

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

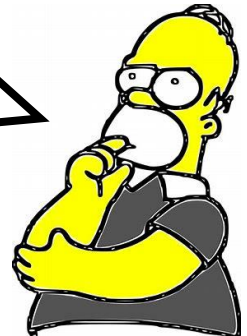
- SMT Solver
 - From DPLL to DPLL(T)
 - EUF

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?





MathSAT 5

An SMT Solver for Formal Verification & More

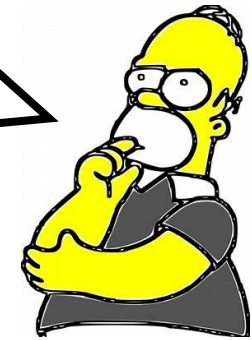
SMT solver can do it!

Z3

CVC3

CVC4

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



We only consider the case of *quantifier-free formulas*.

A Quick Recap

$$\emptyset \parallel \bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4 \implies (\text{Decide})$$

A Quick Recap

$$\begin{array}{llllllll} \emptyset & \parallel & \bar{1} \vee \bar{2}, & 2 \vee 3, & \bar{1} \vee \bar{3} \vee 4, & 2 \vee \bar{3} \vee \bar{4}, & 1 \vee 4 & \implies & (\text{Decide}) \\ 1^d & \parallel & \bar{1} \vee \bar{2}, & 2 \vee 3, & \bar{1} \vee \bar{3} \vee 4, & 2 \vee \bar{3} \vee \bar{4}, & 1 \vee 4 & \implies & (\text{UnitPropagate}) \end{array}$$

A Quick Recap

$$\begin{array}{llllllll} \emptyset & \parallel & \bar{1} \vee \bar{2}, & 2 \vee 3, & \bar{1} \vee \bar{3} \vee 4, & 2 \vee \bar{3} \vee \bar{4}, & 1 \vee 4 & \implies & (\text{Decide}) \\ 1^d & \parallel & \bar{1} \vee \bar{2}, & 2 \vee 3, & \bar{1} \vee \bar{3} \vee 4, & 2 \vee \bar{3} \vee \bar{4}, & 1 \vee 4 & \implies & (\text{UnitPropagate}) \\ 1^d \bar{2} & \parallel & \bar{1} \vee \bar{2}, & 2 \vee 3, & \bar{1} \vee \bar{3} \vee 4, & 2 \vee \bar{3} \vee \bar{4}, & 1 \vee 4 & \implies & (\text{UnitPropagate}) \end{array}$$

A Quick Recap

$\emptyset$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
1^d	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)

A Quick Recap

$\emptyset$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
1^d	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Backtrack)

A Quick Recap

$\emptyset$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
1^d	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Backtrack)
$\bar{1}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)

A Quick Recap

$\emptyset$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
1^d	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Backtrack)
$\bar{1}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$\bar{1} 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)

A Quick Recap

$\emptyset$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
1^d	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Backtrack)
$\bar{1}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$\bar{1} 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
$\bar{1} 4 \bar{3}^d$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)

A Quick Recap

$\emptyset$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
1^d	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Backtrack)
$\bar{1}$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$\bar{1} 4$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(Decide)
$\bar{1} 4 \bar{3}^d$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$	$\implies$	(UnitPropagate)
$\bar{1} 4 \bar{3}^d 2$	$\parallel$	$\bar{1} \vee \bar{2},$	$2 \vee 3,$	$\bar{1} \vee \bar{3} \vee 4,$	$2 \vee \bar{3} \vee \bar{4},$	$1 \vee 4$		

A Quick Recap

$\emptyset$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(Decide)
1^d	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2}$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(UnitPropagate)
$1^d \bar{2} 3 4$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(Backtrack)
$\bar{1}$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(UnitPropagate)
$\bar{1} 4$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(Decide)
$\bar{1} 4 \bar{3}^d$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$	$\implies$	(UnitPropagate)
$\bar{1} 4 \bar{3}^d 2$	$\parallel$	$\bar{1} \vee \bar{2}, 2 \vee 3, \bar{1} \vee \bar{3} \vee 4, 2 \vee \bar{3} \vee \bar{4}, 1 \vee 4$		

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

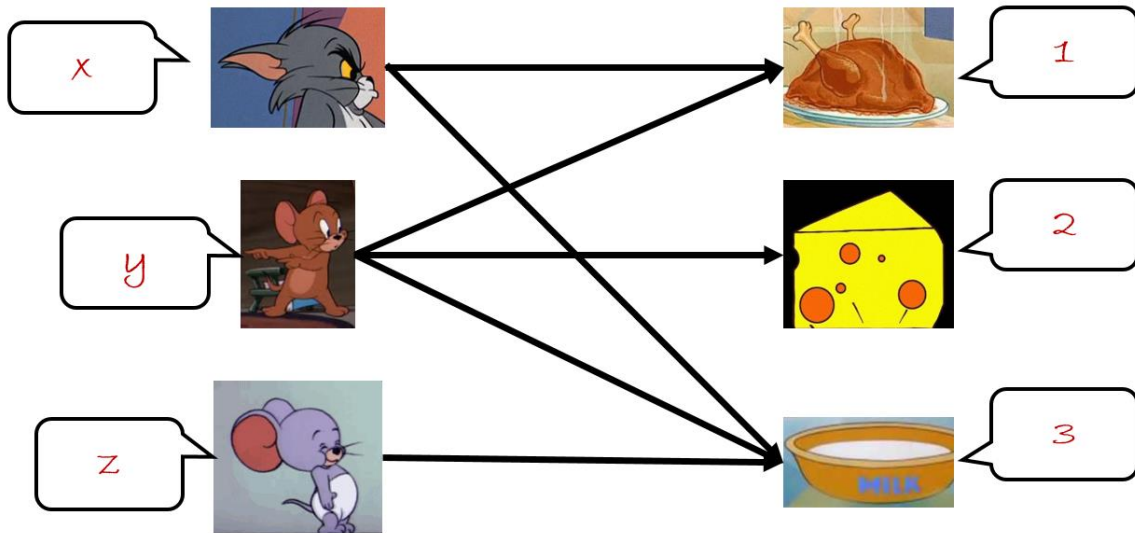


Lazy SMT integrates theory support
into the propositional search

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

$(x = 1 \vee x = 3) \wedge$	$(p1 \vee p2) \wedge$
$(y = 1 \vee y = 2 \vee y = 3) \wedge$	$(p3 \vee p4 \vee p5) \wedge$
$(z = 3) \wedge$	$(p6) \wedge$
$\neg(x = y) \wedge$	$\neg p7 \wedge$
$\neg(x = z) \wedge$	$\neg p8 \wedge$
$\neg(y = z)$	$\neg p9$

Step1: replace atomic formulas with fresh propositional variables.

Lazy SMT Techniques

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



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



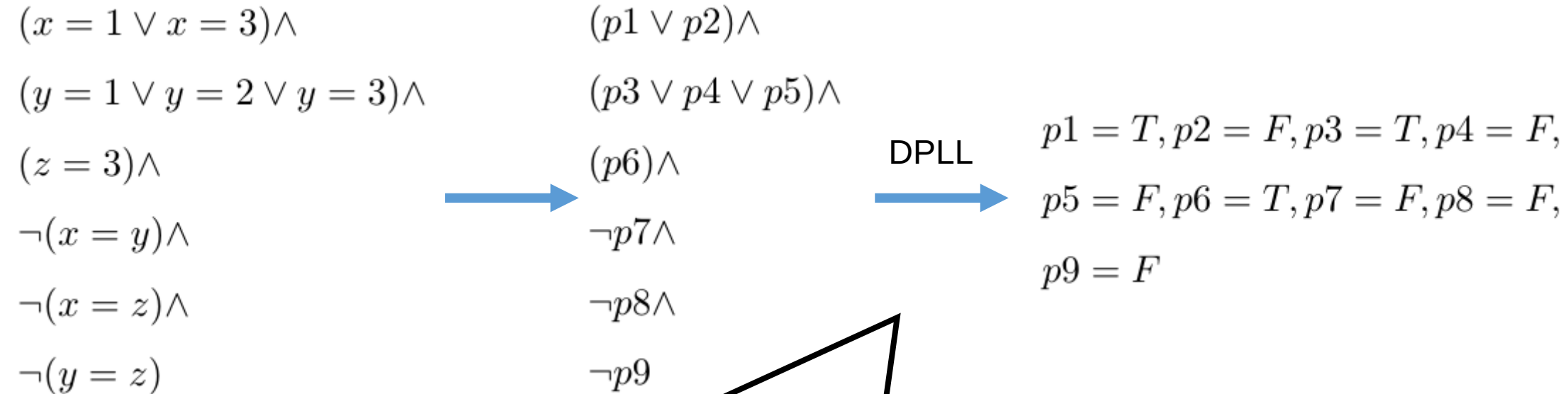
DPLL

UNSAT

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

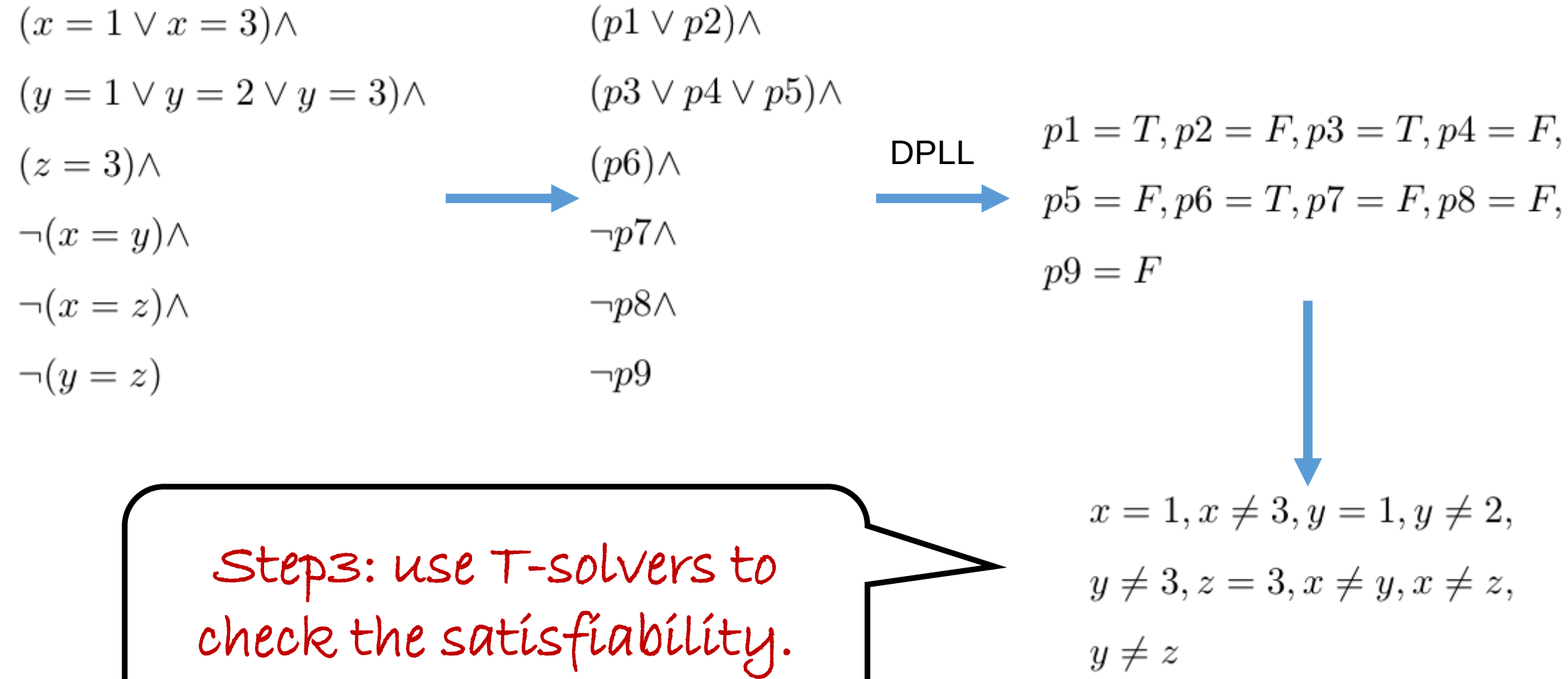
Step2: use SAT solvers to
get assignments.

Lazy SMT Techniques

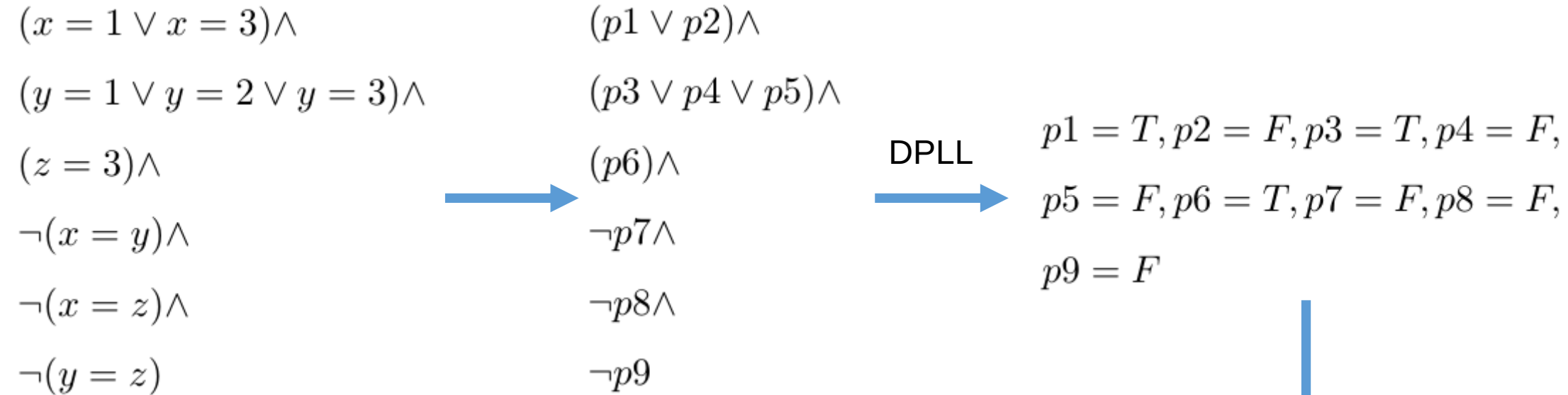


Step2: use SAT solvers to
get assignments.

Lazy SMT Techniques



Lazy SMT Techniques



Step3: use T-solvers to check the satisfiability.

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

FALSE

Lazy SMT Techniques

$(x = 1 \vee x = 3) \wedge$	$(p1 \vee p2) \wedge$
$(y = 1 \vee y = 2 \vee y = 3) \wedge$	$(p3 \vee p4 \vee p5) \wedge$
$(z = 3) \wedge$	$(p6) \wedge$
$\neg(x = y) \wedge$	$\neg p7 \wedge$
$\neg(x = z) \wedge$	$\neg p8 \wedge$
$\neg(y = z)$	$\neg p9$

DPLL

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

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

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

FALSE

Lazy SMT Techniques

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

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

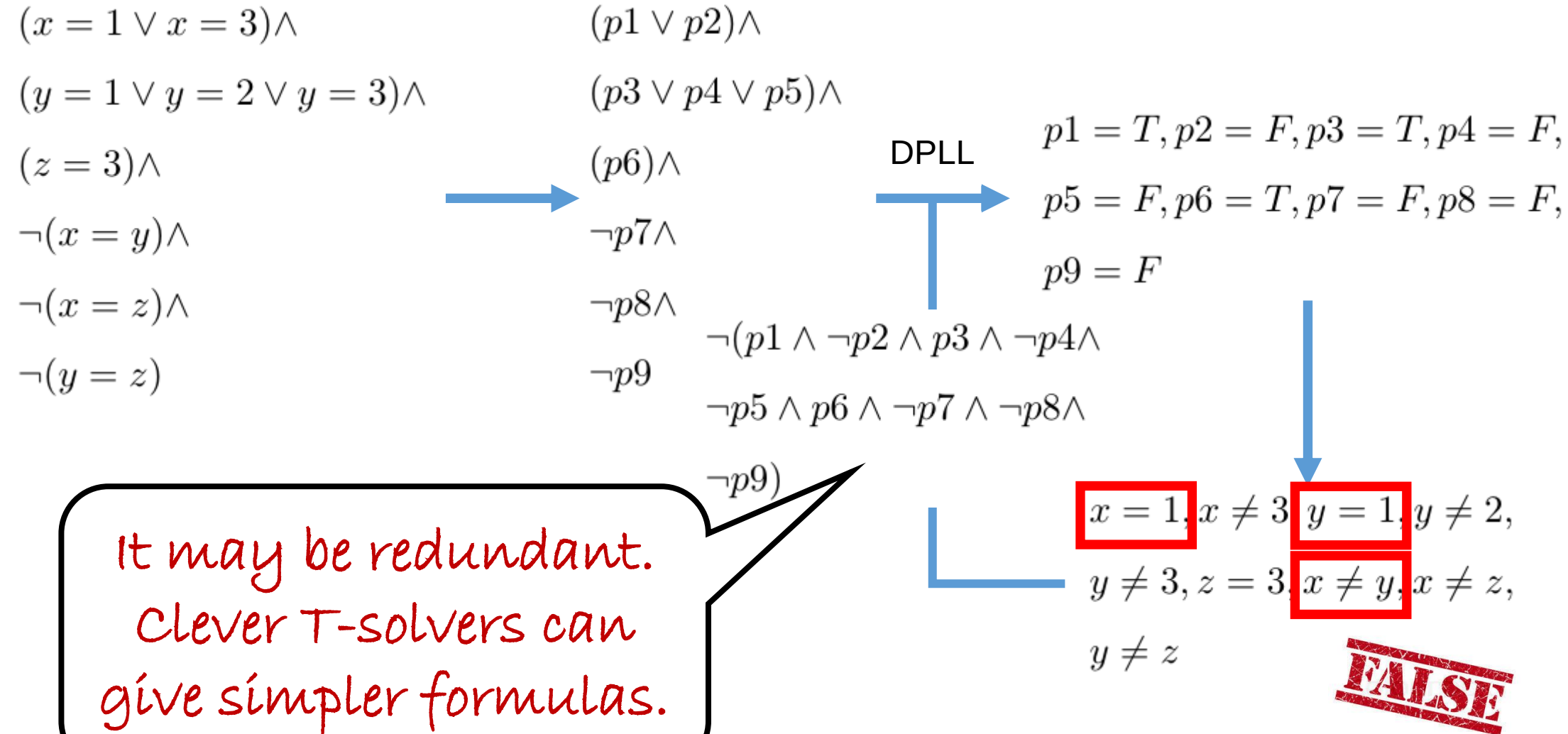
DPLL

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

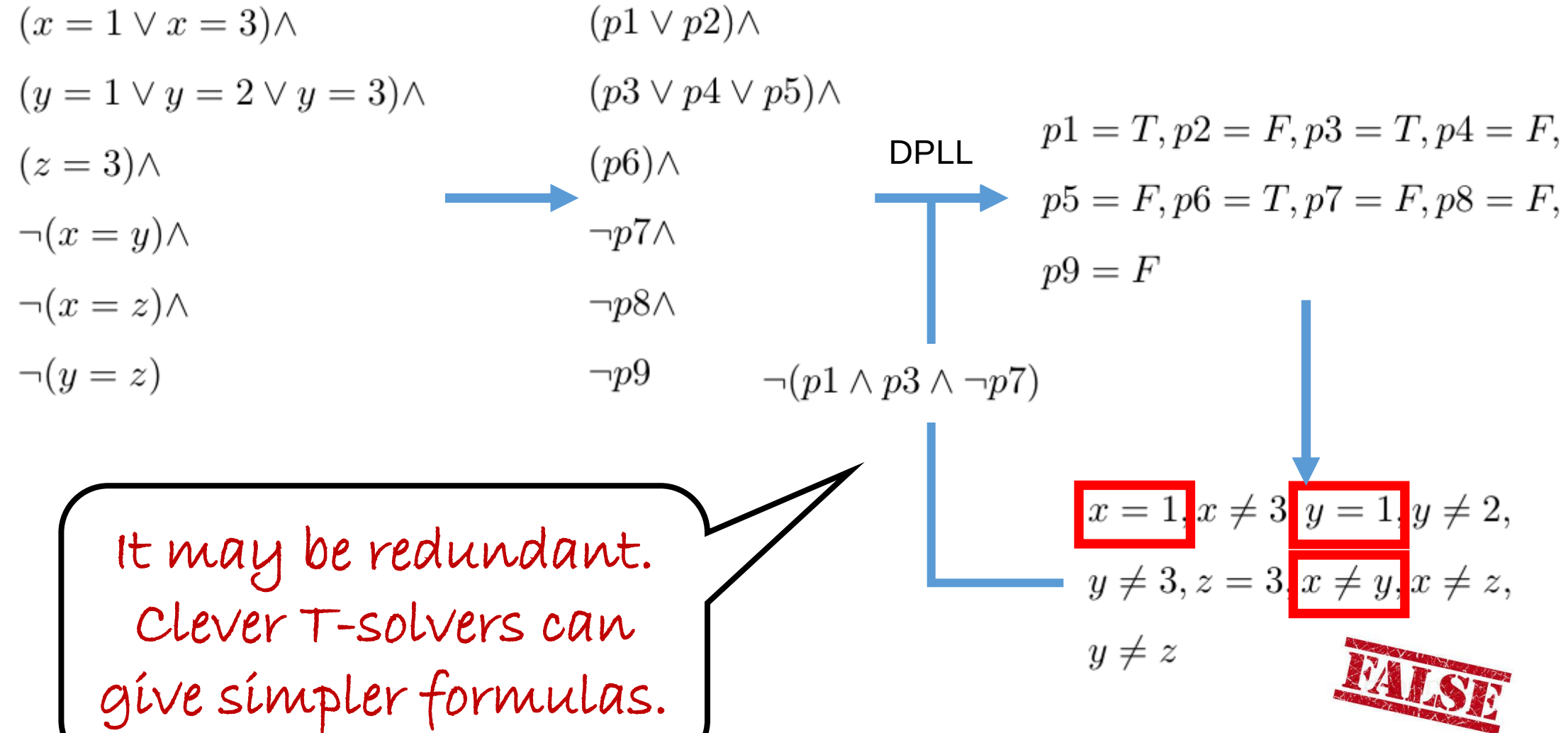
$$\begin{array}{l} x = 1, x \neq 3, y = 1, y \neq 2, \\ y \neq 3, z = 3, x \neq y, x \neq z, \\ y \neq z \end{array}$$

FALSE

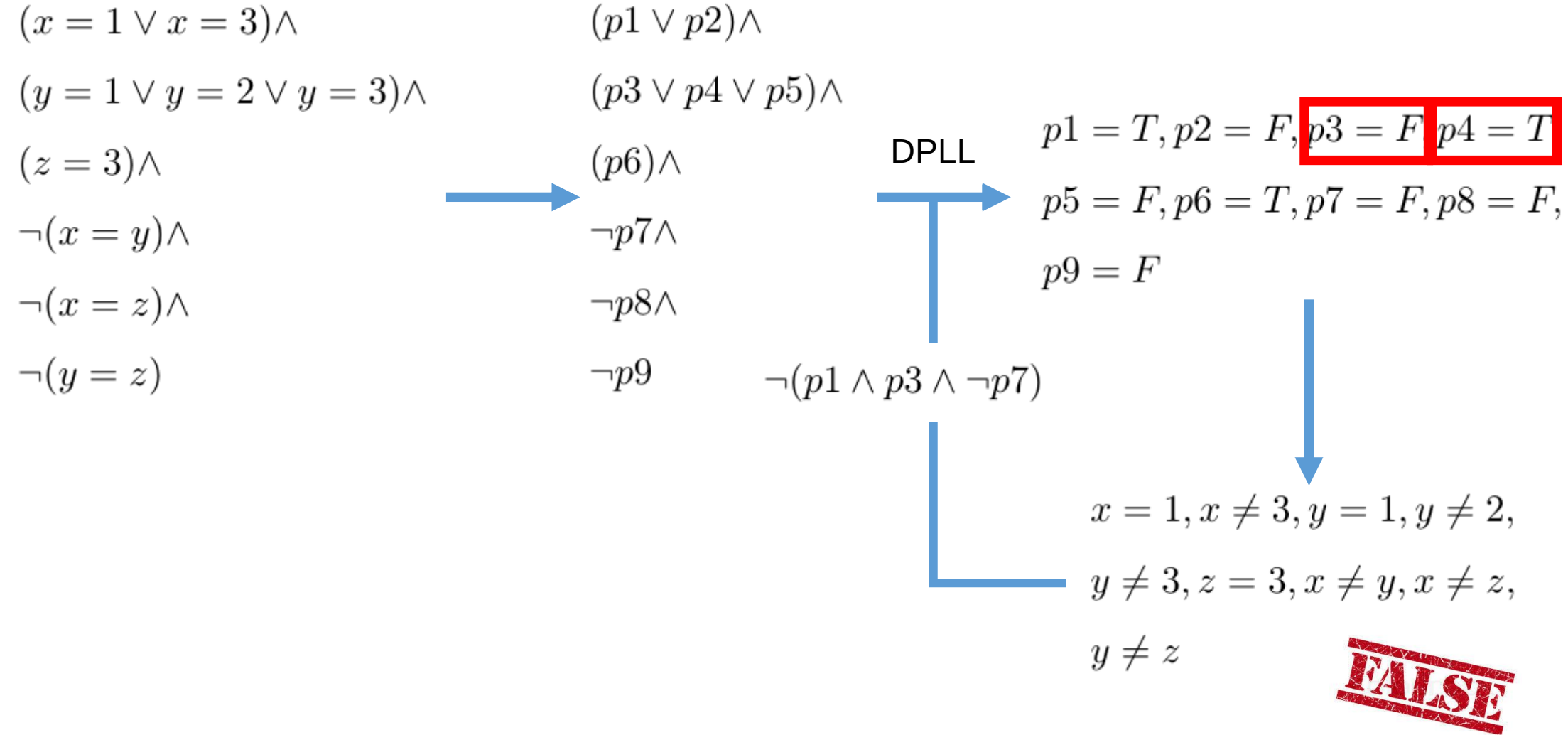
Lazy SMT Techniques



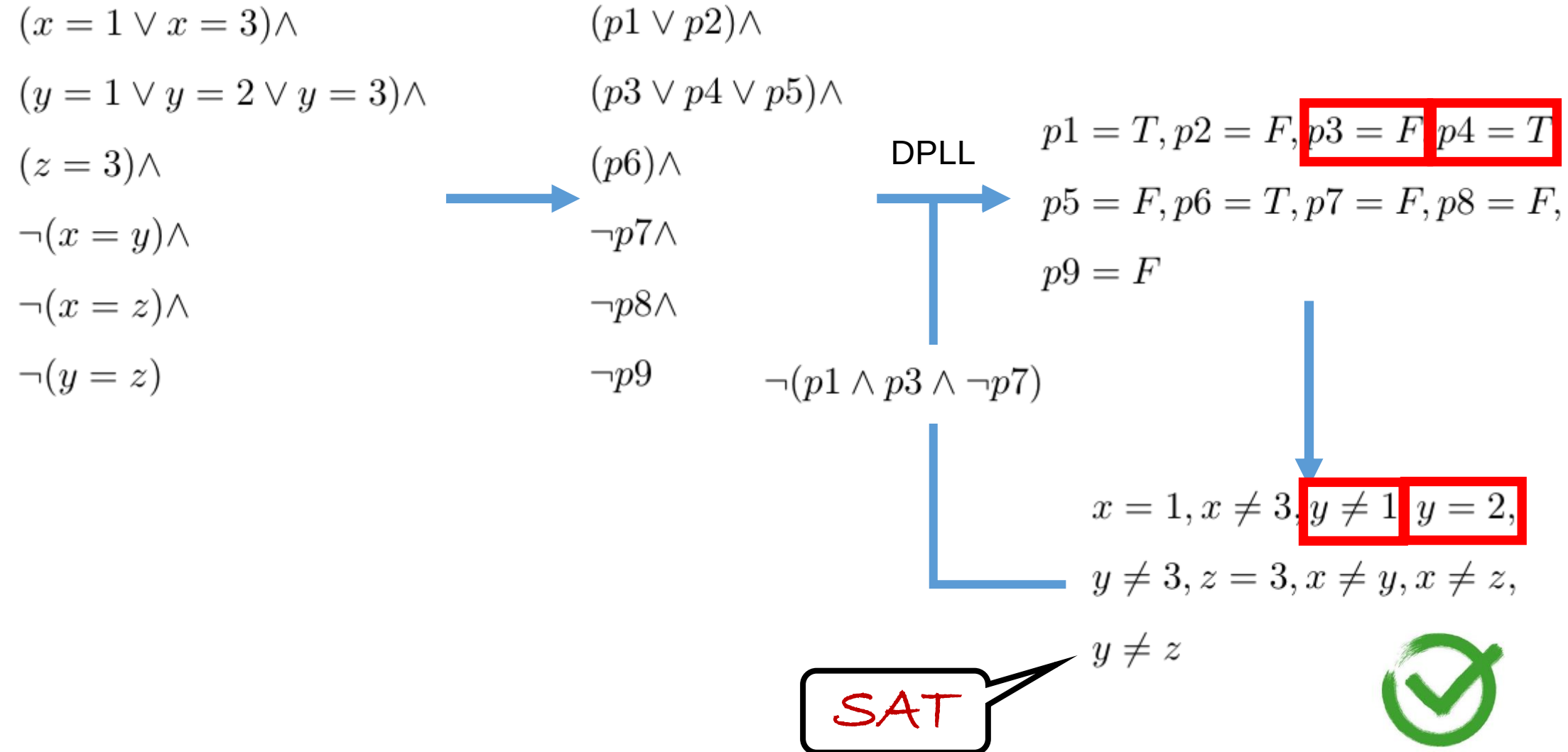
Lazy SMT Techniques



Lazy SMT Techniques



Lazy SMT Techniques



Lazy SMT Techniques

The main advantage of the lazy approach is that 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.

Z3

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

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

$(z = 3) \wedge$

$(x \neq y) \wedge$

$(x \neq z) \wedge$

$(y \neq z)$

individuals

Predicate logic
formulas

Solve it and
return sat

```
from z3 import *
x = Int('x')
y = Int('y')
z = Int('z')

s = Solver()
s.add(Or(x == 1, x == 3))
s.add(Or(y == 1, y == 2, y == 3))
s.add(z == 3)
s.add(x != y)
s.add(x != z)
s.add(y != z)

result = s.check()
print(result)
if result == sat:
    print(s.model())
```

Z3

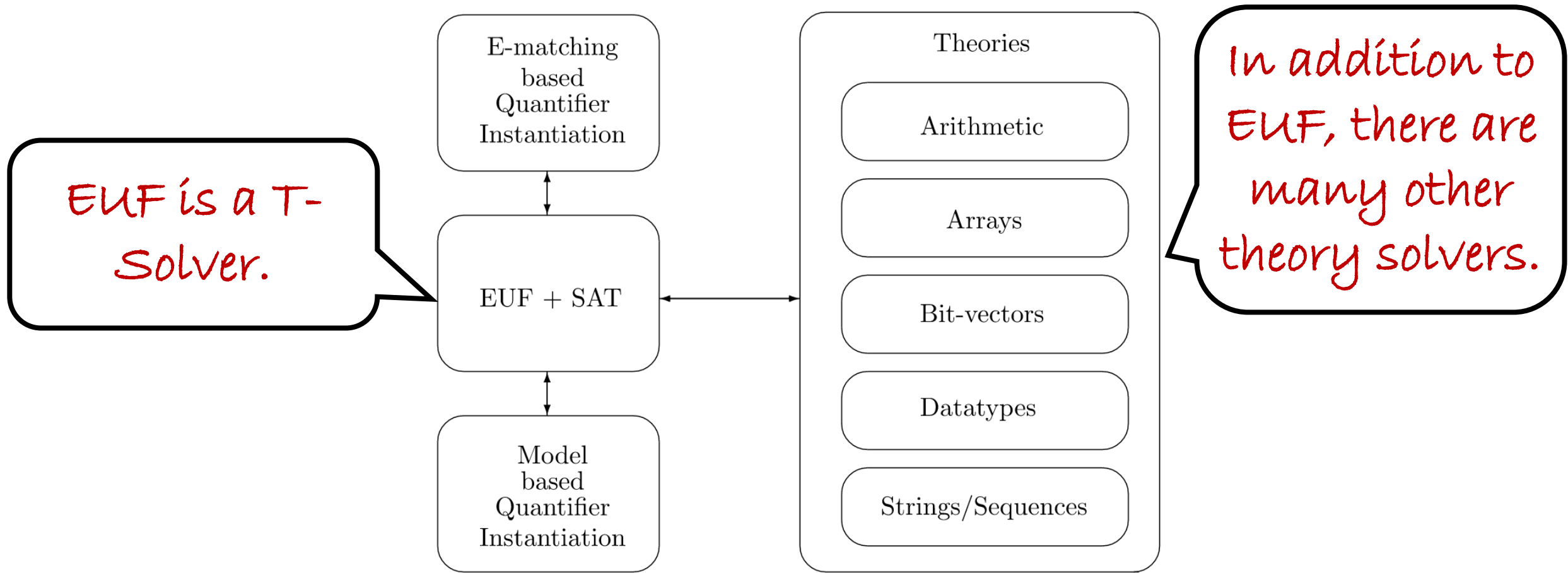
```
from z3 import *
x = Int('x')
y = Int('y')
z = Int('z')

s = Solver()
s.add(Or(x == 1, x == 3))
s.add(Or(y == 1, y == 2, y == 3))
s.add(z == 3)
s.add(x != y)
s.add(x != z)
s.add(y != z)

result = s.check()
print(result)
if result == sat:
    print(s.model())
```

```
> python3 example.py
```

```
sat
[y = 2, z = 3, x = 1]
```



Architecture of Z3's SMT Core solver

Theory Solver

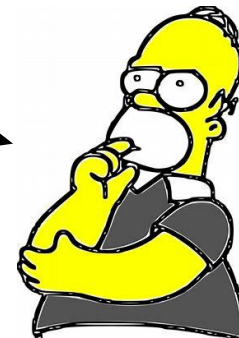
What does T-Solver looks Like?

EUUF

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

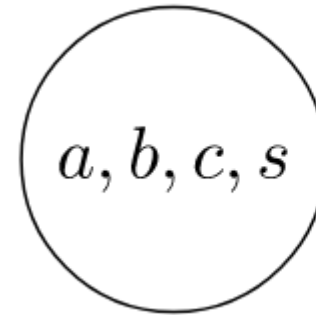
While formulas are not empty:

 remove $a=b$;

 merge(a, b);

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

$$b = s, d = t$$



EUF

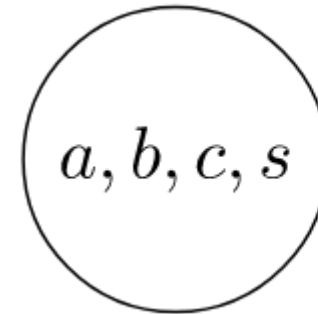
```
While formulas are not empty:  
    remove a=b;  
    merge(a, b);  
If find(a) = find(b) for some a≠b  
    return false;  
else  
    return true;
```

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

$$b = s, d = t$$



SAT



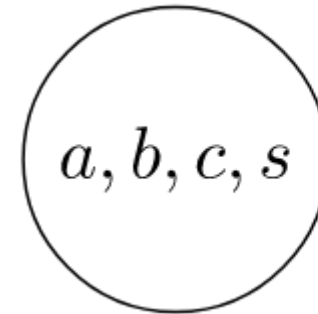
EUF

```
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;
```

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

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

SAT

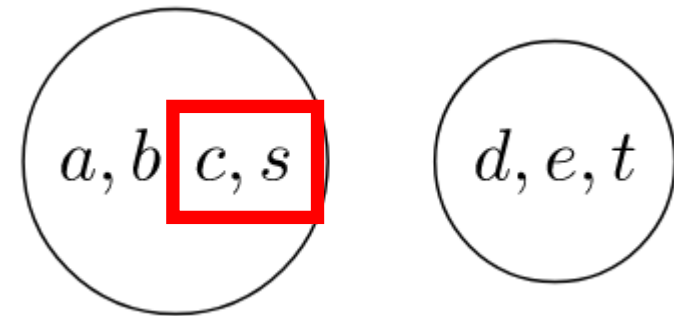


EUF

```
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;
```

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

UNSAT



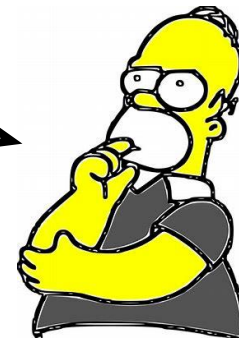
EUF

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

$$b = s, d = t$$

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

What if there are
uninterpreted functions?



EUf

THEOREM

Congruence rule:

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

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

$$b = s, d = t$$

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

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

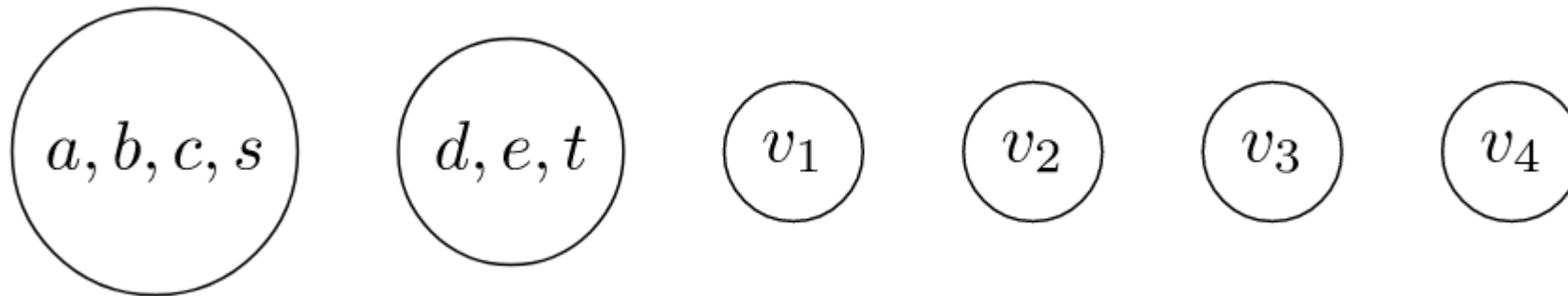
2) Working bottom-up, the congruence rule dictates how to merge groups

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

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

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

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



EUF

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

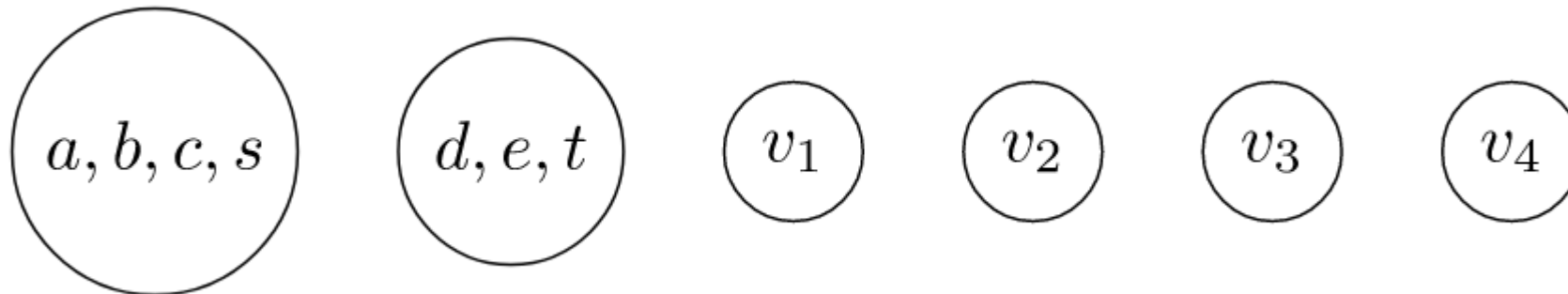
$$a = b, b = c, d = e$$

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

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

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

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



EUF

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

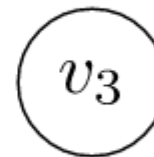
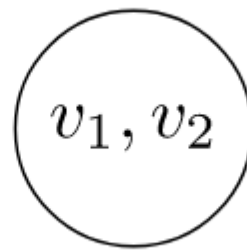
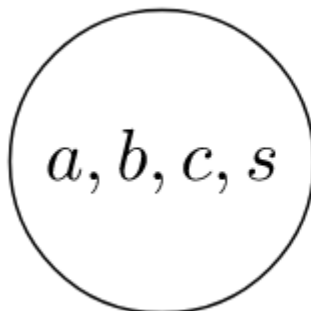
$$a = b, b = c, d = e$$

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

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

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

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



EUF

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

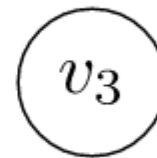
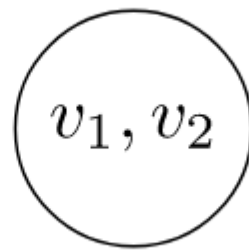
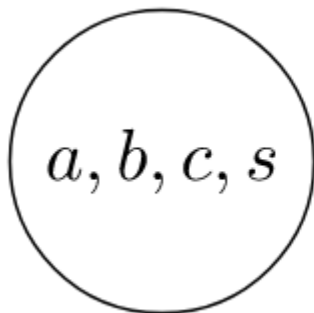
$$a = b, b = c, d = e$$

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

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

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

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



EUF

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

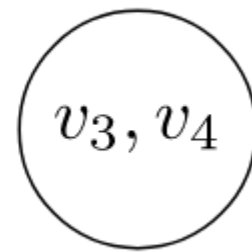
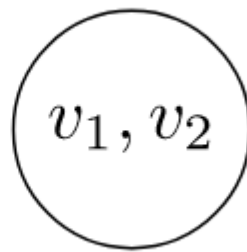
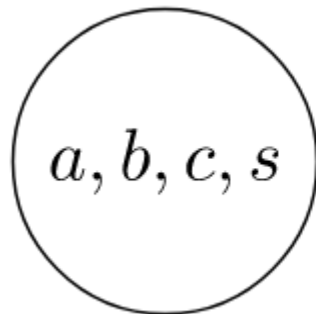
$$a = b, b = c, d = e$$

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

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

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

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



EUF

3) Check satisfiability like before

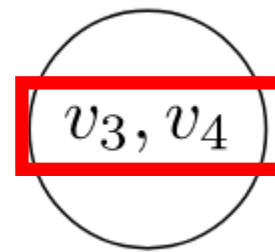
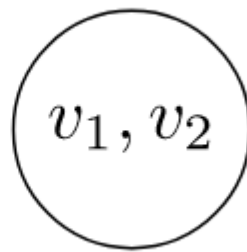
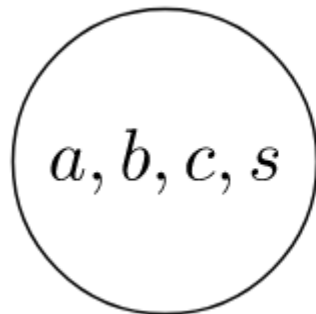
$$a = b, b = c, d = e$$

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

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

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

UNSAT



All in One Example

$$a = b \wedge$$

$$c = f(a) \wedge$$

$$d = f(b) \wedge$$

$$\neg(c = d)$$



$$p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4$$

DPLL



DPLL

$$p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4$$

$$\emptyset \parallel p_1, p_2, p_3, \overline{p_4} \Rightarrow (PureLiteral)$$

$$p_1 \parallel p_1, p_2, p_3, \overline{p_4} \Rightarrow (PureLiteral)$$

$$p_1 p_2 \parallel p_1, p_2, p_3, \overline{p_4} \Rightarrow (PureLiteral)$$

$$p_1 p_2 p_3 \parallel p_1, p_2, p_3, \overline{p_4} \Rightarrow (PureLiteral)$$

$$p_1 p_2 p_3, \overline{p_4} \parallel p_1, p_2, p_3, \overline{p_4}$$

All in One Example

$$a = b \wedge$$

$$c = f(a) \wedge$$

$$d = f(b) \wedge$$

$$\neg(c = d)$$



$$p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4$$

DPLL



$$p_1 = T, p_2 = T,$$

$$p_3 = T, p_4 = F$$



$$b = a,$$

$$c = f(a),$$

$$d = f(b),$$

$$c \neq d$$

EUF

$$a = b \wedge$$

$$c = f(a) \wedge$$

$$d = f(b) \wedge$$

$$\neg(c = d)$$



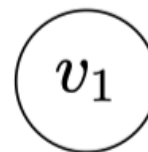
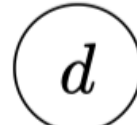
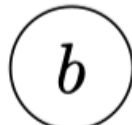
$$a = b, c = v_1, d = v_2, c \neq d,$$

$$v_1 := f(a), v_2 := f(b)$$

EUF

$$a = b, c = v_1, d = v_2, c \neq d,$$

$$v_1 := f(a), v_2 := f(b)$$



EUF

$$\boxed{a = b}, c = v_1, d = v_2, c \neq d,$$
$$v_1 := f(a), v_2 := f(b)$$

a, b

c

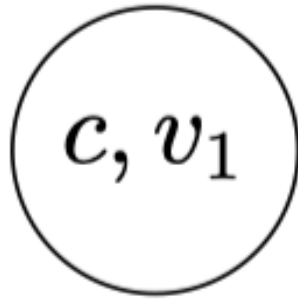
d

v_1

v_2

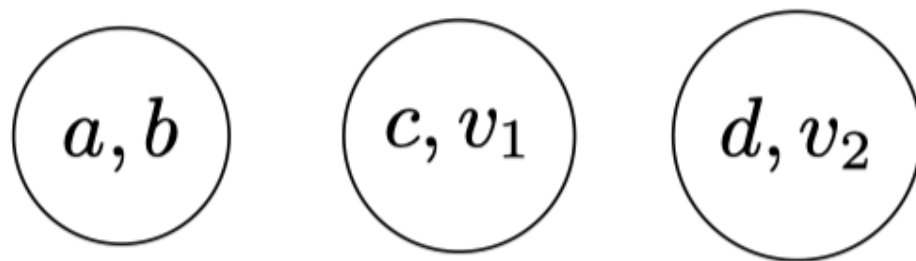
EUUF

$$a = b, c = v_1, d = v_2, c \neq d,$$
$$v_1 := f(a), v_2 := f(b)$$



EUUF

$$a = b, c = v_1, d = v_2, c \neq d,$$
$$v_1 := f(a), v_2 := f(b)$$

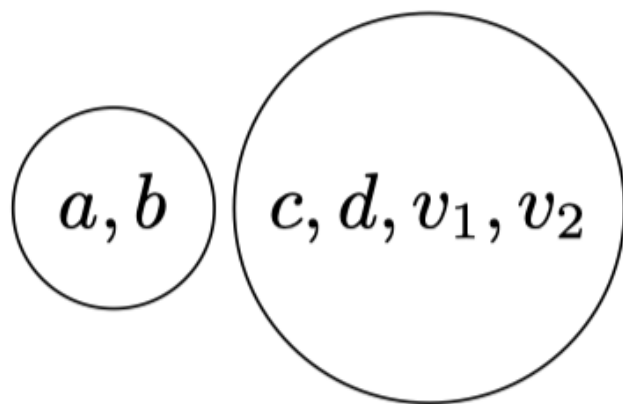


EU

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

$$a = b, c = v_1, d = v_2, c \neq d,$$

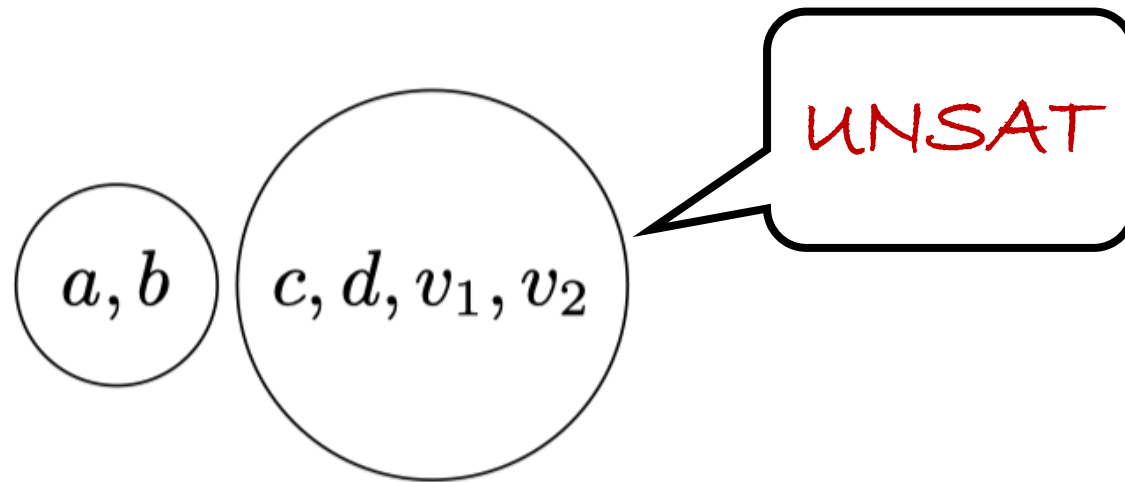
$$v_1 := f(a), v_2 := f(b)$$



EUF

$$a = b, c = v_1, d = v_2, c \neq d,$$

$$v_1 := f(a), v_2 := f(b)$$



ALL IN ONE

$$a = b \wedge$$

$$c = f(a) \wedge$$

$$d = f(b) \wedge$$

$$\neg(c = d)$$



$$p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4$$

DPLL



$$p_1 = T, p_2 = T,$$

$$p_3 = T, p_4 = F$$



$$b = a,$$

$$c = f(a),$$

$$d = f(b),$$

$$c \neq d$$

FALSE

$$\neg(p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4)$$

DPLL

$$p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4 \wedge (\neg p_1 \vee \neg p_2 \vee \neg p_3 \vee p_4)$$

$$\emptyset \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (Decide)$$

$$p_1^d \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (Decide)$$

$$p_1^d p_2^d \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (Decide)$$

$$p_1^d p_2^d p_3^d \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (UnitProp)$$

$$p_1^d p_2^d p_3^d p_4 \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (Backtrack)$$

$$p_1^d p_2^d \overline{p_3} \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (Backtrack)$$

$$p_1^d \overline{p_2} \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (Backtrack)$$

$$\overline{p_1} \parallel p_1, p_2, p_3, \overline{p_4}, \overline{p_1} \vee \overline{p_2} \vee \overline{p_3} \vee p_4 \Rightarrow (Fail)$$

ALL IN ONE

$$a = b \wedge$$

$$c = f(a) \wedge$$

$$d = f(b) \wedge$$

$$\neg(c = d)$$



$$p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4$$

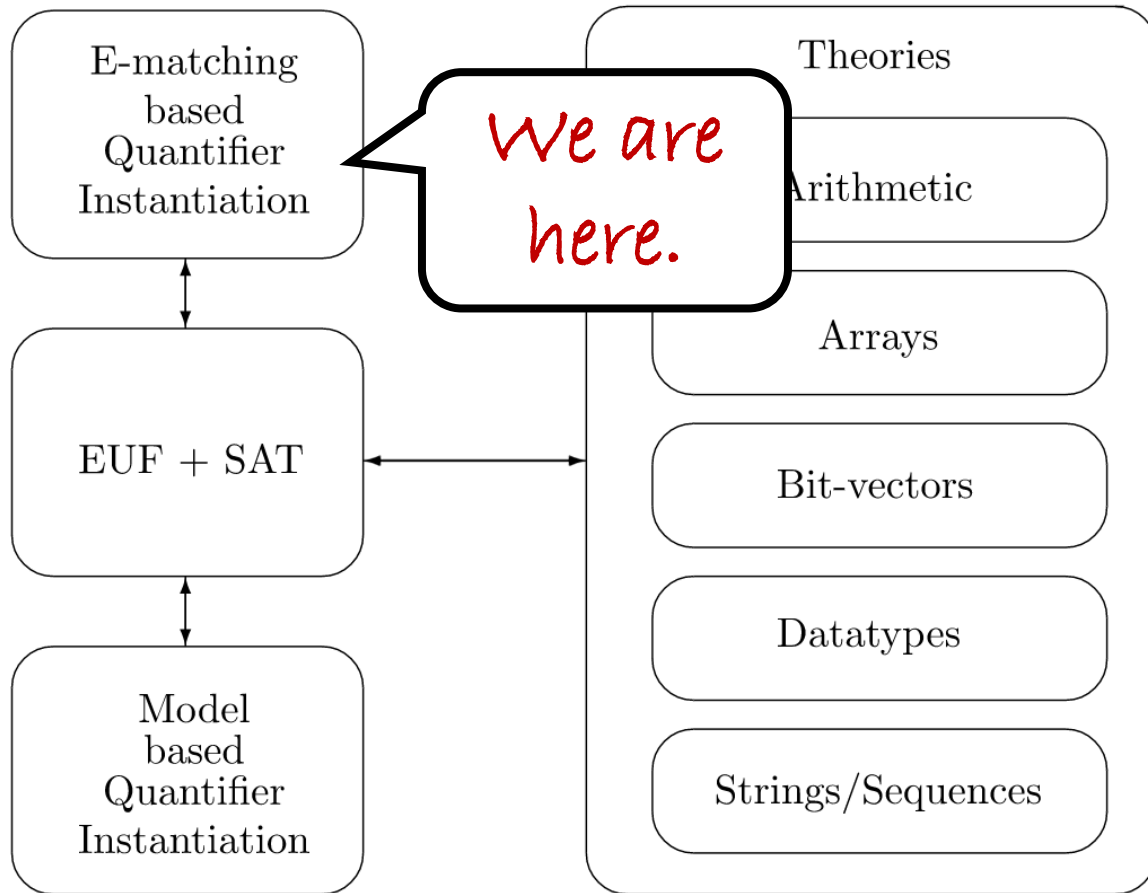
DPLL



$$\neg(p_1 \wedge p_2 \wedge p_3 \wedge \neg p_4)$$

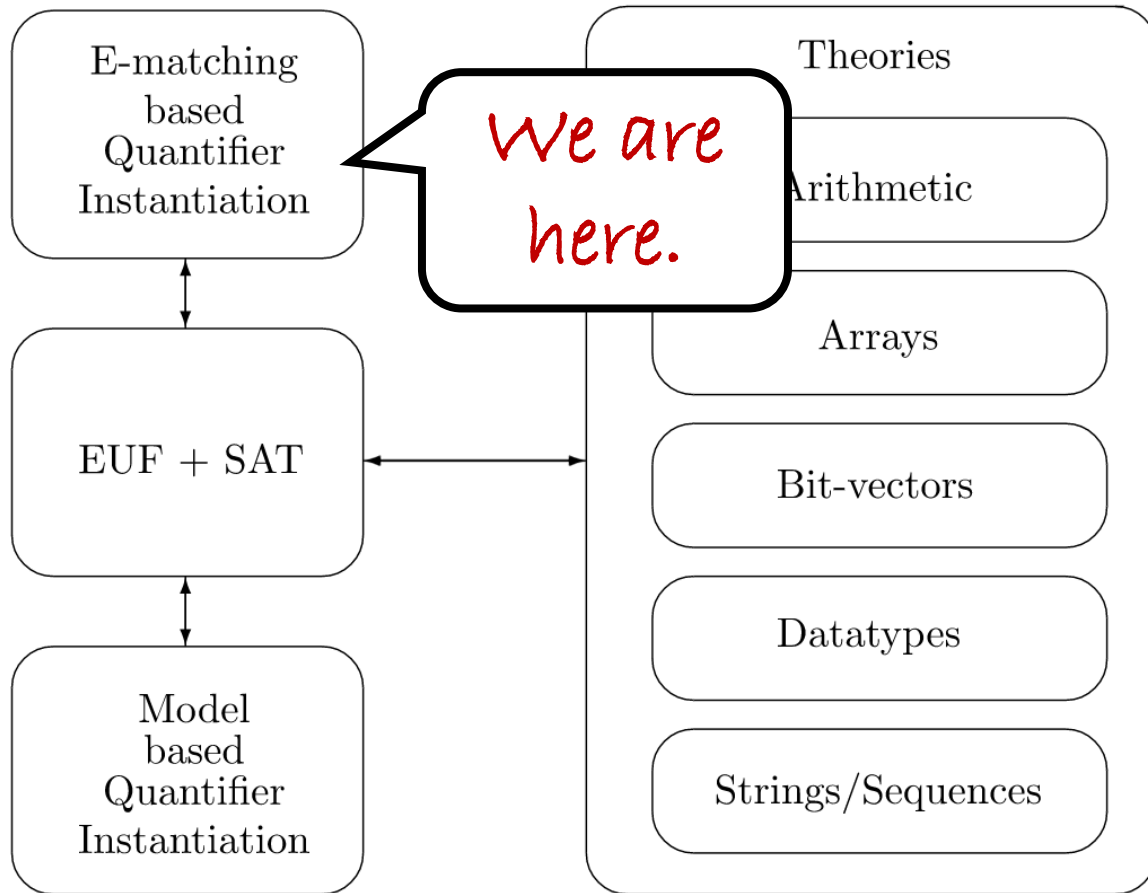


UNSAT



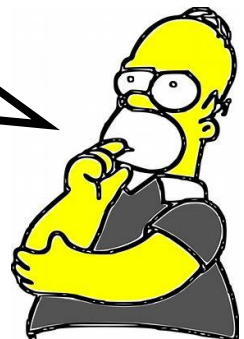
Architecture of Z3's SMT Core solver





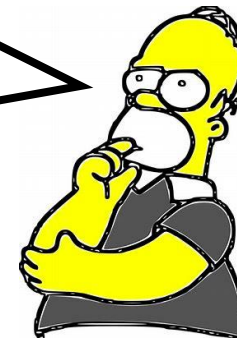
Architecture of Z3's SMT Core solver

It is very difficult because domain of discourse is infinite.



Quantifier

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



Quantifier

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

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

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

*There is no “not”
before quantifiers.*

A Quick Recap

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



y depends on x

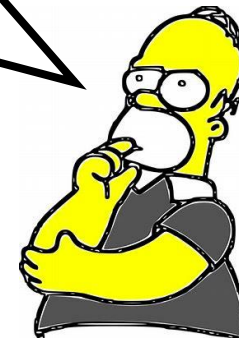
$$(\forall x)(\exists z)S(a, b, x, f(x), z)$$



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

z depends on x

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



Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a

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

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a

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



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

Quantifier

Now there are only universal quantifiers.

Intuitive approach: replace x with a

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

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

Theory Solvers

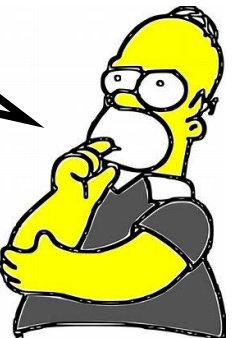
UNSAT

Thanks & Questions

Problem from Last Class

$$a = c \wedge a = f(b) \wedge d = f(e) \wedge b \neq e \wedge c = f(e)$$

Solve this problem



Step by step

$$a = c \wedge$$

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

$$d = f(e) \wedge$$

$$\neg(b = e) \wedge$$

$$c = f(e)$$



$$p_1 \wedge p_2 \wedge p_3 \\ \wedge \neg p_4 \wedge p_5$$

DPLL


$$p_1 = T, p_2 = T, \\ p_3 = T, p_4 = F, \\ p_5 = T$$



$$a = c,$$

$$a = f(b),$$

$$d = f(e),$$

$$b \neq e,$$

$$c = f(e)$$

EUUF

$$a = c,$$

$$a = f(b),$$

$$d = f(e),$$

$$b \neq e,$$

$$c = f(e)$$



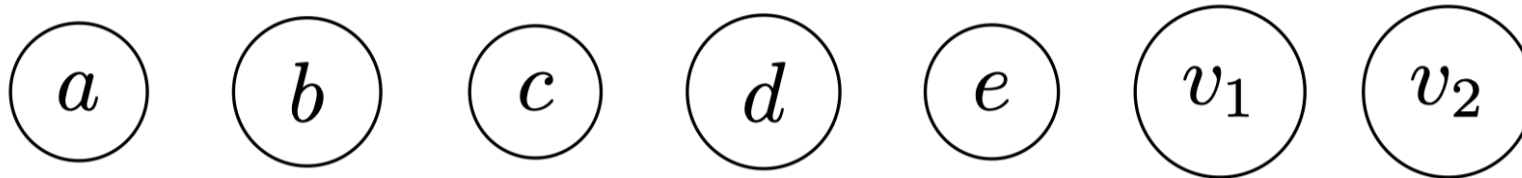
$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$

EUF

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

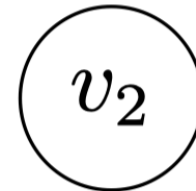
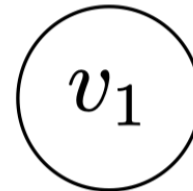
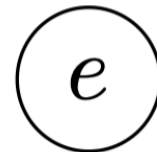
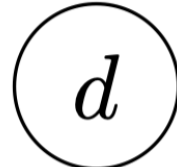
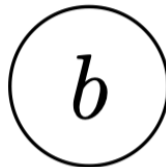
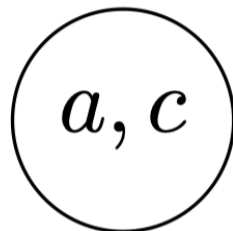
$$v_1 := f(b), v_2 := f(e)$$



EUF

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

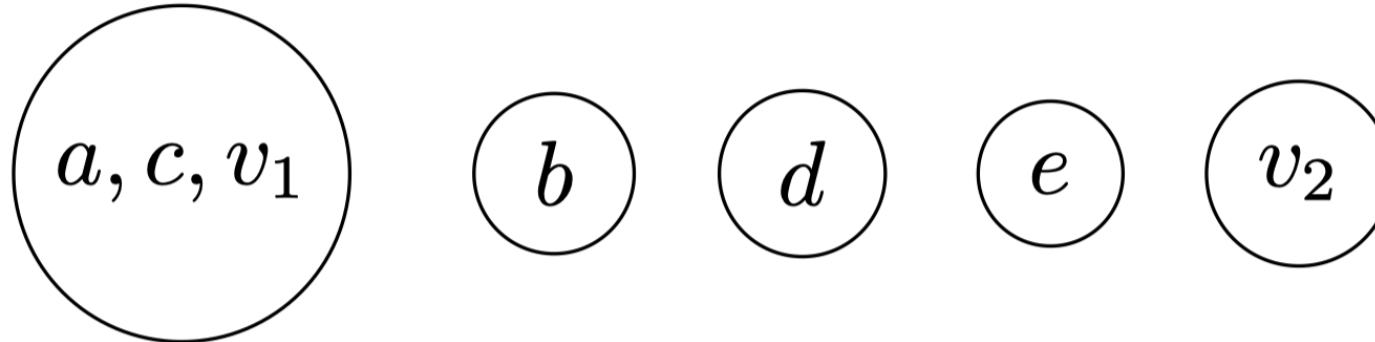
$$v_1 := f(b), v_2 := f(e)$$



EUF

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

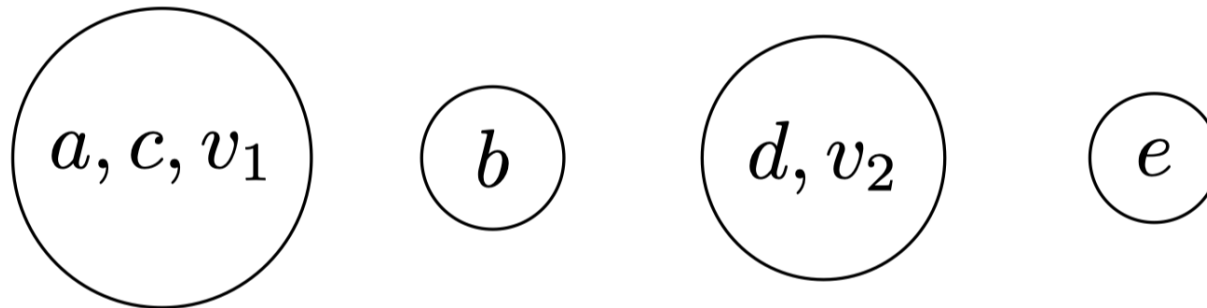
$$v_1 := f(b), v_2 := f(e)$$



EUF

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

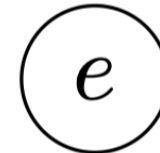
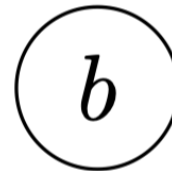
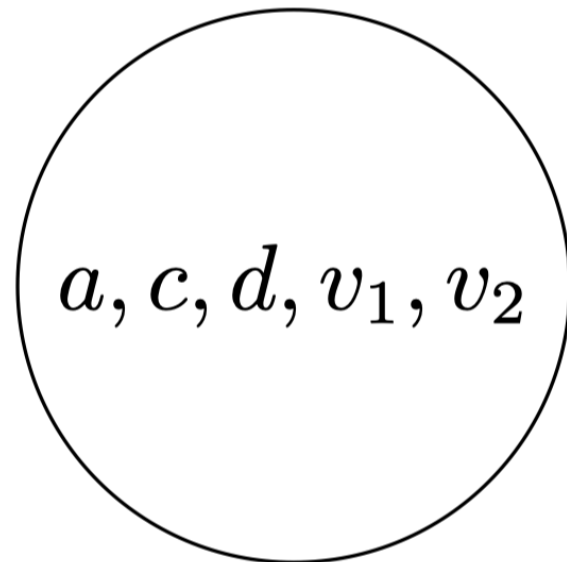
$$v_1 := f(b), v_2 := f(e)$$



EUF

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$



We need to give an assignment



EUF

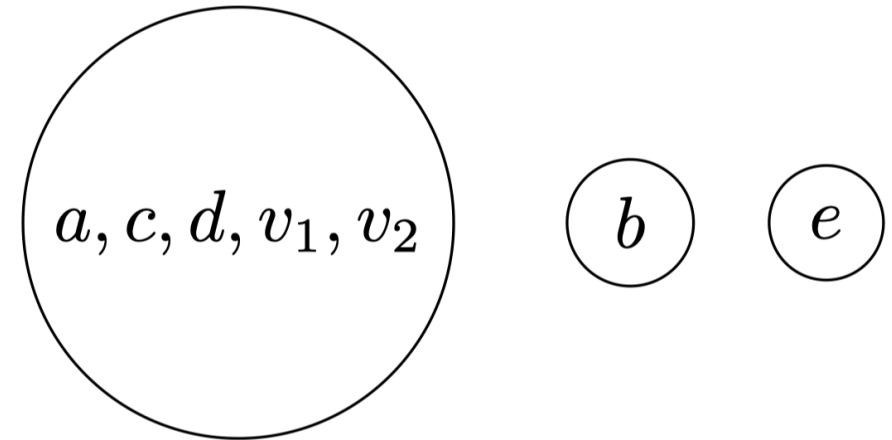
Assuming that the domain of a, b, c, d, e is S .

S can be expressed as a set $\{S_0, S_1, \dots, S_n\}$.

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$

We need to assign
 S_i to the circles



EUUF

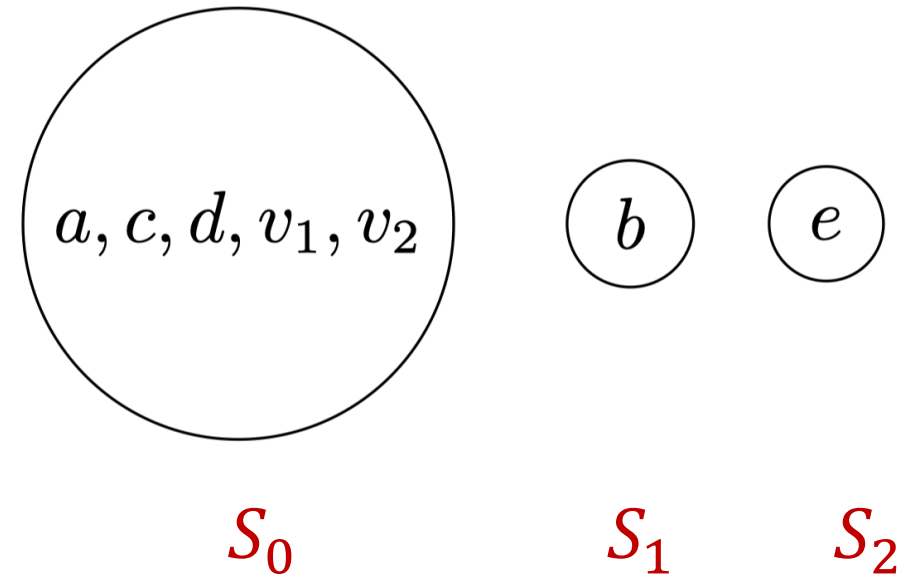
For c in circles:

choose a **new** s from S

assign s to c

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$



EUF

Assuming that the domain of a, b, c, d, e is S .

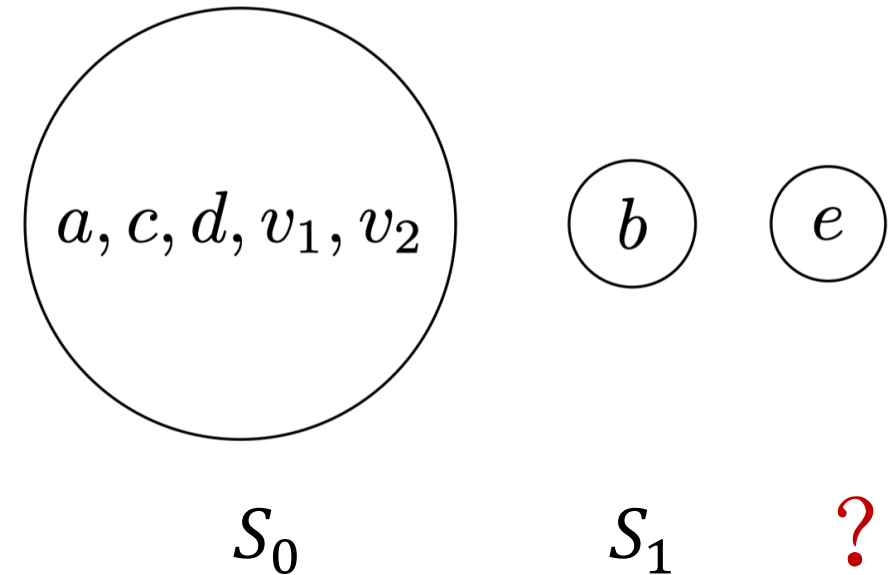
S can be expressed as a set $\{S_0, S_1\}$.

What if S has no enough values to be assigned? E.g., S contains "T" and "F"



$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$



EUUF

For c in circles:

choose a **non-conflict** s from S

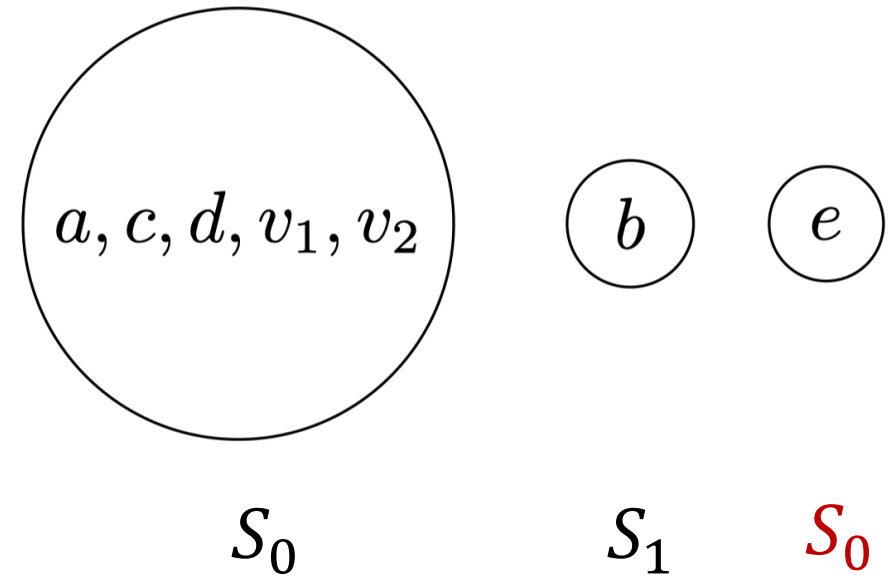
assign s to c

if no available s

return **false**

return **true**

$$a = c, a = v_1, d = v_2, \boxed{b \neq e}, c = v_2$$
$$v_1 := f(b), v_2 := f(e)$$



EUf

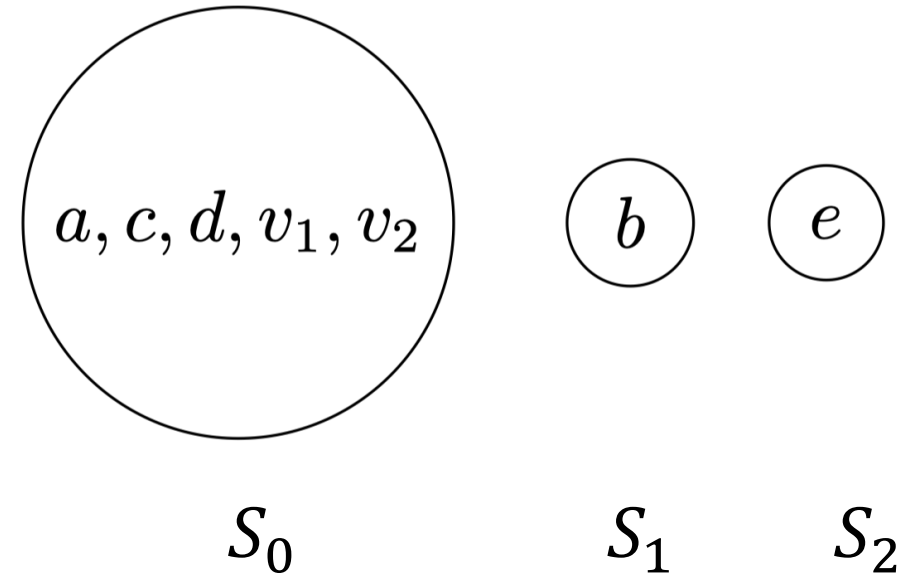
Assuming that the domain of a, b, c, d, e is S .

S can be expressed as a set $\{S_0, S_1, \dots, S_n\}$.

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$

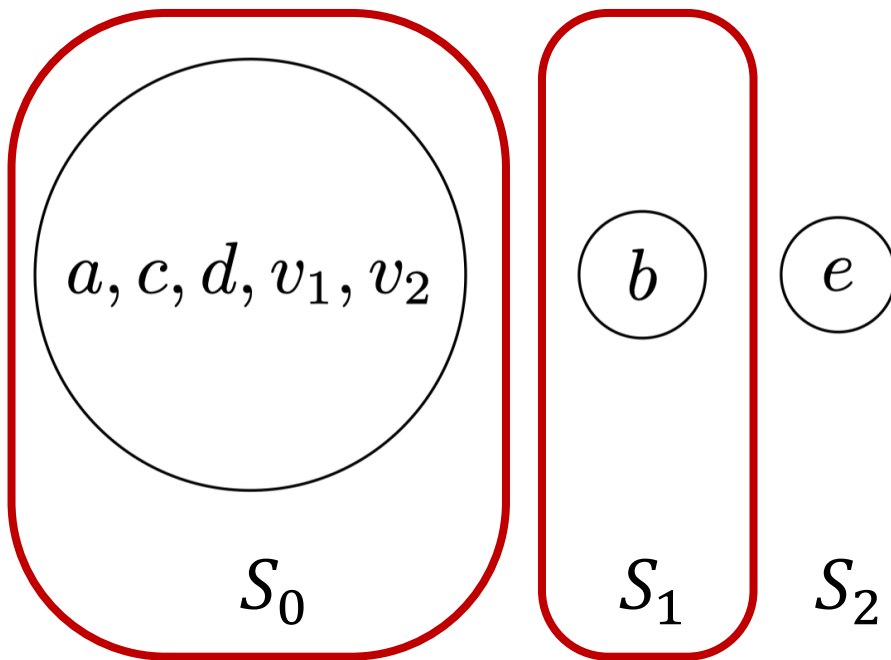
How to assign
functions?



EUF

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$



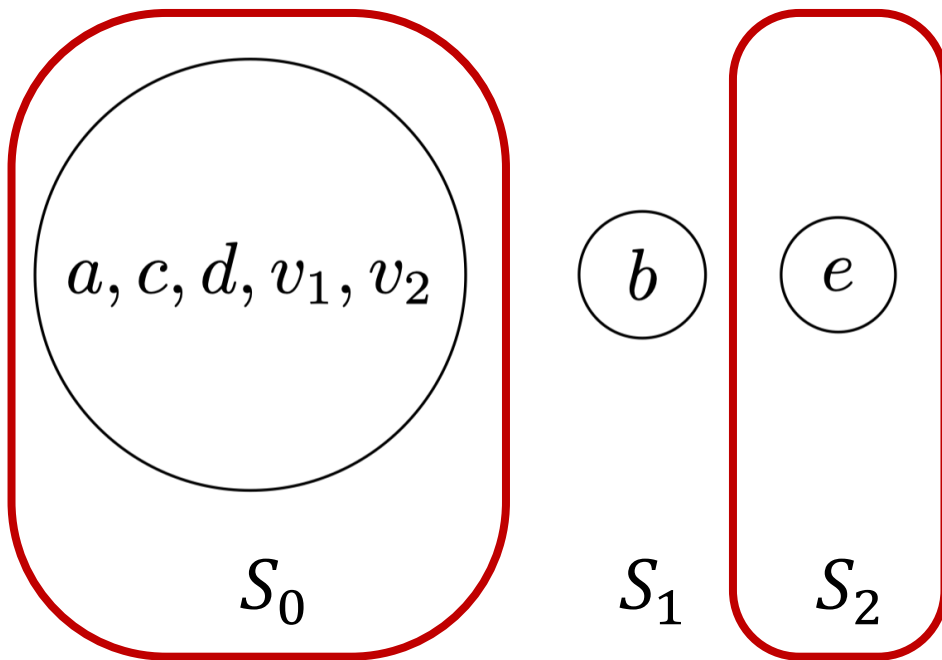
$$f(S_1) = S_0, f(S_2) = S_0$$

$$f(x) = S_0 \text{ when } x \text{ is not } S_1 \text{ or } S_2$$

EUf

$$a = c, a = v_1, d = v_2, b \neq e, c = v_2$$

$$v_1 := f(b), v_2 := f(e)$$



$$f(S_1) = S_0, f(S_2) = S_0$$

$$f(x) = S_0 \text{ when } x \text{ is not } S_1 \text{ or } S_2$$

Code in Z3

```
from z3 import *

// Int type
sort = IntSort()

a = Const('a', sort)
b = Const('b', sort)
c = Const('c', sort)
d = Const('d', sort)
e = Const('e', sort)
f = Function('f', sort, sort)

solve(a==c, a == f(b), d ==f(e),
      b != e, c == f(e))
```



```
[e = 0,
 b = 1,
 d = 2,
 a = 2,
 c = 2,
 f = [else -> 2]]
```