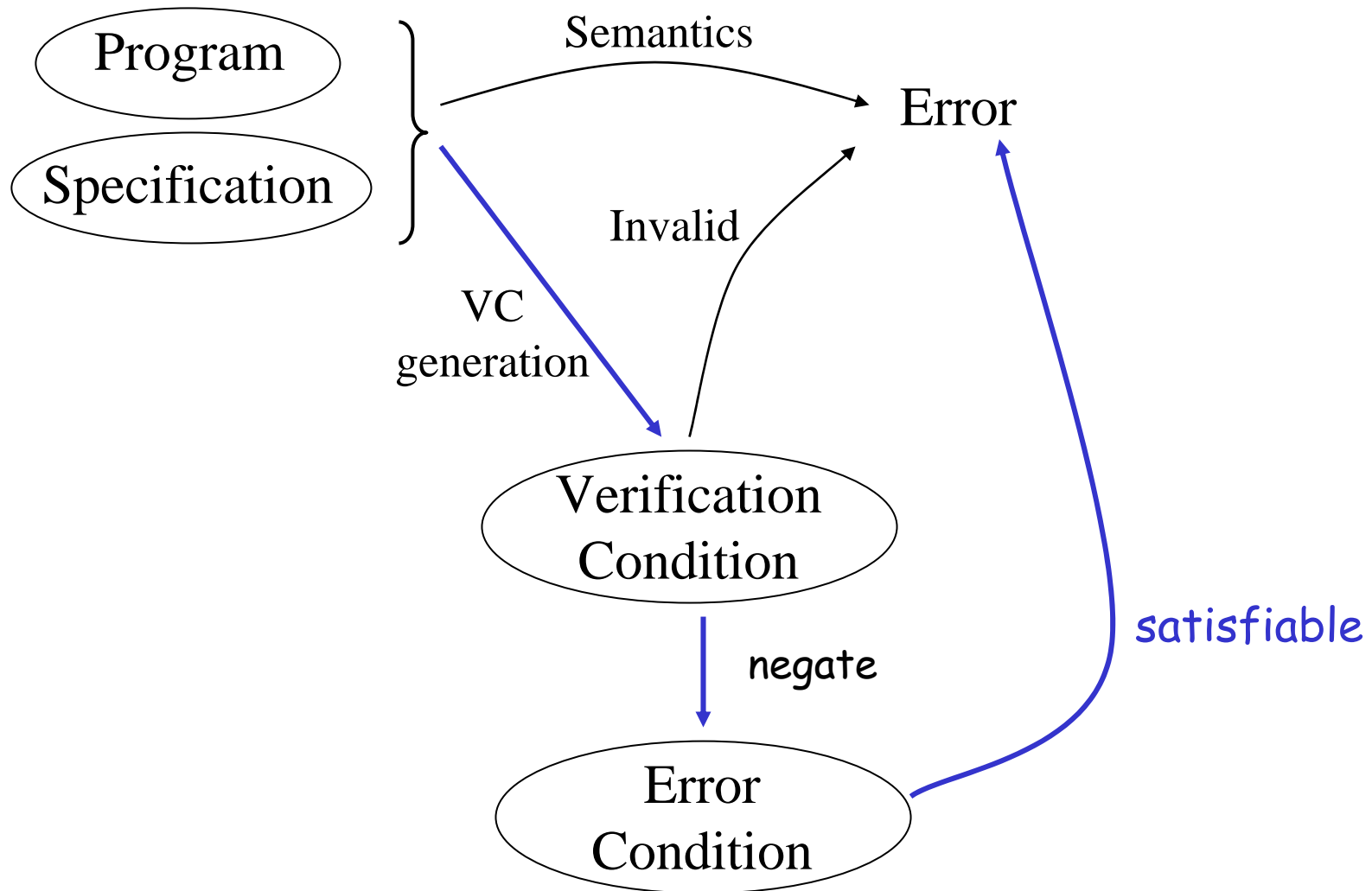


# Satisfying Error Conditions 2

## Lecture 14

# Review: Where Are We?

---



# Satisfiability of ECs: Example

---

- EC

$$(a=b) \wedge (\neg(f(a)=f(b)) \vee b=c) \wedge \neg(f(a)=f(c))$$

- SAT problem

$$\{a=b\} \wedge (\neg\{f(a)=f(b)\} \vee \{b=c\}) \wedge \neg\{f(a)=f(c)\}$$

$$\{a=b\} \wedge (\neg\{f(a)=f(b)\}) \vee \{b=c\} \wedge \neg\{f(a)=f(c)\}$$

$\{a=b\}$	$\{f(a)=f(b)\}$	$\{b=c\}$	$\{f(a)=f(c)\}$	Sat?
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

$$a=b$$

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

$$\wedge b \neq c$$

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

$$a=b$$

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

$$\wedge b=c$$

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

$$a=b$$

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

$$\wedge b=c$$

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

$$\{a=b\} \wedge (\neg\{f(a)=f(b)\} \vee \{b=c\}) \wedge \neg\{f(a)=f(c)\}$$

$\{a=b\}$	$\{f(a)=f(b)\}$	$\{b=c\}$	$\{f(a)=f(c)\}$	Sat?
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	<del>X</del> 0
1	0	0	1	0
1	0	1	0	<del>X</del> 0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

$$\begin{aligned}
 & a=b \\
 & \wedge f(a) \neq f(b) \\
 & \wedge b \neq c \\
 & \wedge f(a) \neq f(c)
 \end{aligned}$$

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

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

Explicated tautology  
removes many other  
truth assignments

# Implementing $\text{checkSatLits}(L_1, \dots, L_k)$

---

- A theory consists of:
  - Syntax: function, constant, and predicate symbols
  - Semantics
- The Satisfiability Problem: Decide whether a conjunction of literals in the theory is satisfiable

# Examples of Theories. Equality.

---

- Syntax  $=, \neq, f, g, \dots$
- Semantics: as expected

$$\frac{}{E = E} \quad \frac{E_2 = E_1}{E_1 = E_2} \quad \frac{E_1 = E_2 \quad E_2 = E_3}{E_1 = E_3} \quad \frac{E_1 = E_2}{f(E_1) = f(E_2)}$$

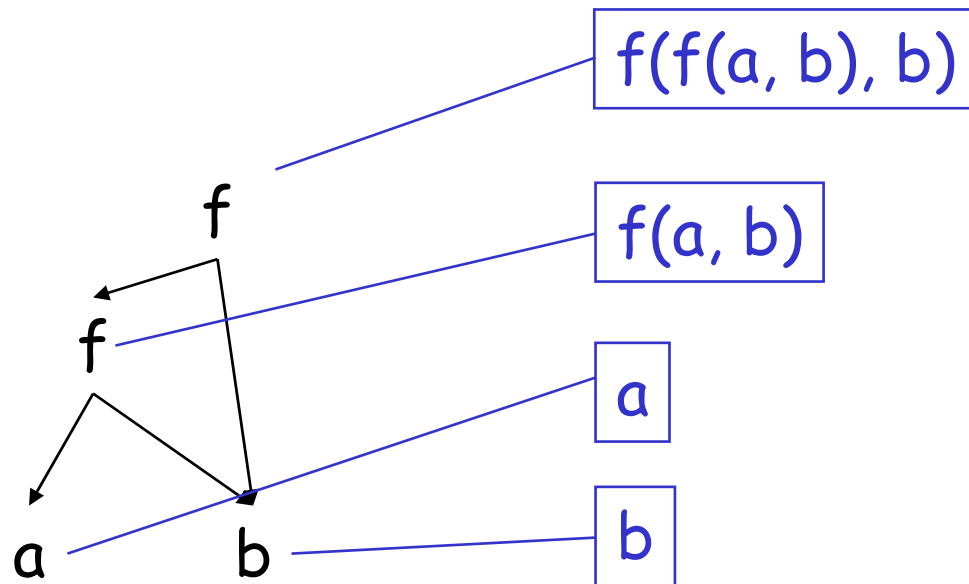
- Example

$$g(g(g(x))) = x \wedge g(g(g(g(g(x)))))) = x \wedge g(x) \neq x$$

# Satisfiability Algorithm for Equality (1)

---

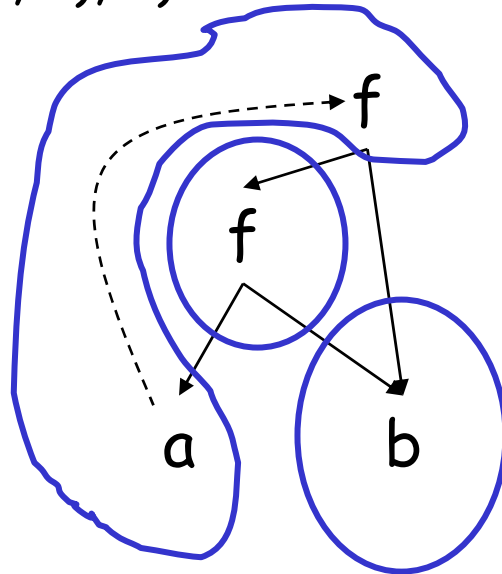
- We represent terms as DAGs
  - Share common subexpressions
  - E.g.  $f(f(a, b), b)$ :



# Satisfiability Algorithm for Equality (2)

---

- Terms are in *equivalence classes*
  - each class has representative element
  - dotted arrow points towards representative element
    - unless term is representative element of its equivalence class
- E.g.  $f(f(a, b), b) = a$

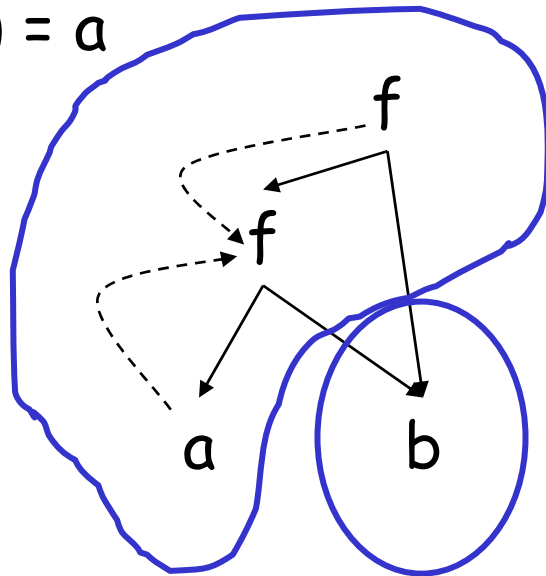


# Satisfiability Algorithm for Equality (3)

---

- $t^*$  is representative element for  $t$
- For all nodes  $t = f(t_1, \dots, t_n)$  and  $s = f(s_1, \dots, s_n)$ 
  - If  $t_i^* = s_i^*$  for all  $i = 1..n$  (find)
  - We add dotted arrow between  $t^*$  and  $s^*$

- E.g.  $f(a,b) = a$



# Satisfiability Procedure for Equality

---

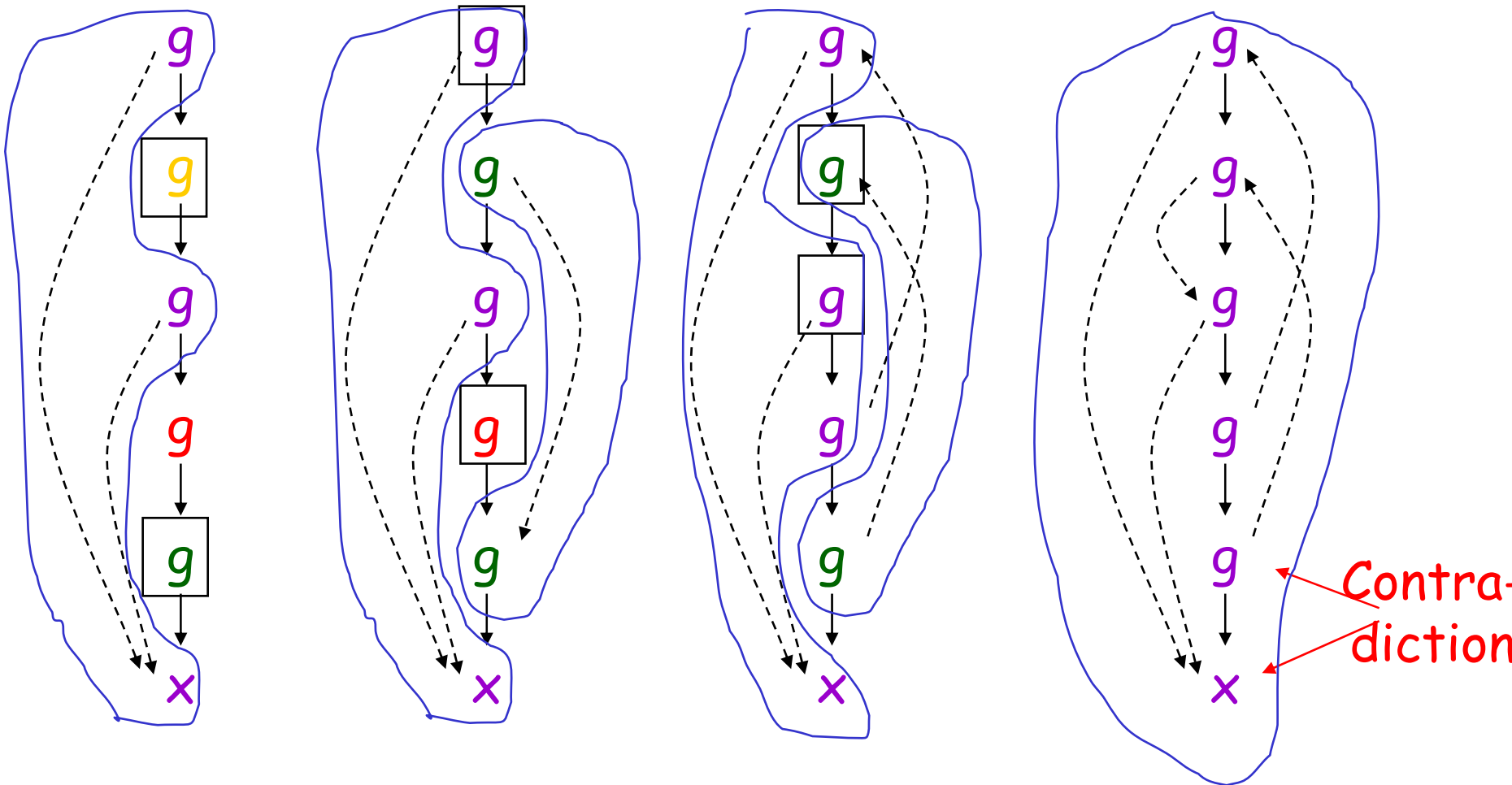
1. Given  $F = \bigwedge_i t_i = t'_i \wedge \bigwedge_j u_j \neq u'_j$
2. Represent all terms in the same DAG
3. Add dotted edges for  $t_i = t'_i$
4. Construct the congruence closure of those edges
5. Check that for all  $j$  we have  $u_j^* \neq u'_j^*$

## Theorem:

$F$  is satisfiable if and only if for all  $j$   $u_j^* \neq u'_j^*$

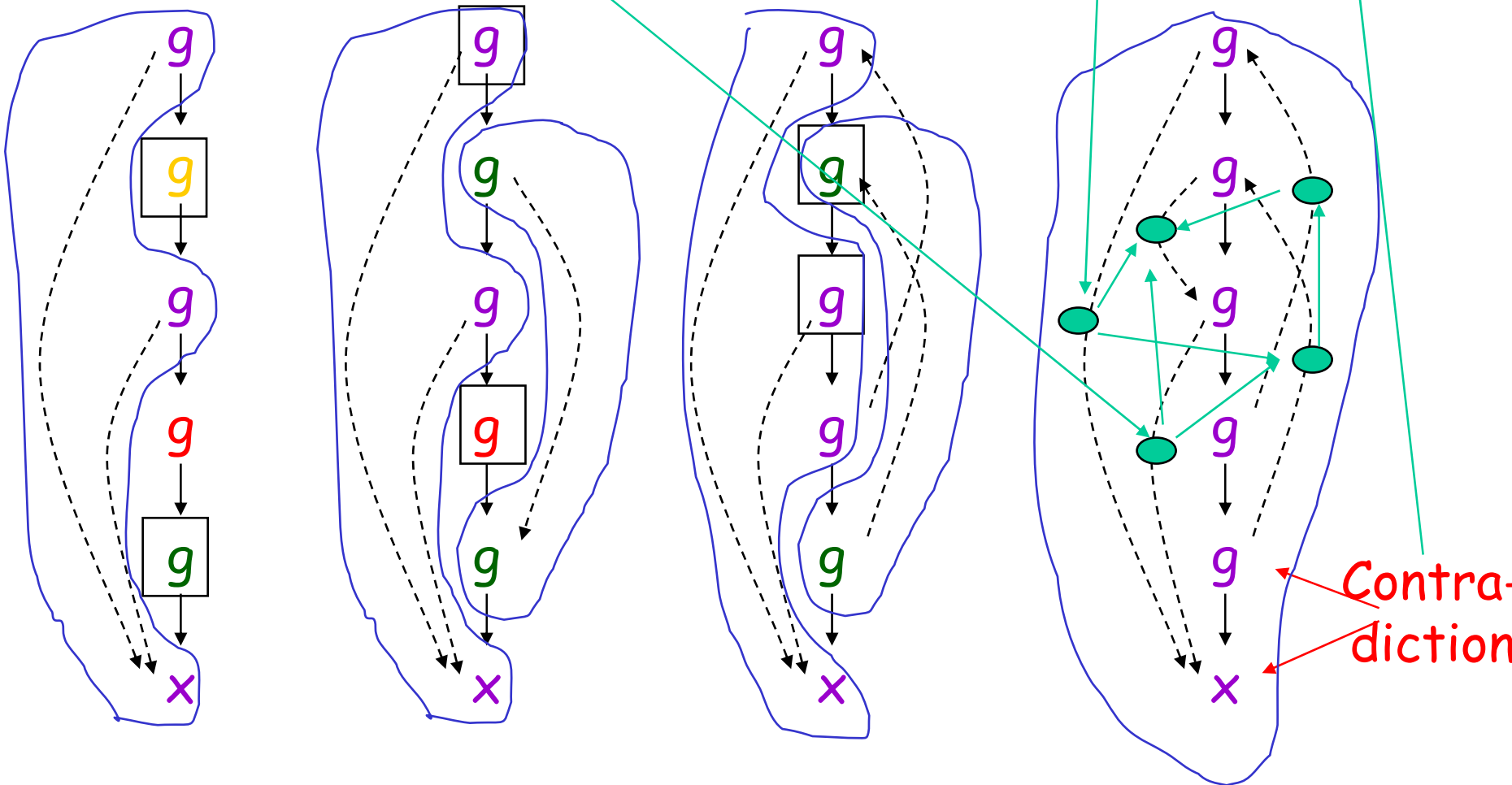
# Example with Congruence Closure

- Consider:  $g(g(g(x)) = x \wedge g(g(g(g(g(x)))) = x \wedge g(x) \neq x$



# Example with Congruence Closure

- Consider:  $g(g(g(x)) = x \wedge g(g(g(g(g(x)))) = x \wedge g(x) \neq x$



# Presburger Arithmetic

---

- Syntax:  $+, -, =, >, 0, 1, -1, 2, -2, \dots$
- Semantics: as expected
- The most useful in program verification after equality
  - checking array bounds, etc
- Example:  $y > 2x + 1 \wedge y + x > 1 \wedge y < 0$

# Difference Constraints

---

- A special case of linear arithmetic
- All constraints of the form:  
 $x + c \leq y$
- $c$  is a constant
- Special variable  $z$  representing 0
- Can express many common linear constraints
  
- How to check satisfiability?
- Hint: Think of a directed graph
  - with a node for each variable
  - represent constraint in each graph
  - and think about graph algorithms

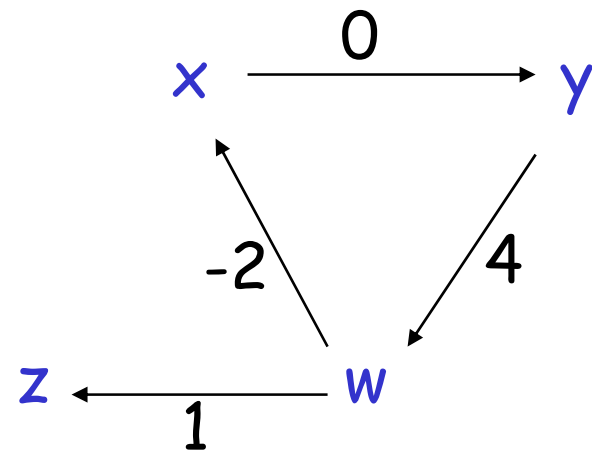
# Difference Constraints (cont)

---

- Hint: Think of a directed graph
  - with a node for each variable
  - represent constraint in each graph
  - and think about graph algorithms

- Example

- $x \leq y$
- $y+4 \leq w$
- $w-2 \leq x$
- $w+1 \leq z$



- What is explanation?

# Difference Constraints

---

## Theorem:

*A set of difference constraints is satisfiable  
in and only if  
there is no positive weight cycle in the graph*

- Can be solved in  $O(n^2)$
- Algorithm is complete !
- Was used successfully in array-bounds checking elimination and induction variable discovery

# Extensions of Difference Constraints

---

- Shostak extended the algorithm to  $ax + by \geq c$
- Construct a graph as before
  - One node for each variable
  - One undirected edge for each constraint
- An admissible loop in this graph
  - is a loop in which any two adjacent edges
    - $ax + by \geq c$
    - $dy + ez \geq f$
  - have  $\text{sgn}(b) \neq \text{sgn}(d)$
- Can cancel all variables in loop to get a variable-free constraint
- Consider:  $3x \geq 2y \wedge 3y \geq 4 \wedge 3 \geq 2x \wedge x \geq w$

# How Complete are These Procedures?

---

- Consider:  $3x \geq 2y \wedge 3y \geq 4 \wedge 3 \geq 2x \wedge x \geq w$
- Is it satisfiable?
- What is solution?
- Is there a solution in  $Z$ ?
  
- The unsat procedure is sound:  $\text{unsat } Q \Rightarrow \text{unsat } Z$
- But it is incomplete !
  
- Not a problem in practice
- Goes away with tricks like:
- Transform " $ax \geq b$ " into " $x \geq \lceil b/a \rceil$ "

# Arithmetic. Discussion

---

- There are many satisfiability algorithms for  $\mathbb{Q}$ 
  - even for the general case (e.g. Simplex)
  - polynomial time
  - complete
- If we add the requirement that solutions are in  $\mathbb{Z}$ 
  - then the problem is NP-complete
  - but  $\mathbb{Z}$  can be handled well with heuristics
- No practical satisfiability procedures for  $(\mathbb{Q}, \times)$
- The satisfiability of  $(\mathbb{Z}, \times)$  is only semi-decidable

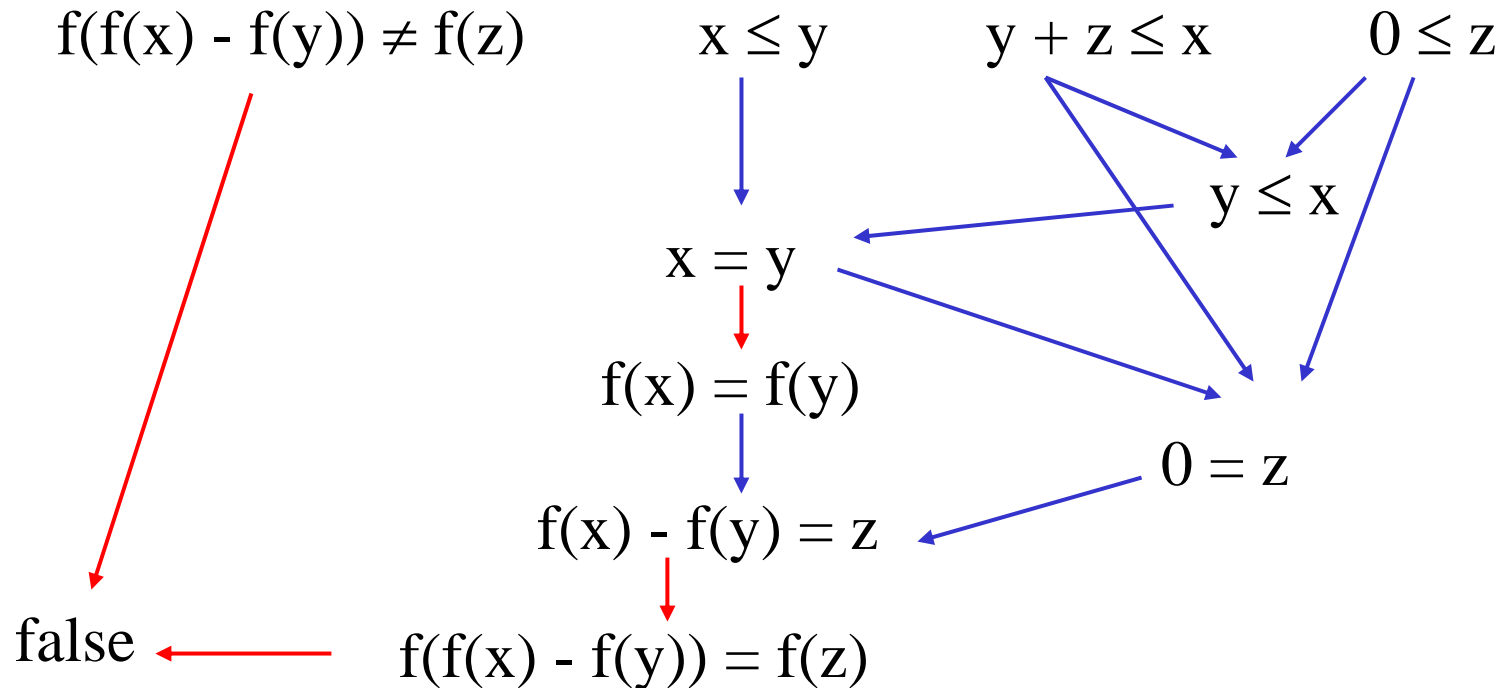
# Combining Satisfiability Procedures

---

- We have developed satisfiability procedures for two theories
  - equality
  - arithmetic (Shostak)
- Can we combine satisfiability procedures to handle error conditions with both equality and arithmetic?

# Combining Satisfiability Procedures. Example

- Consider **equality** and **arithmetic**



# Combining Satisfiability Procedures

---

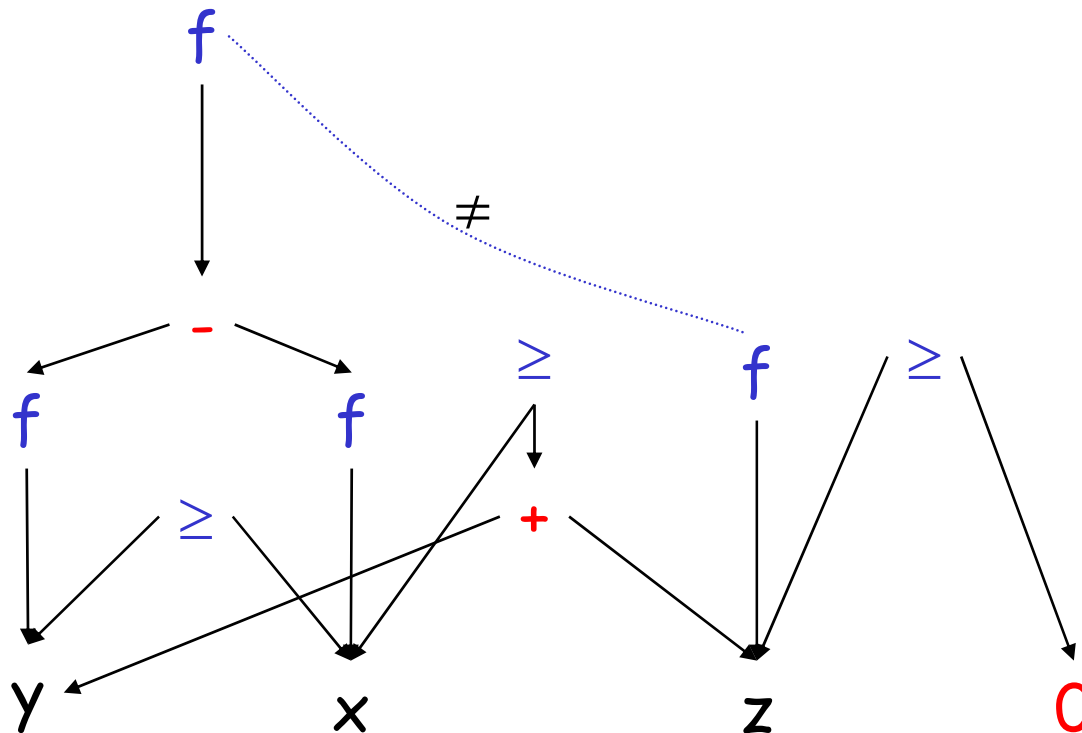
- Combining satisfiability procedures is non trivial
  - Equality was solved by Ackerman in 1924
  - arithmetic by Fourier even before
  - but equality and arithmetic only in 1979 !
- Yet in any single verification problem we will have literals from several theories:
  - Equality, arithmetic, lists, ...
- When and how can we combine separate satisfiability procedures?

# Nelson-Oppen Method (1)

---

1. Represent all conjuncts in the same DAG

$$f(f(x) - f(y)) \neq f(z) \wedge y \geq x \wedge x \geq y + z \wedge z \geq 0$$

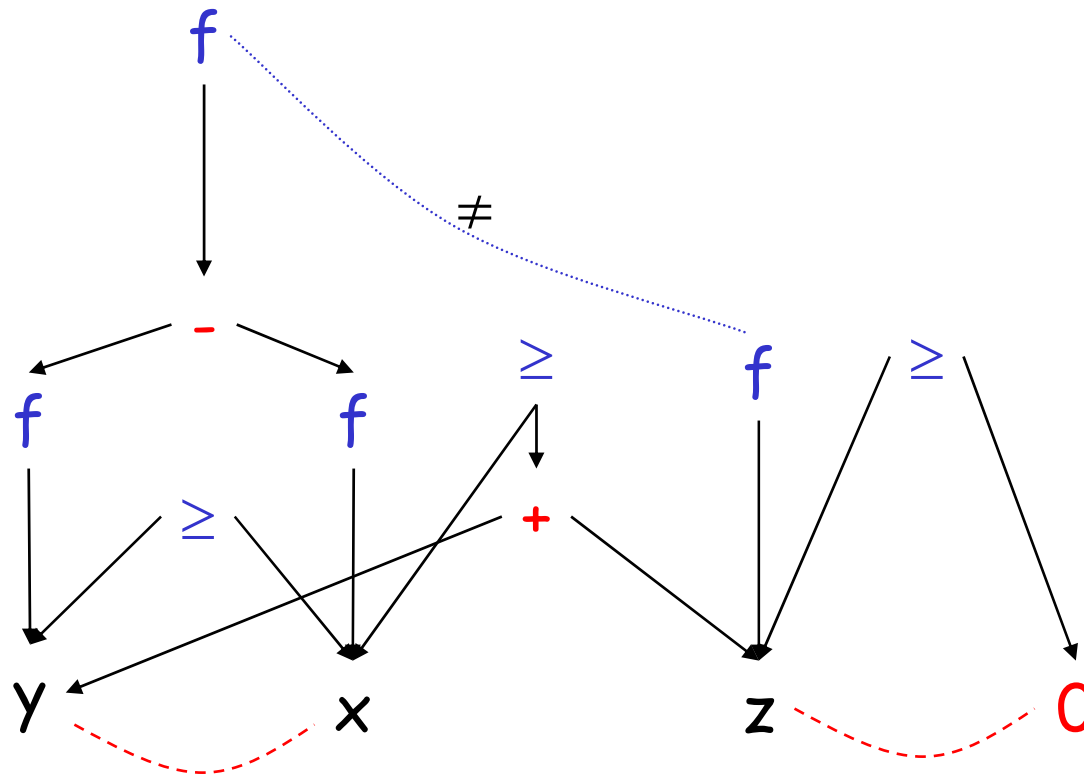


# Nelson-Oppen Method (2)

---

## 2. Run each sat. procedure

- Require it to report all contradictions (as usual)
- Also require it to report all equalities between nodes



# Nelson-Oppen Method (3)

3. Broadcast all discovered equalities and re-run sat. procedures

- Until no more equalities are discovered or a contradiction arises

