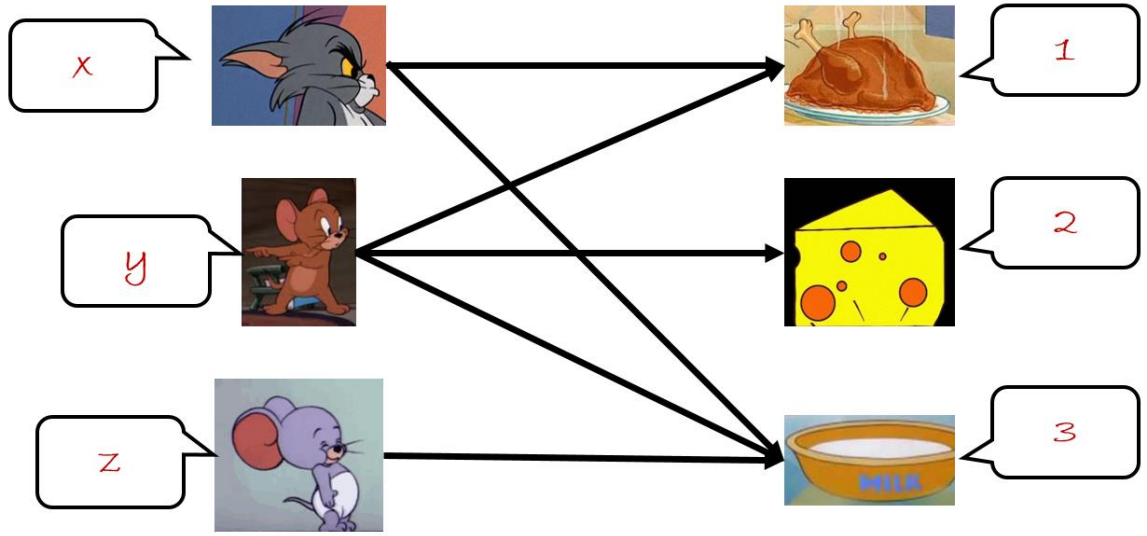
Eager SMT Techniques



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

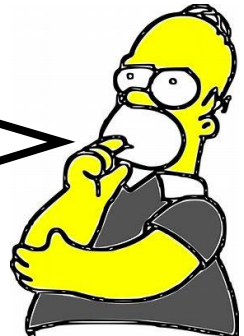
predicate logic

Eager SMT Techniques

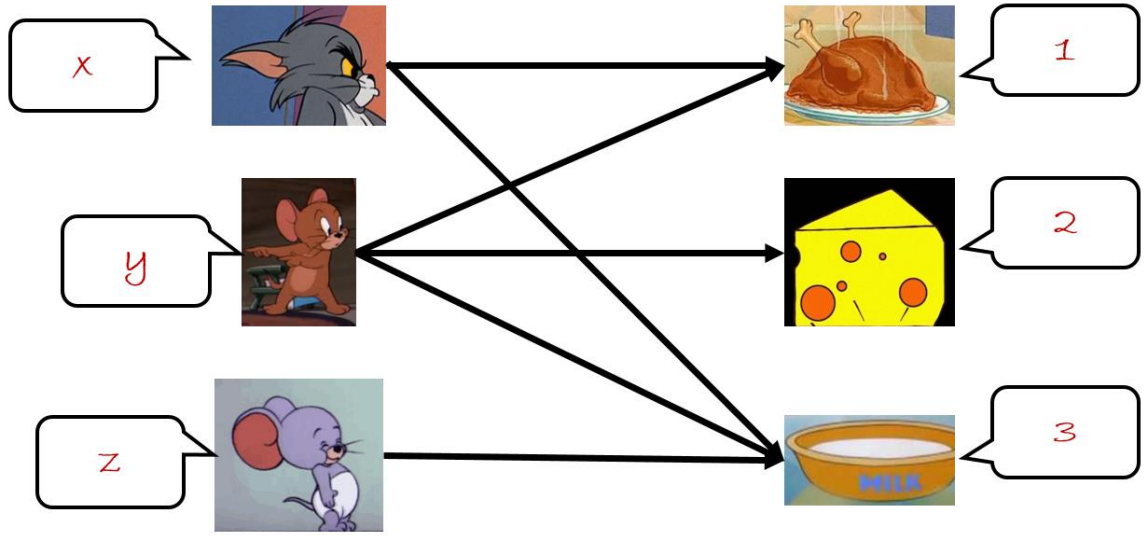


$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$(x \neq y) \wedge$$
$$(x \neq z) \wedge$$
$$(y \neq z)$$

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



Eager SMT Techniques

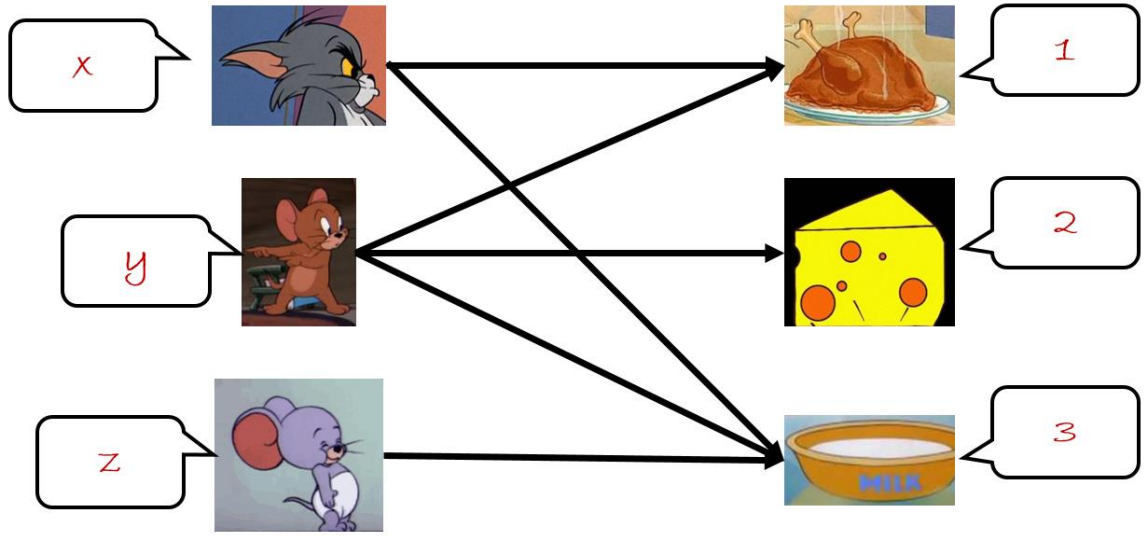


$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$(x \neq y) \wedge$$
$$(x \neq z) \wedge$$
$$(y \neq z)$$

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

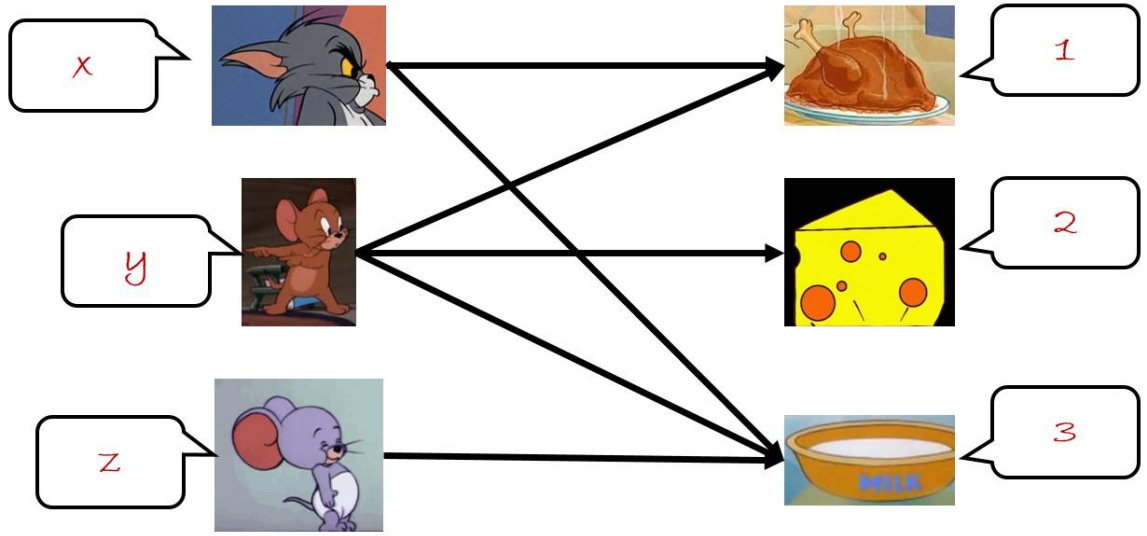


$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$(x \neq y) \wedge$$
$$(x \neq z) \wedge$$
$$(y \neq z)$$

$$((\neg p1 \wedge p0) \vee (p1 \wedge p0)) \wedge$$
$$((\neg q1 \wedge q0) \vee (q1 \wedge \neg q0) \vee (q1 \wedge q0)) \wedge$$
$$(r1 \wedge r0) \wedge$$
$$\neg((p1 \leftrightarrow q1) \wedge (p0 \leftrightarrow q0)) \wedge$$
$$\neg((q1 \leftrightarrow r1) \wedge (q0 \leftrightarrow r0)) \wedge$$
$$\neg((p1 \leftrightarrow r1) \wedge (p0 \leftrightarrow r0))$$

x	p1	p0
y	q1	q0
z	r1	r0

Eager SMT Techniques



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

$$((\neg p1 \wedge p0) \vee (p1 \wedge p0)) \wedge$$
$$((\neg q1 \wedge q0) \vee (q1 \wedge \neg q0) \vee (q1 \wedge q0)) \wedge$$
$$(r1 \wedge r0) \wedge$$
$$\neg((p1 \leftrightarrow q1) \wedge (p0 \leftrightarrow q0)) \wedge$$
$$\neg((q1 \leftrightarrow r1) \wedge (q0 \leftrightarrow r0)) \wedge$$
$$\neg((p1 \leftrightarrow r1) \wedge (p0 \leftrightarrow r0))$$

x → | p1 | p0 |

Leave the rest to powerful SAT solvers.

Eager SMT Techniques

Eager SMT techniques require to encode problems to SAT:

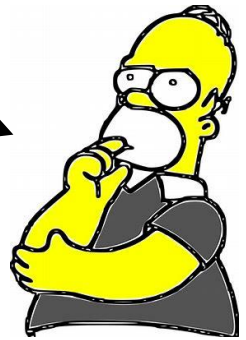
1) Represent the state space

2) Convert the problem into the new representation

3) Check the equisatisfiability

4) (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

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

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

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

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

UNSAT

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

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

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

Eager SMT Techniques

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

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

UNSAT

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

SAT

$$y = T, f_z = T, x = T$$
$$f_{fz} = T, f_y = F$$

Eager SMT Techniques

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

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

UNSAT

FALSE

SAT

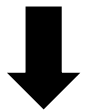
$$(y \leftrightarrow f_z) \wedge$$
$$(x \leftrightarrow f_{fz}) \wedge$$
$$\neg(x \leftrightarrow f_y)$$

$$y = T, f_z = T, x = T$$
$$f_{fz} = T, f_y = F$$

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

Eager SMT Techniques

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



$$((y \leftrightarrow z) \rightarrow (f_y \leftrightarrow f_z)) \wedge$$

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

$$((z \leftrightarrow f_z) \rightarrow (f_z \leftrightarrow f_{fz})) \wedge$$

Property of function f:

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

UNSAT

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

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

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

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.

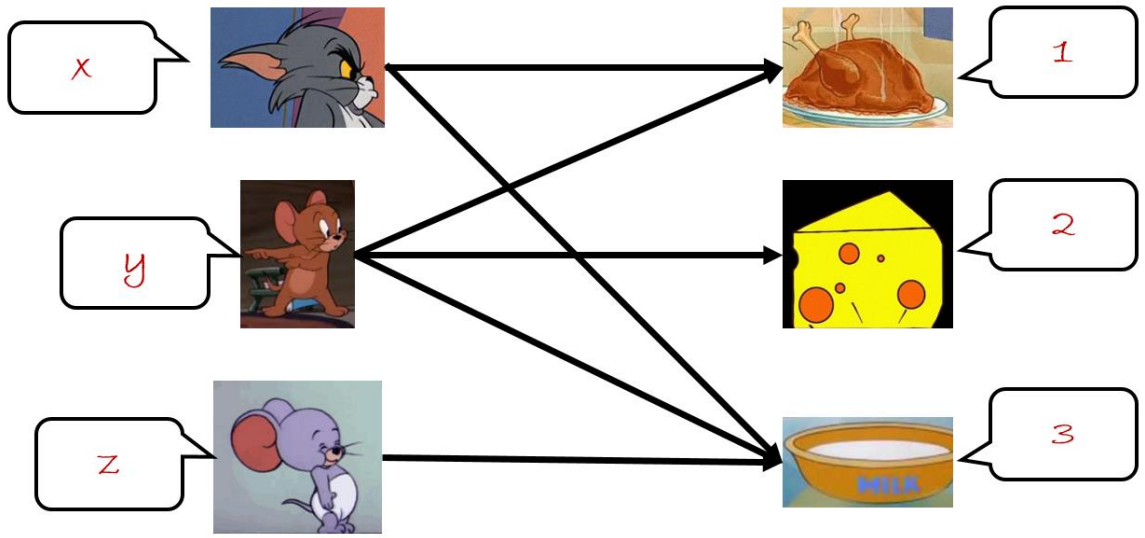
On many practical problems the translation process or the SAT solver run out of time or memory.

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.



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

atom formulas

Lazy SMT Techniques

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

Step1: replace atomic formulas with fresh propositional variables.

Lazy SMT Techniques

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

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

DPLL

UNSAT

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

Step2: use SAT solvers to get assignments.

Lazy SMT Techniques

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

DPLL →

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = T, p4 = F, \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

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

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

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

FALSE

Step3: use T-solvers to check the satisfiability.

Lazy SMT Techniques

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

DPLL

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = T, p4 = F, \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

$$
\begin{aligned}
&x = 1, x \neq 3, y = 1, y \neq 2, \\
&y \neq 3, z = 3, x \neq y, x \neq z, \\
&y \neq z
\end{aligned}
$$

FALSE

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

Lazy SMT Techniques

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

DPLL

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = T, p4 = F, \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

$$
\begin{aligned}
&x = 1, x \neq 3, y = 1, y \neq 2, \\
&y \neq 3, z = 3, x \neq y, x \neq z, \\
&y \neq z
\end{aligned}
$$

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

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

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

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

FALSE

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

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

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

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

FALSE

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

Lazy SMT Techniques

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

DPLL

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = F, p4 = T \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

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

$$
\begin{aligned}
&x = 1, x \neq 3, y = 1, y \neq 2, \\
&y \neq 3, z = 3, x \neq y, x \neq z, \\
&y \neq z
\end{aligned}
$$

FALSE

Lazy SMT Techniques

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

DPLL

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = F \; p4 = T \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

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

$$
\begin{aligned}
&x = 1, x \neq 3, y \neq 1 \; y = 2, \\
&y \neq 3, z = 3, x \neq y, x \neq z, \\
&y \neq z
\end{aligned}
$$

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.

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

$$\emptyset \parallel \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad\Longrightarrow\quad \text{(Decide)}$$

$$1^{\mathsf{d}} \parallel \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 \quad\Longrightarrow\quad \text{(UnitPropagate)}$$

A Quick Recap

$$\begin{aligned}
\emptyset &\parallel \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 &&\Longrightarrow \quad \text{(Decide)}\\
1^{\mathsf{d}} &\parallel \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 &&\Longrightarrow \quad \text{(UnitPropagate)}\\
1^{\mathsf{d}}\,\overline{2} &\parallel \overline{1}\vee\overline{2},\ \ 2\vee3,\ \ \overline{1}\vee\overline{3}\vee4,\ \ 2\vee\overline{3}\vee\overline{4},\ \ 1\vee4 &&\Longrightarrow \quad \text{(UnitPropagate)}
\end{aligned}$$

A Quick Recap

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

A Quick Recap

$$
\begin{aligned}
\emptyset &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(Decide)}\\
1^{\mathsf{d}} &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(UnitPropagate)}\\
1^{\mathsf{d}}\,\bar{2} &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(UnitPropagate)}\\
1^{\mathsf{d}}\,\bar{2}\,3 &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(UnitPropagate)}\\
1^{\mathsf{d}}\,\bar{2}\,3\,4 &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(Backtrack)}
\end{aligned}
$$

A Quick Recap

$$
\begin{aligned}
\emptyset &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(Decide)}\\
1^{\mathsf{d}} &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(UnitPropagate)}\\
1^{\mathsf{d}}\,\bar{2} &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(UnitPropagate)}\\
1^{\mathsf{d}}\,\bar{2}\,3 &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(UnitPropagate)}\\
1^{\mathsf{d}}\,\bar{2}\,3\,4 &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(Backtrack)}\\
\bar{1} &\parallel \bar{1}\vee\bar{2},\ 2\vee3,\ \bar{1}\vee\bar{3}\vee4,\ 2\vee\bar{3}\vee\bar{4},\ 1\vee4 && \Longrightarrow && \text{(UnitPropagate)}
\end{aligned}
$$

A Quick Recap

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

A Quick Recap

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

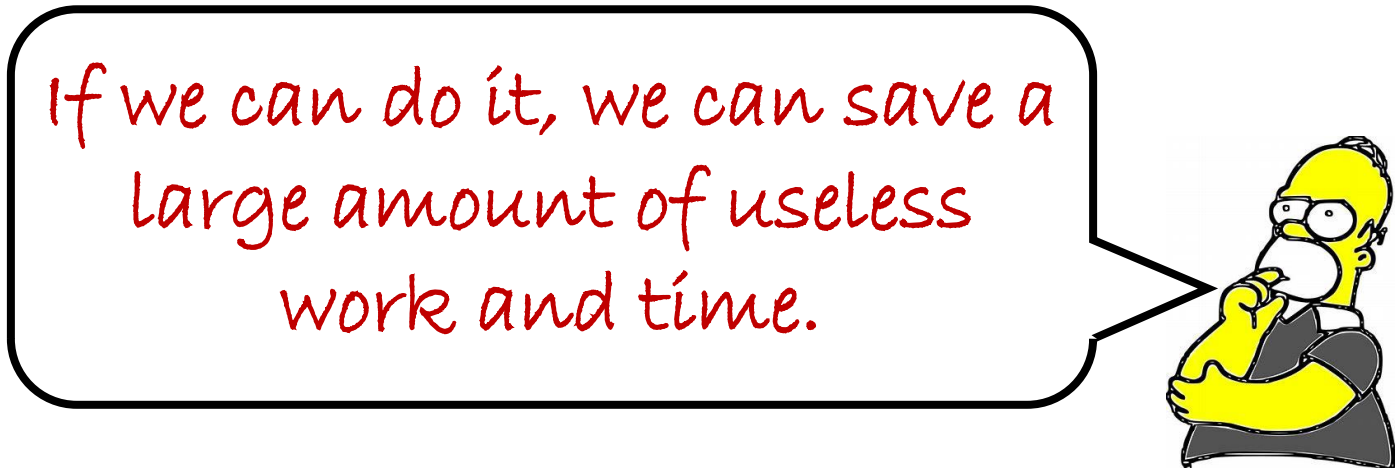
A Quick Recap

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

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

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

Incremental T-solver

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

Ignore pure literal rule here.

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

Incremental T-solver

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

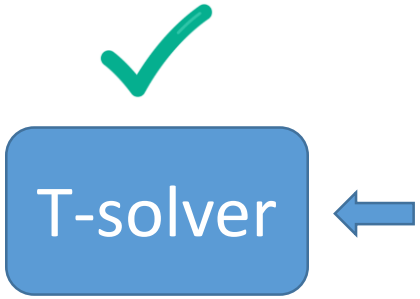
$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

T-solver

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

Incremental T-solver

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



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

Incremental T-solver

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

Restart the SAT solver.

T-solver

$$
\begin{array}{rl}
\emptyset \parallel & 1 \vee 2,\ 3 \vee 4 \vee 5,\ 6,\ \overline{7},\ \overline{8},\ \overline{9} \Longrightarrow (UnitProp) \\
6\ \overline{7}\ \overline{8}\ \overline{9} \parallel & 1 \vee 2,\ 3 \vee 4 \vee 5,\ 6,\ \overline{7},\ \overline{8},\ \overline{9} \Longrightarrow (Decide) \\
6\ \overline{7}\ \overline{8}\ \overline{9}\ \overline{1}^{d} \parallel & 1 \vee 2,\ 3 \vee 4 \vee 5,\ 6,\ \overline{7},\ \overline{8},\ \overline{9} \Longrightarrow (UnitProp) \\
6\ \overline{7}\ \overline{8}\ \overline{9}\ \overline{1}^{d}\ 2 \parallel & 1 \vee 2,\ 3 \vee 4 \vee 5,\ 6,\ \overline{7},\ \overline{8},\ \overline{9}
\end{array}
$$

Incremental T-solver

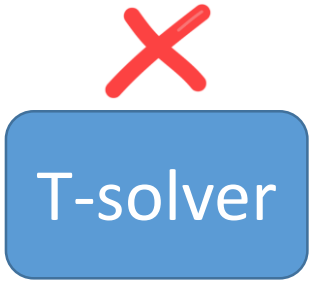
It can be done fully eagerly, detecting unsatisfiability as soon as they are generated. But it may be much too expensive.

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



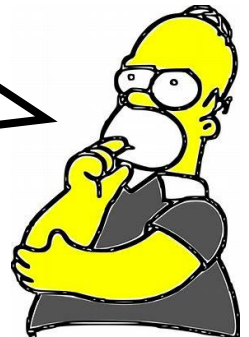
$$\emptyset \parallel\ 1 \vee 2,\ 3 \vee 4 \vee 5,\ 6,\ \overline{7},\ \overline{8},\ \overline{9} \implies (UnitProp)$$

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

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

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

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



Theory Propagation

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

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

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

T-solver

Theory Propagation

$x \neq 3$

$y \neq 3$

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

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

$(z = 3) \wedge$

$\neg(x = y) \wedge$

$\neg(x = z) \wedge$

$\neg(y = z)$

$(p1 \vee p2) \wedge$

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

$(p6) \wedge$

$\neg p7 \wedge$

$\neg p8 \wedge$

$\neg p9$

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

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

T-solver

Theory Propagation

$x \neq 3$

$y \neq 3$

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

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

$(z = 3) \wedge$

$\neg(x = y) \wedge$

$\neg(x = z) \wedge$

$\neg(y = z)$

$(p1 \vee p2) \wedge$

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

$(p6) \wedge$

$\neg p7 \wedge$

$\neg p8 \wedge$

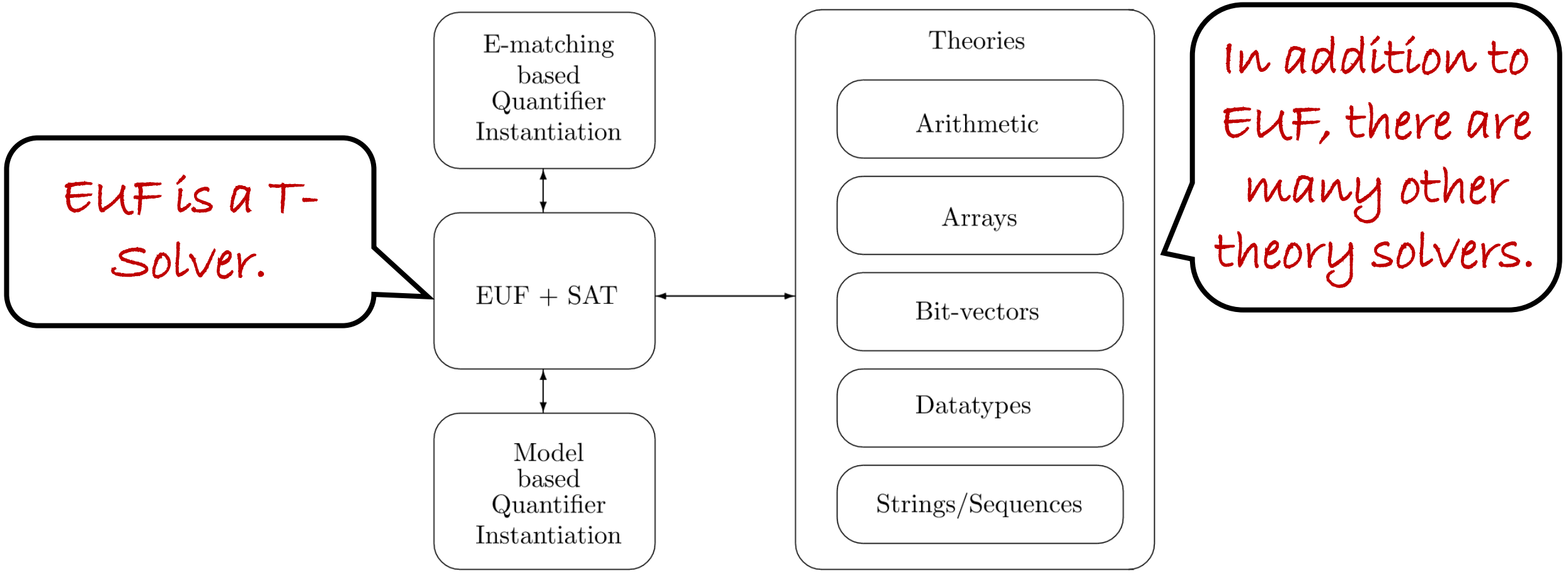
$\neg p9$

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

T-solver

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

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



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.

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

EUF

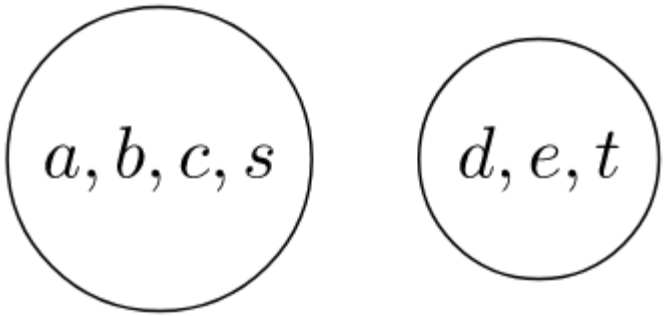
While formulas are not empty:

remove a=b;

merge(a, b);

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

$$b = s, d = t$$



EUF

While formulas are not empty:

remove a=b;

merge(a, b);

If find(a) = find(b) for some a≠b

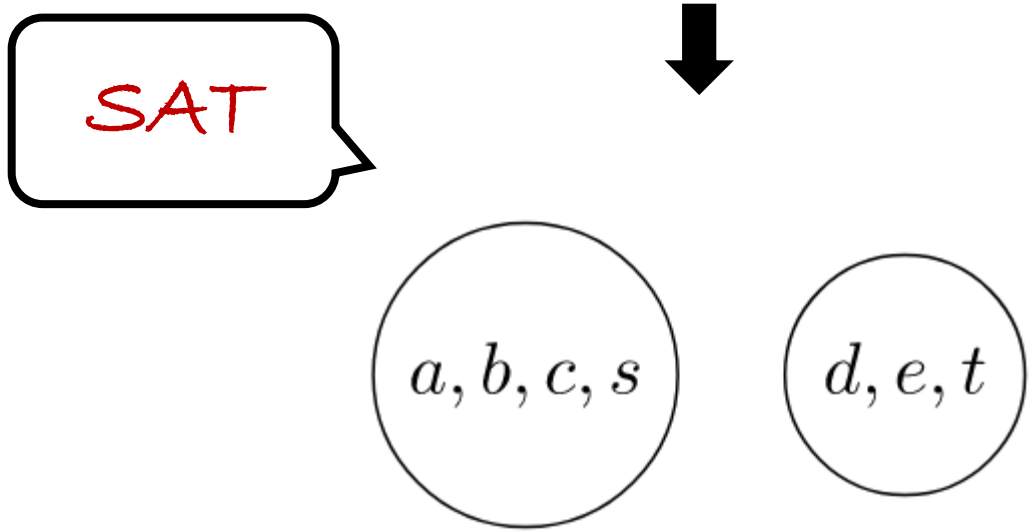
return false;

else

return true;

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

$$b = s, d = t$$



EUF

While formulas are not empty:

remove a=b;

merge(a, b);

If find(a) = find(b) for some a≠b

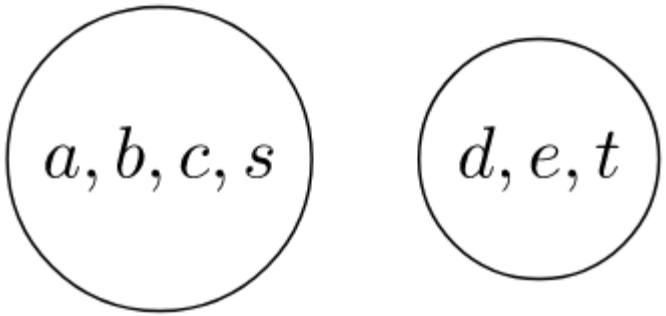
return false;

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

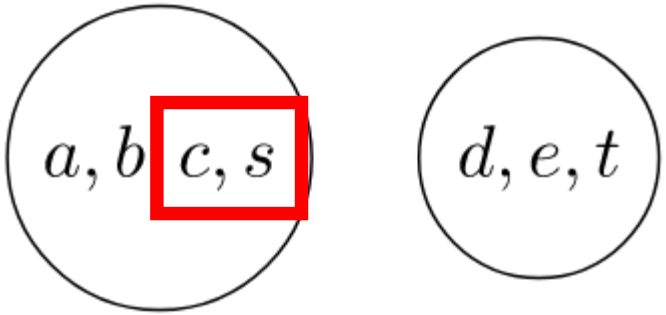
return false;

else

return true;

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

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



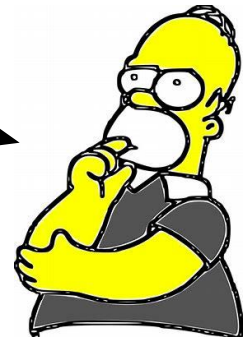
EUF

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

$$b = s, d = t$$

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

What if there are uninterpreted functions?



EUF

THEOREM

Congruence rule:

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

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

$$b = s, d = t$$

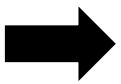
$$f(a, g(d)) \neq f(b, g(e))$$

We can construct the graph with congruence rule.

EUF

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

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



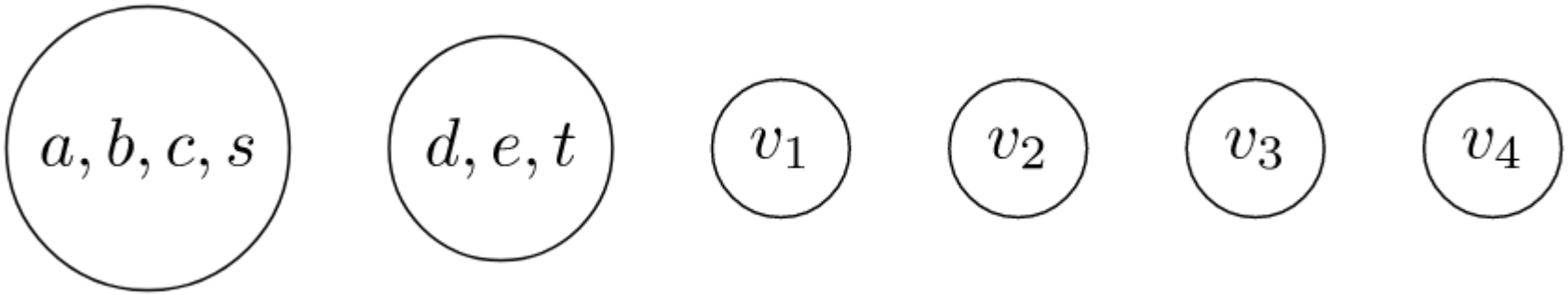
$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$

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

EUF

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

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



EUF

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

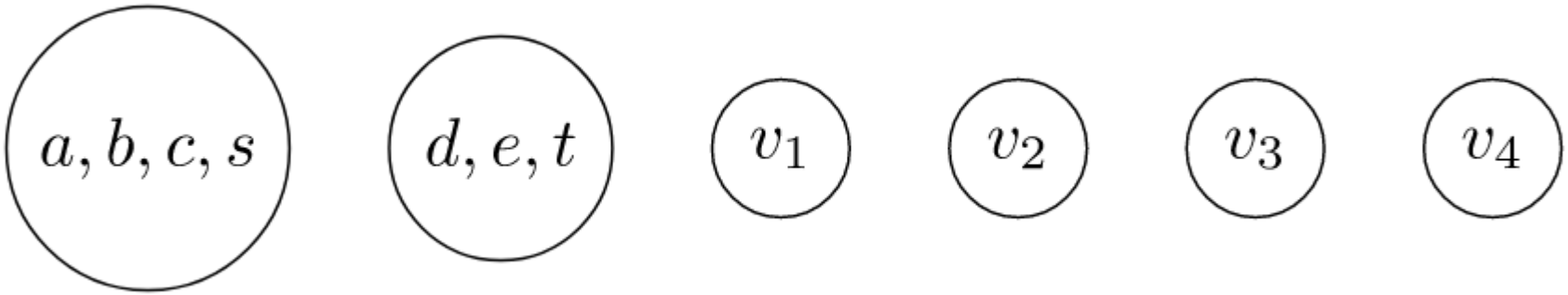
$$a = b, b = c, d = e$$

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

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

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

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

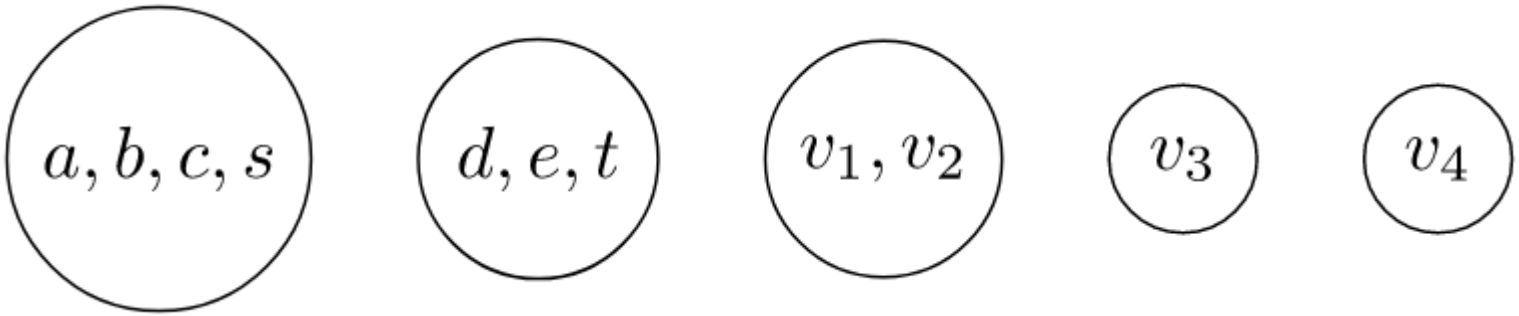


EUF

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

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

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

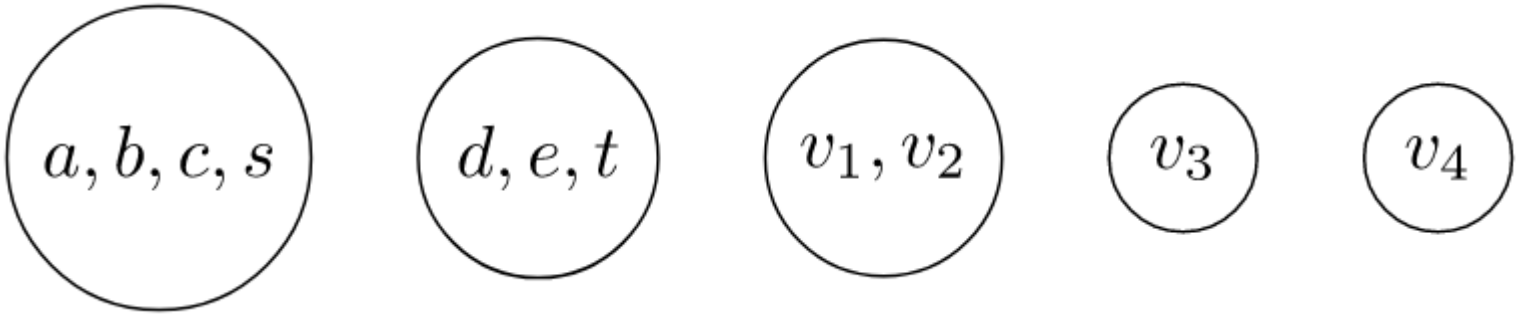


EUF

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

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

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

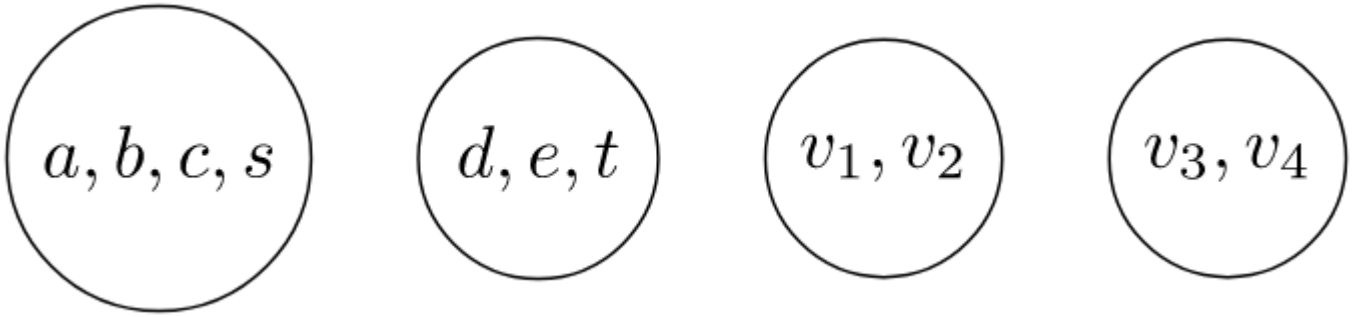


EUF

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

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

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

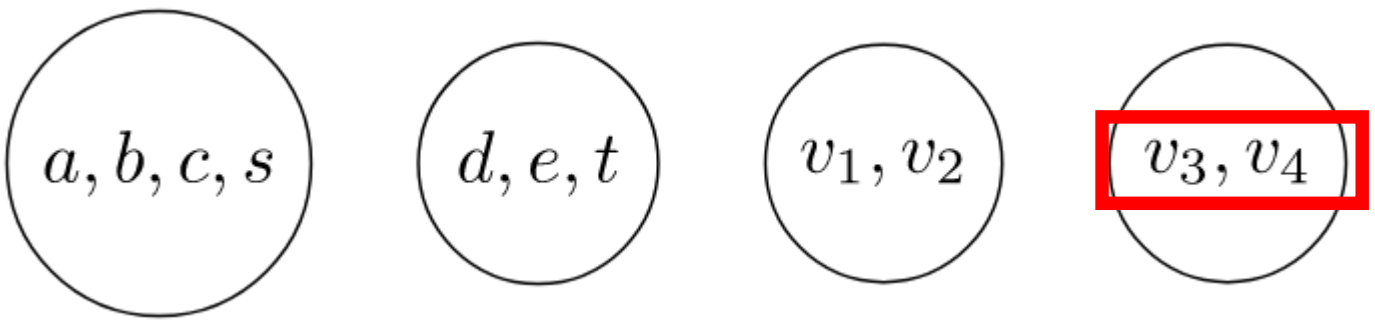


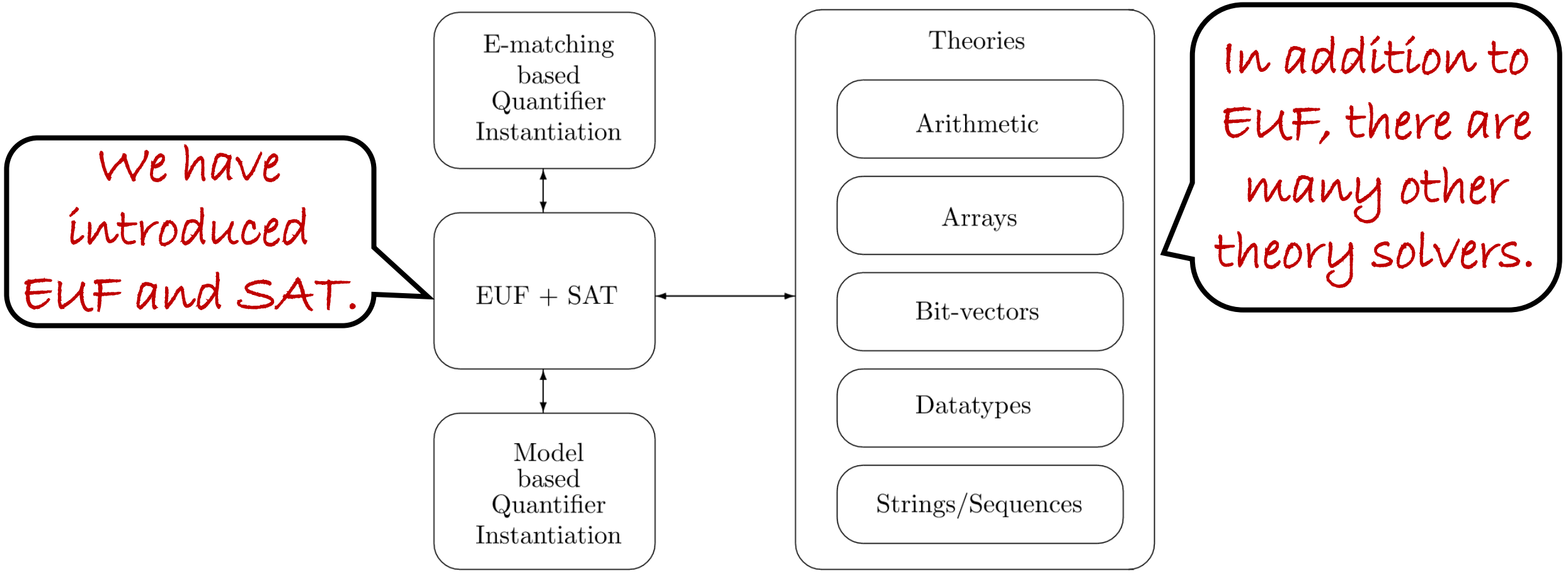
EUF

3) Check satisfiability like before

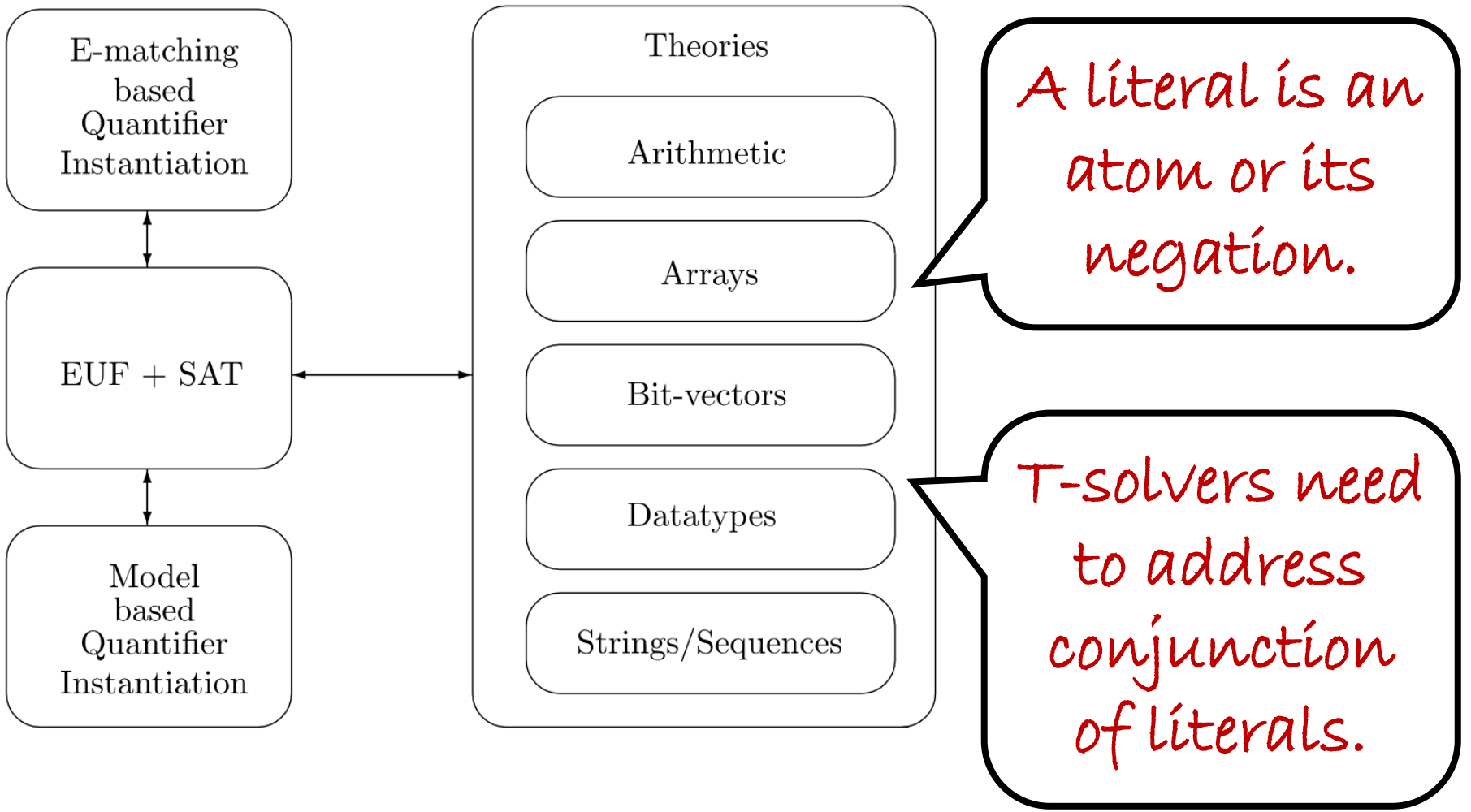
$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$
$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

UNSAT

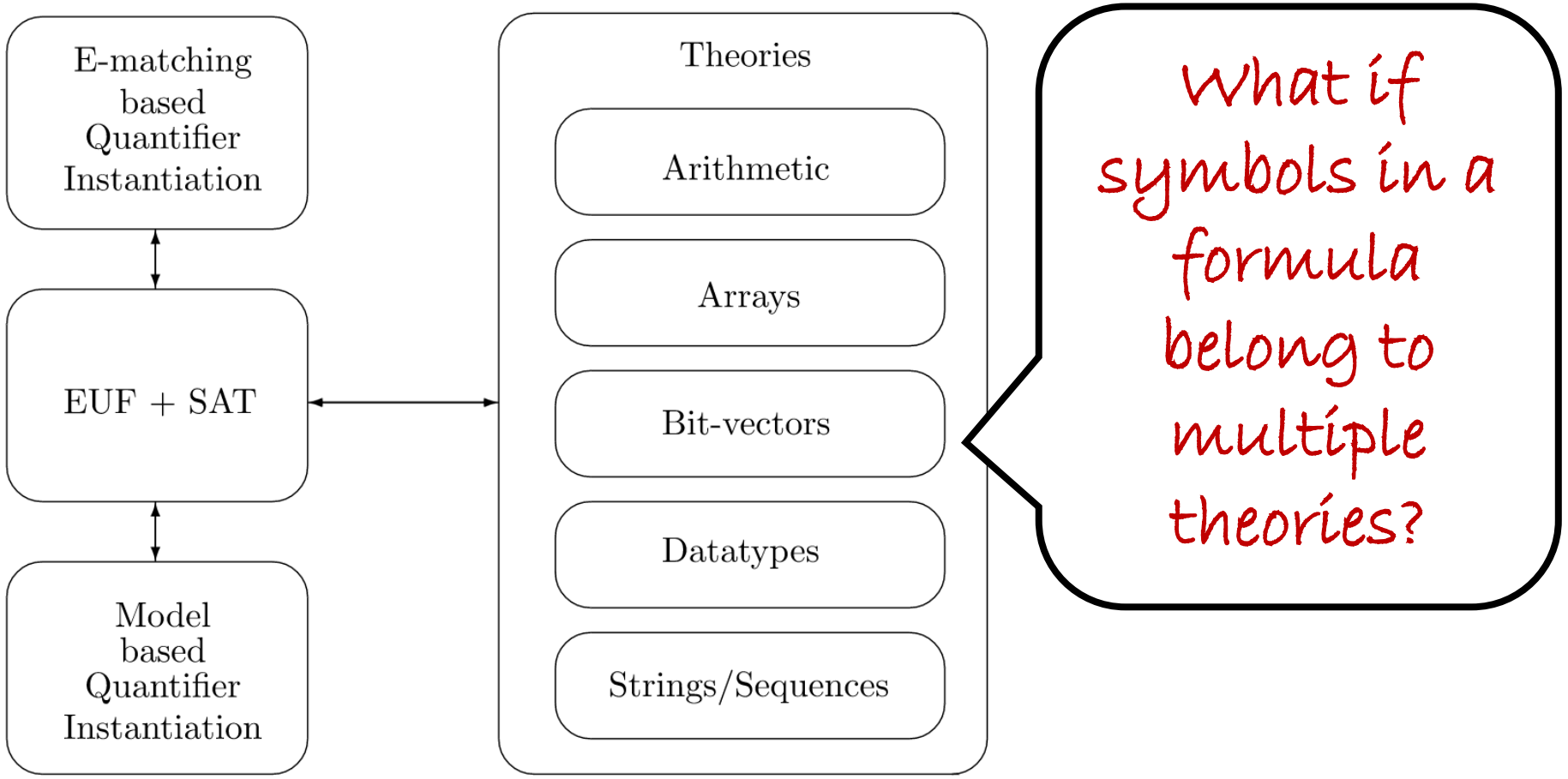




Architecture of Z3's SMT Core solver



Architecture of Z3's SMT Core solver



Architecture of Z3's SMT Core solver

Strawman

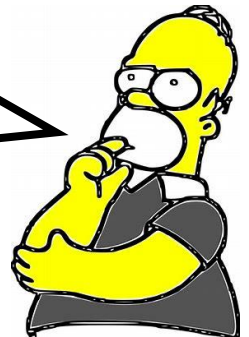
$$b = f(d) \wedge c = f(f(d)) \wedge a > 10 \wedge \neg(c = f(b))$$

EUF Solver

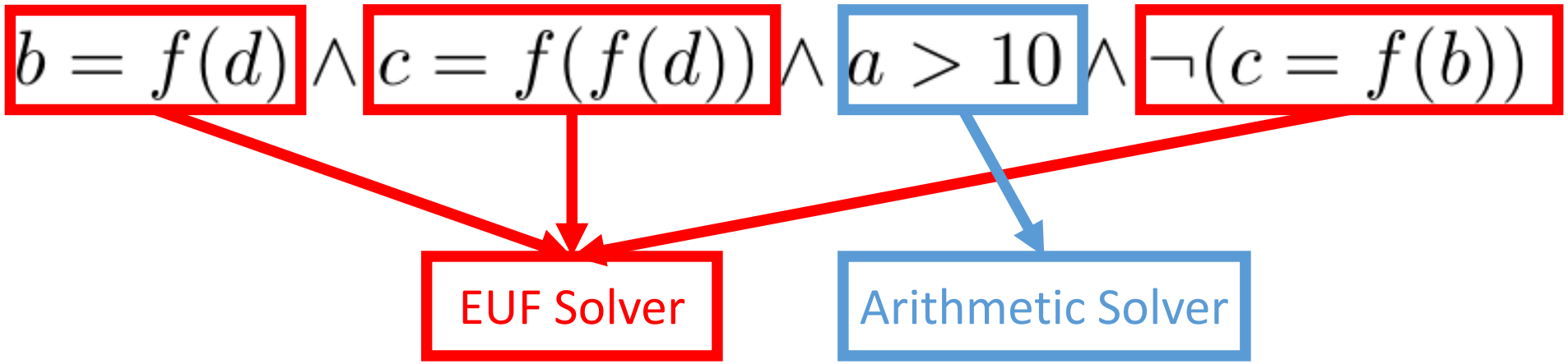
Arithmetic Solver

Integers, +, -, >, =, ...

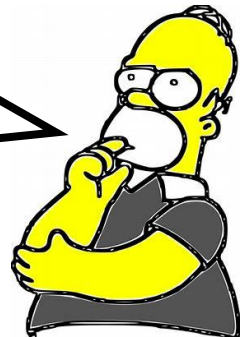
We can assign literals to corresponding theory solvers.



Strawman



If both solvers give SAT, the formula is satisfiable. Otherwise, it must be UNSAT.



Strawman

Even one literal belongs to multiple theories.

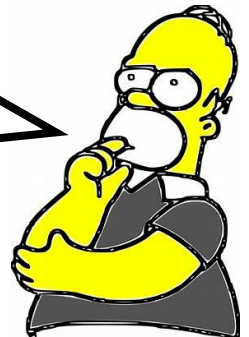
$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(1) \wedge f(a) \neq f(2)$$

Arithmetic Solver

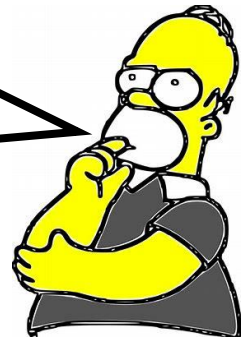
EUF Solver

How to assign them?

However, things seem to be more complicated than we imaged.



Intuitively, we need to convert every literal to a pure form whose symbols belong to only one theory.



The Nelson-Oppen Method

DEFINITION

Terms in predicate logic include variables, constants and function symbols.

Purification

For each literal l in formula A involving terms from both theories:

$$1 \leqslant a \wedge a \leqslant 2 \wedge \boxed{f(a) \neq f(1)} \wedge \boxed{f(a) \neq f(2)}$$

The Nelson-Oppen Method

DEFINITION

Terms in predicate logic include variables, constants and function symbols.

Purification

For each literal l in formula A involving terms from both theories:

1) Select a subterm t of l containing symbols from one theory but not the other

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(1) \wedge f(a) \neq f(2)$$

The Nelson-Oppen Method

DEFINITION

Terms in predicate logic include variables, constants and function symbols.

Purification

For each literal l in formula A involving terms from both theories:

1) Select a subterm t of l containing symbols from one theory but not the other

2) Rewrite t in l by replacing t with a fresh constant symbol c

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(b) \wedge f(a) \neq f(2)$$

The Nelson-Oppen Method

DEFINITION

Terms in predicate logic include variables, constants and function symbols.

Purification

For each literal l in formula A involving terms from both theories:

1) Select a subterm t of l containing symbols from one theory but not the other

2) Rewrite t in l by replacing t with a fresh constant symbol c

3) Conjoin c = t to A

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(b) \wedge f(a) \neq f(2) \wedge \boxed{b = 1}$$

The Nelson-Oppen Method

DEFINITION

Terms in predicate logic include variables, constants and function symbols.

Purification

For each literal l in formula A involving terms from both theories:

1) Select a subterm t of l containing symbols from one theory but not the other

2) Rewrite t in l by replacing t with a fresh constant symbol c

3) Conjoin c = t to A

4) Repeat until each literal's symbols are from single theory

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(b) \wedge f(a) \neq f(c) \wedge b = 1 \wedge c = 2$$

The Nelson-Oppen Method

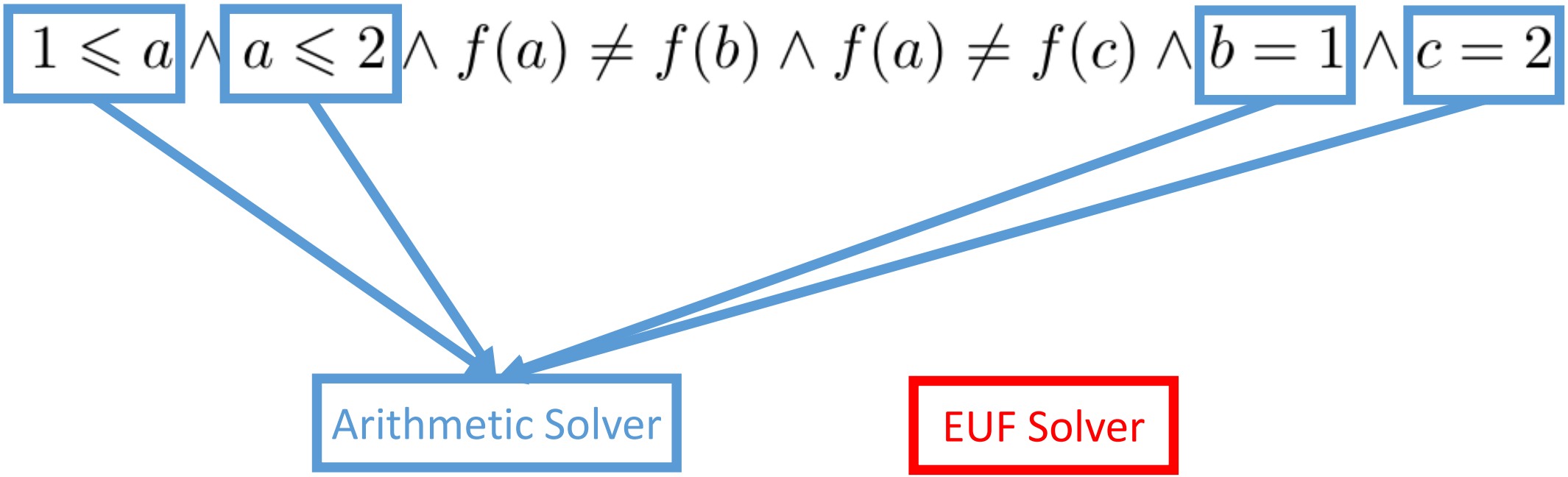
$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(1) \wedge f(a) \neq f(2)$$

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(b) \wedge f(a) \neq f(c) \wedge b = 1 \wedge c = 2$$

After purification, every literal belongs to only one theory.

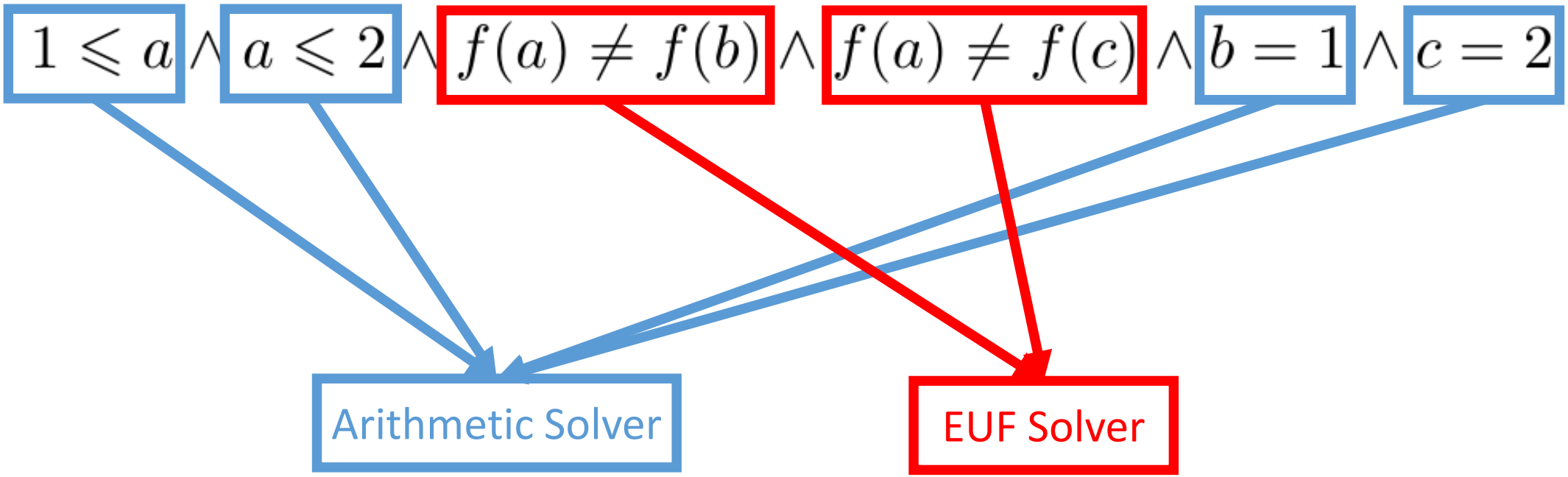
The Nelson-Oppen Method

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(1) \wedge f(a) \neq f(2)$$

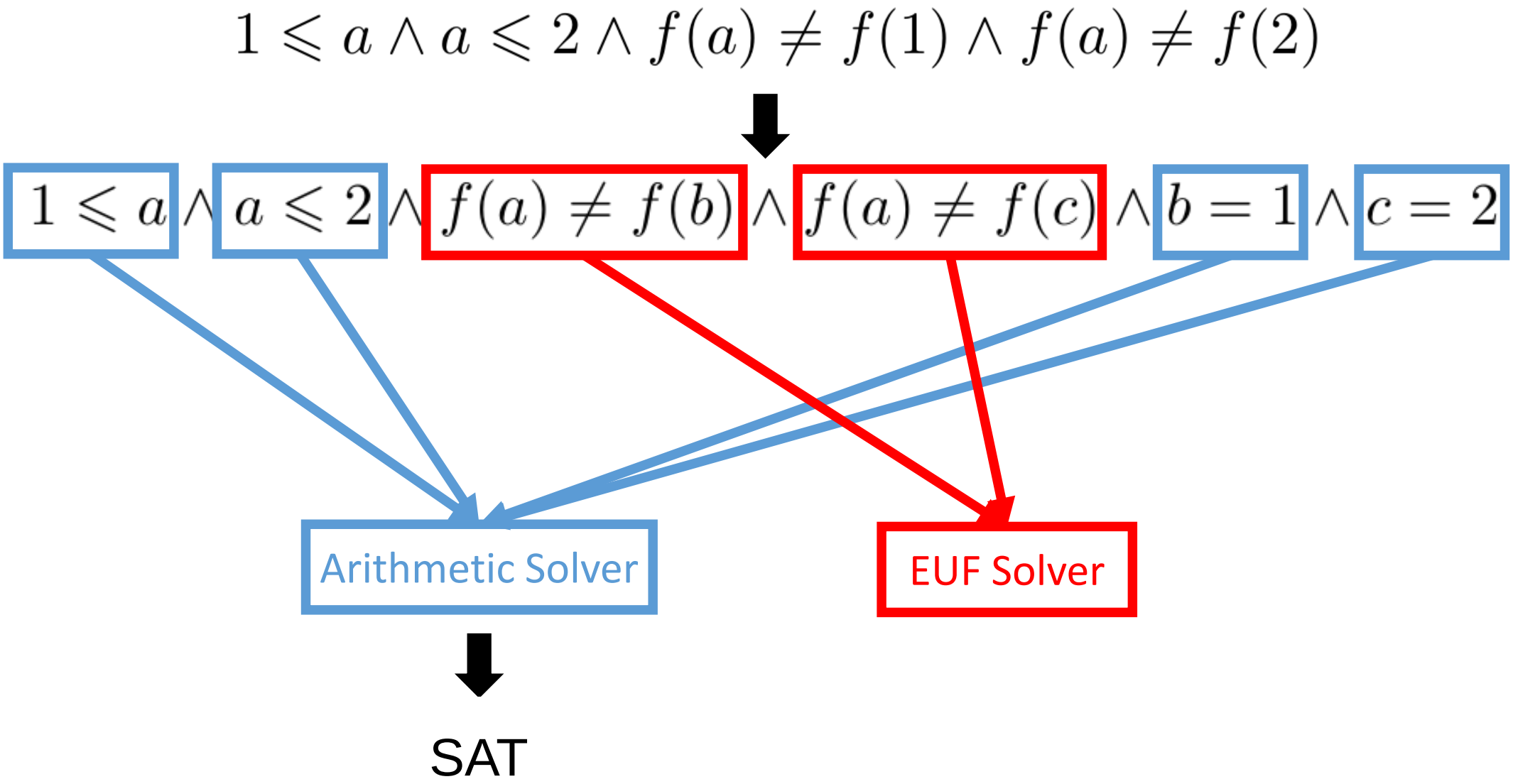


The Nelson-Oppen Method

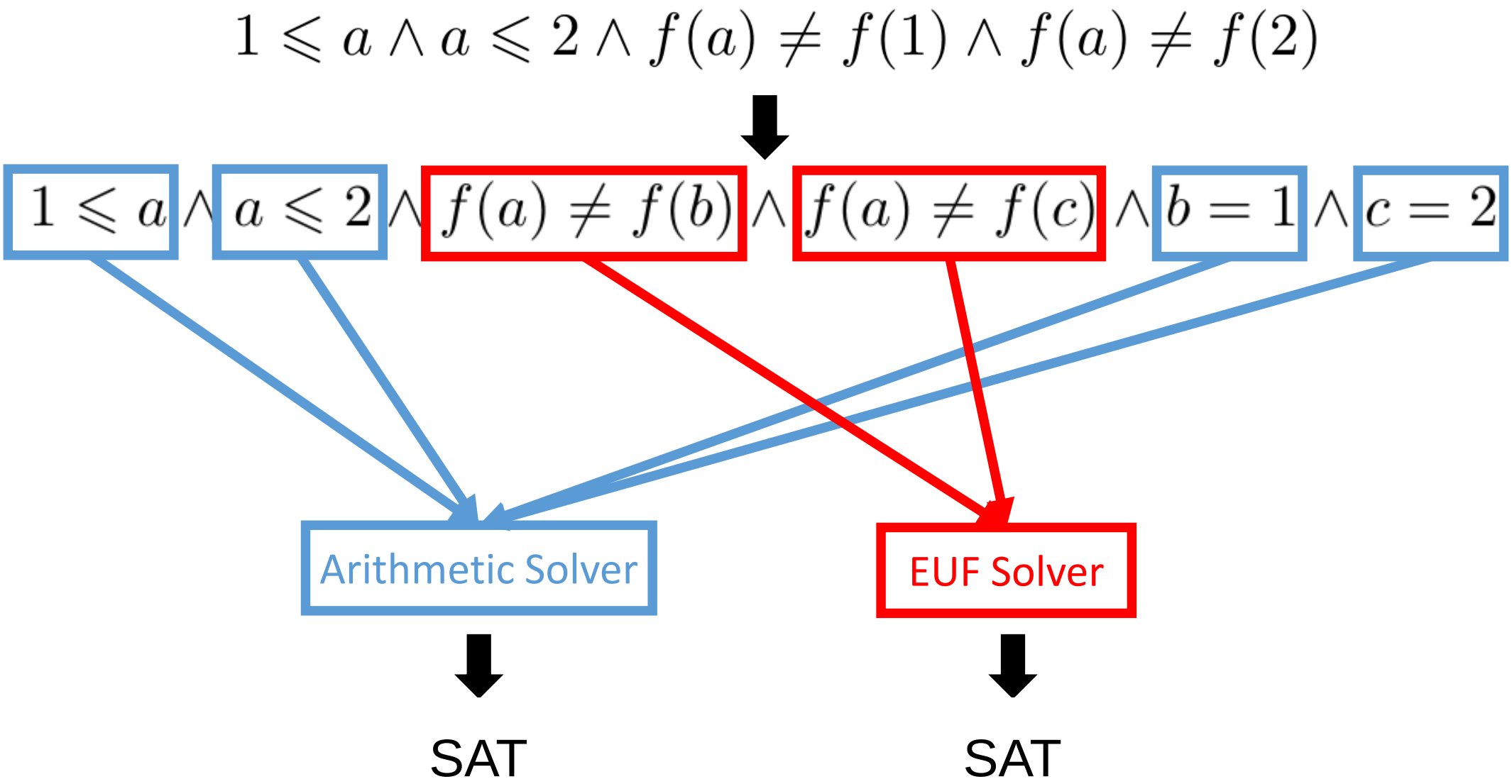
$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(1) \wedge f(a) \neq f(2)$$



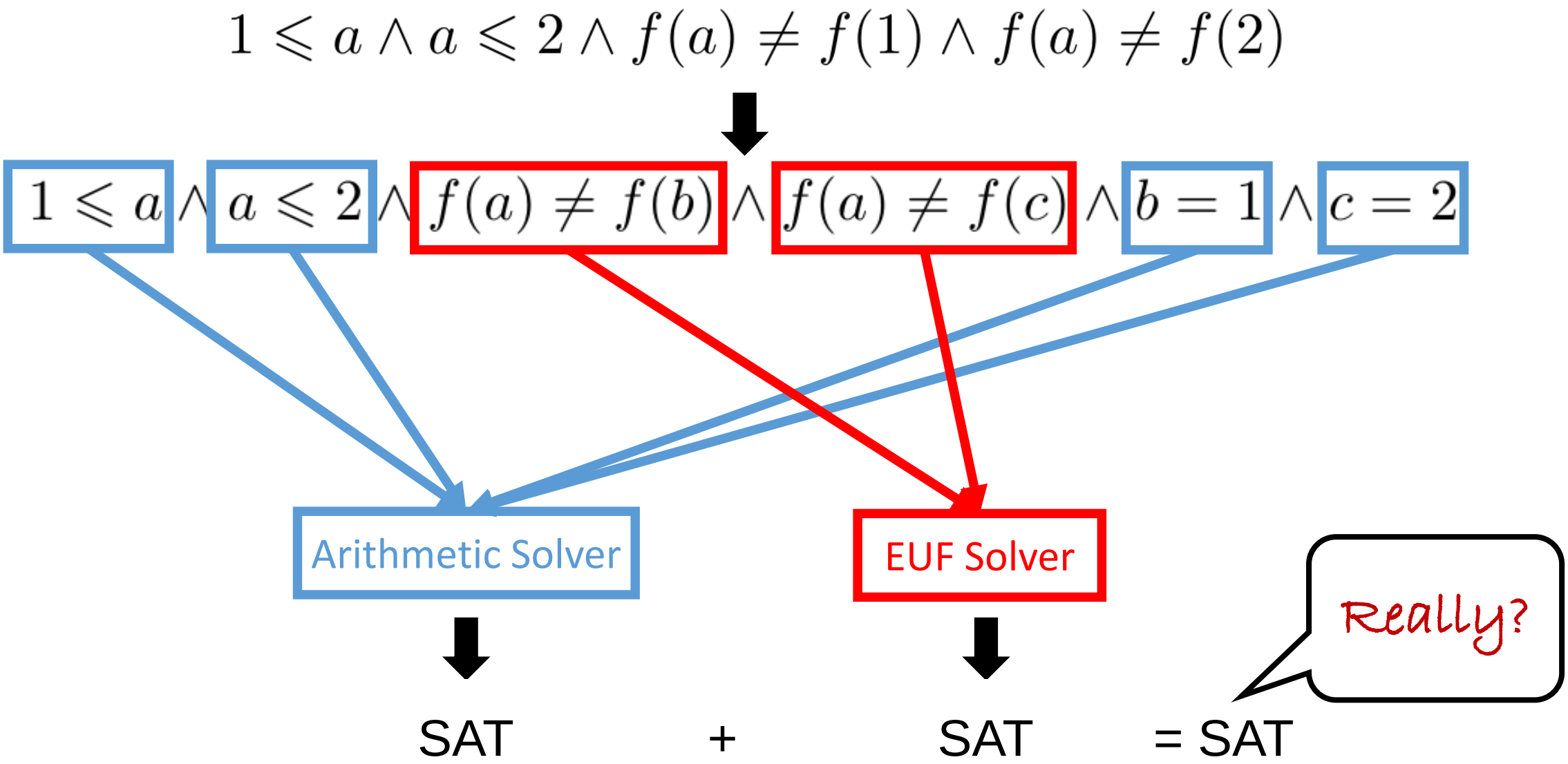
The Nelson-Oppen Method



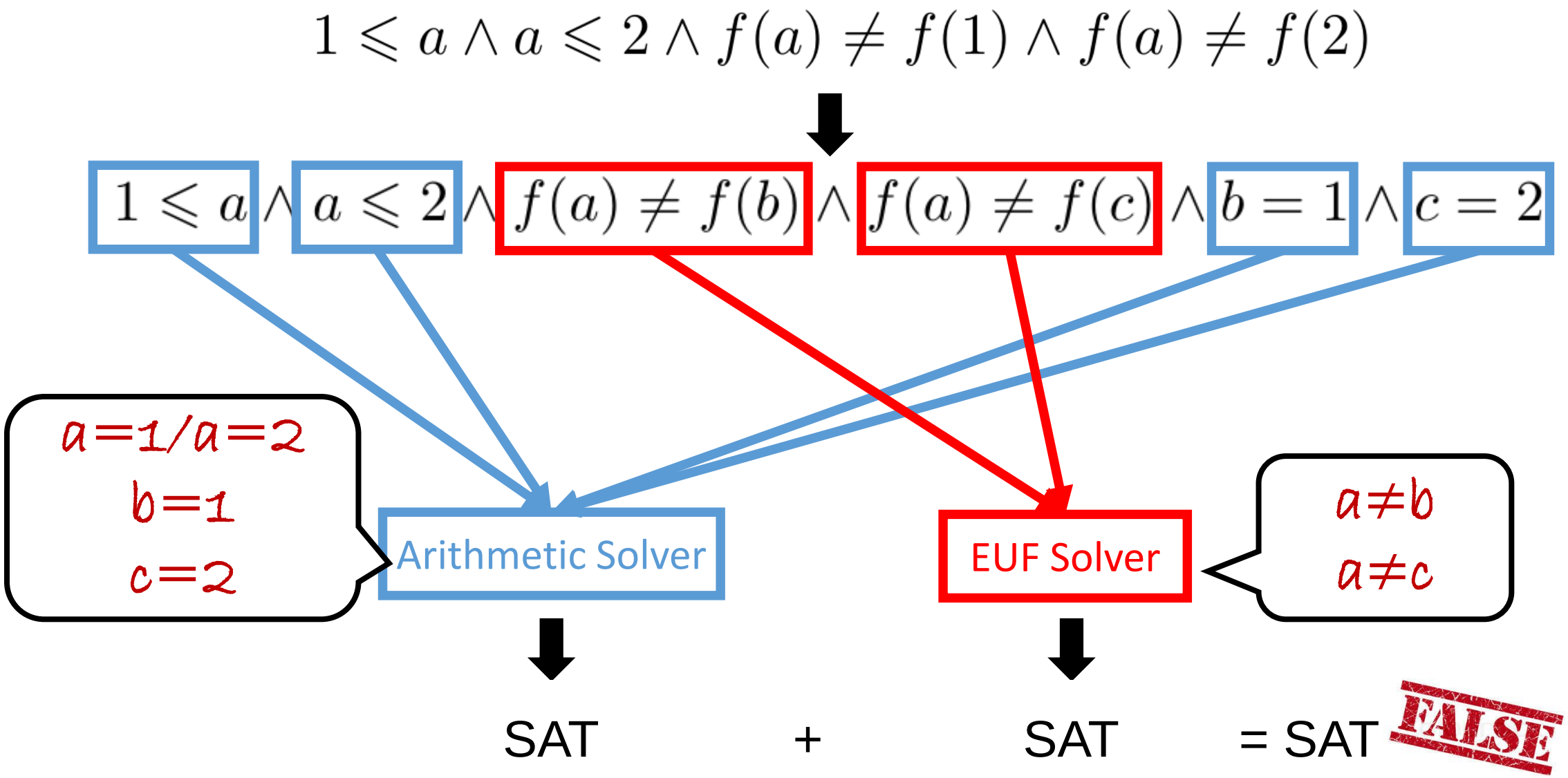
The Nelson-Oppen Method



The Nelson-Oppen Method



The Nelson-Oppen Method



The Nelson-Oppen Method

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(1) \wedge f(a) \neq f(2)$$

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(b) \wedge f(a) \neq f(c) \wedge b = 1 \wedge c = 2$$

a=1/a=2
b=1
c=2

Arithmetic Solver

SAT

EUF Solver

SAT

a≠b
a≠c

It seems to be because of lack of communication between two solvers.

The Nelson-Oppen Method

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(1) \wedge f(a) \neq f(2)$$

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(b) \wedge f(a) \neq f(c) \wedge b = 1 \wedge c = 2$$

DEFINITION

Remaining constants in literals from both theories are called shared constants.

They need to share (dis-)equality information that involves the shared constants.

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

$$1 \leqslant a \wedge a \leqslant 2 \wedge f(a) \neq f(b) \wedge f(a) \neq f(c) \wedge b = 1 \wedge c = 2$$

$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$

Arithmetic Solver

EUF Solver

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

$$a = b, a = c, b = c$$

$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

3) Run EUF solver on A1 and R. Run arithmetic solver on A2 and R

$$a = b, a = c, b = c$$

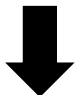
$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

UNSAT

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$



UNSAT

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

3) Run EUF solver on A1 and R. Run arithmetic solver on A2 and R

4) If both return SAT, then return sat. Otherwise, goto 2) and pick a new equivalence relation on S

$$a = b, a = c, b = c$$

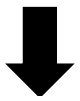
$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

UNSAT

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$



UNSAT

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

3) Run EUF solver on A1 and R. Run arithmetic solver on A2 and R

4) If both return SAT, then return sat. Otherwise, goto 2) and pick a new equivalence relation on S

$$a = b, a = c, b = c$$

$$a = b, a \neq c, b \neq c$$

$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

SAT

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$

UNSAT

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

3) Run EUF solver on A1 and R. Run arithmetic solver on A2 and R

4) If both return SAT, then return sat. Otherwise, goto 2) and pick a new equivalence relation on S

$$a = b, a = c, b = c$$

$$a = b, a \neq c, b \neq c$$

$$a \neq b, a = c, b \neq c$$

$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

SAT

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$

UNSAT

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

3) Run EUF solver on A1 and R. Run arithmetic solver on A2 and R

4) If both return SAT, then return sat. Otherwise, goto 2) and pick a new equivalence relation on S

$$a = b, a = c, b = c$$

$$a = b, a \neq c, b \neq c$$

$$a \neq b, a = c, b \neq c$$

$$a \neq b, a \neq c, b = c$$

$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

UNSAT

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$

SAT

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

3) Run EUF solver on A1 and R. Run arithmetic solver on A2 and R

4) If both return SAT, then return sat. Otherwise, goto 2) and pick a new equivalence relation on S

$$a = b, a = c, b = c$$

$$a = b, a \neq c, b \neq c$$

$$a \neq b, a = c, b \neq c$$

$$a \neq b, a \neq c, b = c$$

$$a \neq b, a \neq c, b \neq c$$

$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

UNSAT

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$

SAT

The Nelson-Oppen Method

After purification, let S be the shared constants and let A be the formula

1) Assign literals to each T-solver (assign A1 to EUF solver and A2 to arithmetic solver)

2) Guess an equivalence relation R on S

3) Run EUF solver on A1 and R. Run arithmetic solver on A2 and R

4) If both return SAT, then return sat. Otherwise, goto 2) and pick a new equivalence relation on S

5) If all have equivalence relation been tried, return UNSAT

$$a = b, a = c, b = c$$

$$a = b, a \neq c, b \neq c$$

$$a \neq b, a = c, b \neq c$$

$$a \neq b, a \neq c, b = c$$

$$a \neq b, a \neq c, b \neq c$$

$$1 \leqslant a \wedge a \leqslant 2$$

$$b = 1 \wedge c = 2$$

$$f(a) \neq f(b) \wedge$$

$$f(a) \neq f(c)$$

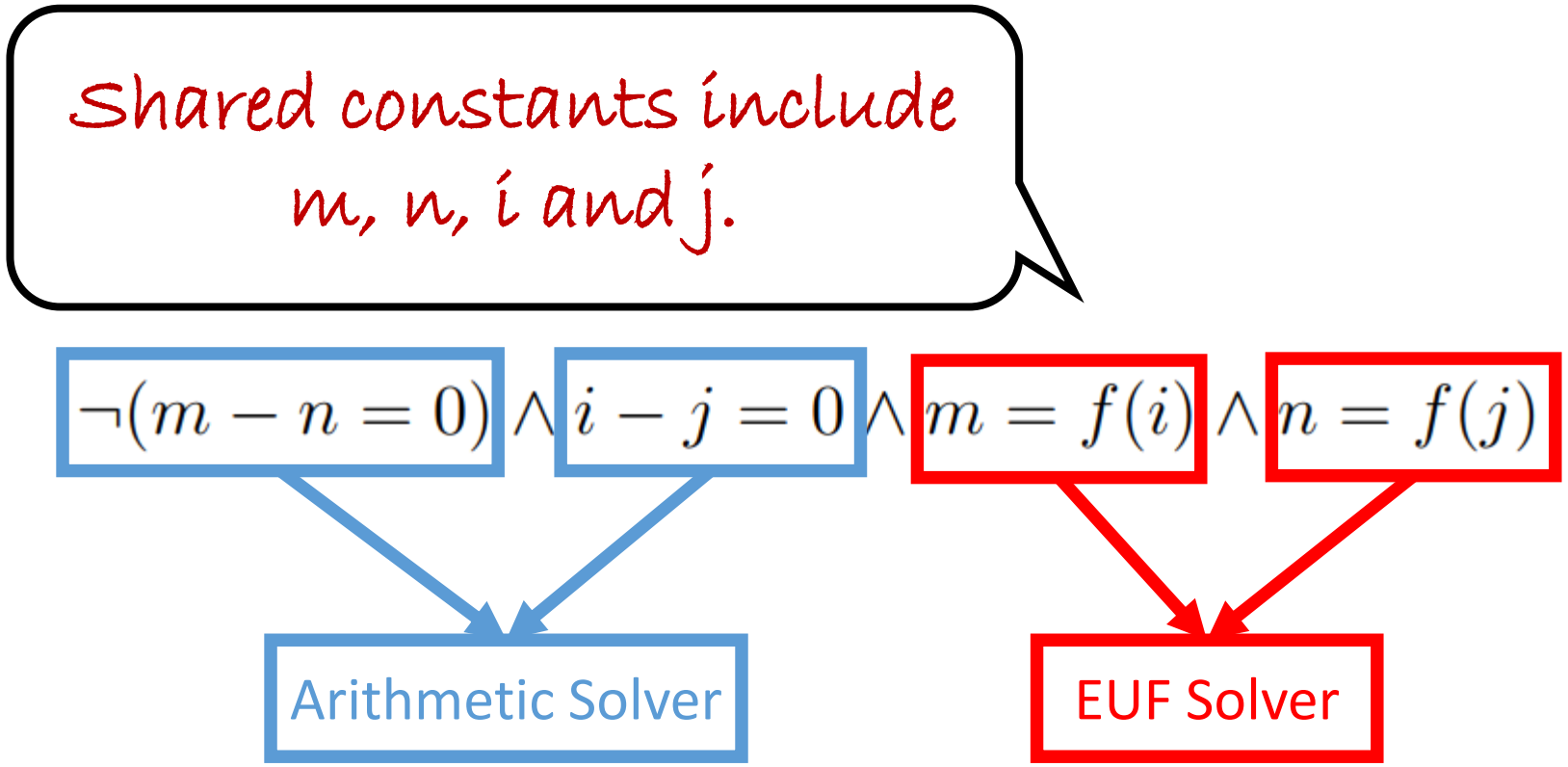
Exercise

$$\neg(f(i) - f(j) = 0) \wedge i - j = 0$$

Translate the formula and identify the shared constants.

Exercise

$$\neg(f(i) - f(j) = 0) \wedge i - j = 0$$



Thanks & Questions