

Computational Logic

CS157

# Equality IV

Selene Makarios

*(docet pro* Mike Genesereth)

Autumn 2005

## Redux – Proving by Rewriting with Equalities

Suppose we have a theory that includes

1.  $f(e, x) = x$
2.  $f(x, e) = x$
3.  $f(g(x), x) = e$
4.  $f(x, g(x)) = e$
5.  $g(e) = e$
6.  $g(g(x)) = x$
7.  $P(a, b, c)$

At some point during inference, we obtain

$$8. P(f(e, f(e, g(g(a))))), f(g(g(b))), f(a, g(a)), f(g(g(e)), c))$$

## Redux – Proof By Equation-Based Rewriting

We saw that our *unit equalities* 1-6 could be used to simplify 8:

$$\begin{aligned}
 & P(f(e, f(e, g(g(a))))), f(g(g(b)), f(a, g(a))), f(g(g(e)), c)) \\
 1 & P(f(e, g(g(a))), f(g(g(b)), f(a, g(a))), f(g(g(e)), c)) \\
 1 & P(g(g(a)), f(g(g(b)), f(a, g(a))), f(g(g(e)), c)) \\
 6 & P(a, f(g(g(b)), f(a, g(a))), f(g(g(e)), c)) \\
 4 & P(a, f(g(g(b)), e), f(g(g(e)), c)) \\
 6 & P(a, f(b, e), f(g(g(e)), c)) \\
 2 & P(a, b, f(g(g(e)), c)) \\
 6 & P(a, b, f(e, c)) \\
 1 & P(a, b, c)
 \end{aligned}$$

and as result, we discovered that this new clause was *redundant* “modulo equality”.

## Term Rewrite Systems

Why are we confident that applying equations 1-6, in left-to-right order, that is, using the *term rewrite system*:

$$1. f(e, x) \Rightarrow x$$

$$2. f(x, e) \Rightarrow x$$

$$3. f(g(x), x) \Rightarrow e$$

$$4. f(x, g(x)) \Rightarrow e$$

$$5. g(e) \Rightarrow e$$

$$6. g(g(x)) \Rightarrow x$$

will simplify things? Starting with formula 8, after each rewrite (i.e. equational substitution), we had fewer symbols, and so we must eventually stop (i.e. get a term that no rule applies to). Is there *any* term we can start with that could allow the rewriting fun to go on forever?

## Term Ordering

So, we have a *term ordering*,  $\triangleright$ , for this system of equalities; the left side is always *larger than* the right side, according to the ordering. In this example, the term ordering is based on the number of symbols  $\|t\|$  in a term a term  $t$ :

$$s \triangleright t \quad \text{iff} \quad \|s\| > \|t\| .$$

so,

1.  $f(e, x) \Rightarrow x \quad (3 > 1)$
2.  $f(x, e) \Rightarrow x \quad (3 > 1)$
3.  $f(g(x), x) \Rightarrow e \quad (4 > 1)$
4.  $f(x, g(x)) \Rightarrow e \quad (4 > 1)$
5.  $g(e) \Rightarrow e \quad (2 > 1)$
6.  $g(g(x)) \Rightarrow x \quad (3 > 1)$

*Usually*, finding a way to order terms is *not so simple*.

## Term Ordering

But, given some system of equations, if we *can* find a way of ordering the terms, such that

$$s_i \Rightarrow t_i \text{ implies } s_i \triangleright t_i$$

and

any sequence of terms  $u_1 \triangleright u_2 \triangleright \dots$  is finite,

then term rewriting with this system will *always terminate*.

Furthermore, if  $\triangleright$  has the properties

$$(s \triangleright t) \text{ implies } (u[s] \triangleright u[t])$$

and

$$(s \triangleright t) \text{ implies } (s\theta \triangleright t\theta)$$

then it is called a *reduction ordering*.

## Term Rewrite Systems

Given this rewrite system

$$\left\{ \begin{array}{l} f(x) \Rightarrow g(x) \\ g(x) \Rightarrow h(x) \\ h(x) \Rightarrow f(x) \end{array} \right\}$$

and starting with  $f(a)$ ,

$$f(a) \Rightarrow g(a) \Rightarrow h(a) \Rightarrow f(a) \Rightarrow \dots$$

will this ever stop...? Is there some input for which it *will* stop?

## Term Rewrite Systems

What about this...?

$$\{ f(f(x)) \Rightarrow f(x) \}$$

$$f(f(f(f(a)))) \Rightarrow f(f(f(a))) \Rightarrow f(f(a)) \Rightarrow \dots$$

How about this...?

$$\left\{ \begin{array}{l} f(g(x)) \Rightarrow g(f(x)) \\ g(f(x)) \Rightarrow f(g(x)) \\ g(x) \Rightarrow x \end{array} \right\}$$

Starting from  $f(g(g(g(a))))$ , is it possible to use these rewrite rules in *some* way so that the system goes forever? In some way so that it stops (no rules apply)?

## Term Rewrite Systems

How about this...?

$$h(f(x, y)) \Rightarrow g(h(x), h(y))$$

$$h(g(x, y)) \Rightarrow f(h(x), h(y))$$

$$h(h(x)) \Rightarrow x$$

Will it:

- Run forever for some inputs?
- Stop for all inputs?

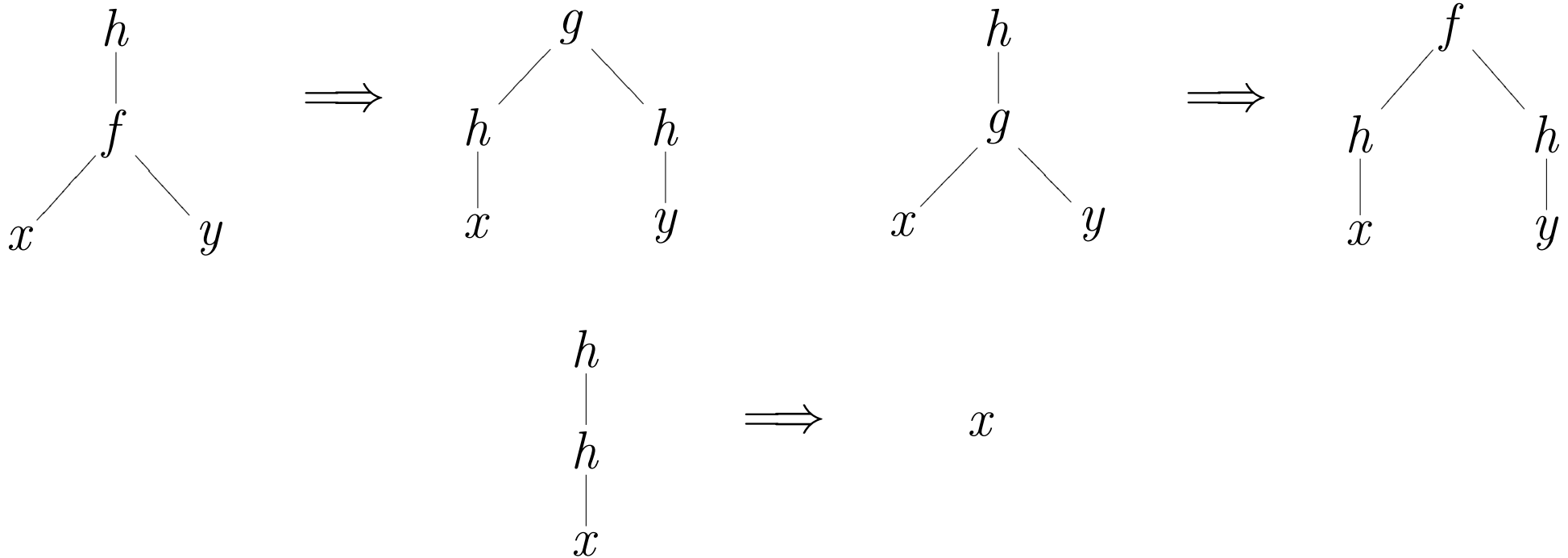
# Term Rewrite Systems

A different look:

$$h(f(x, y)) \Rightarrow g(h(x), h(y))$$

$$h(g(x, y)) \Rightarrow f(h(x), h(y))$$

$$h(h(x)) \Rightarrow x$$



## Term Rewrite Systems

Every rule moves some  $h$  deeper into the term, and no rule makes the term itself deeper – we can only do that finitely many times (for a finite term).

$$h(f(x, y)) \Rightarrow g(h(x), h(y))$$

$$h(g(x, y)) \Rightarrow f(h(x), h(y))$$

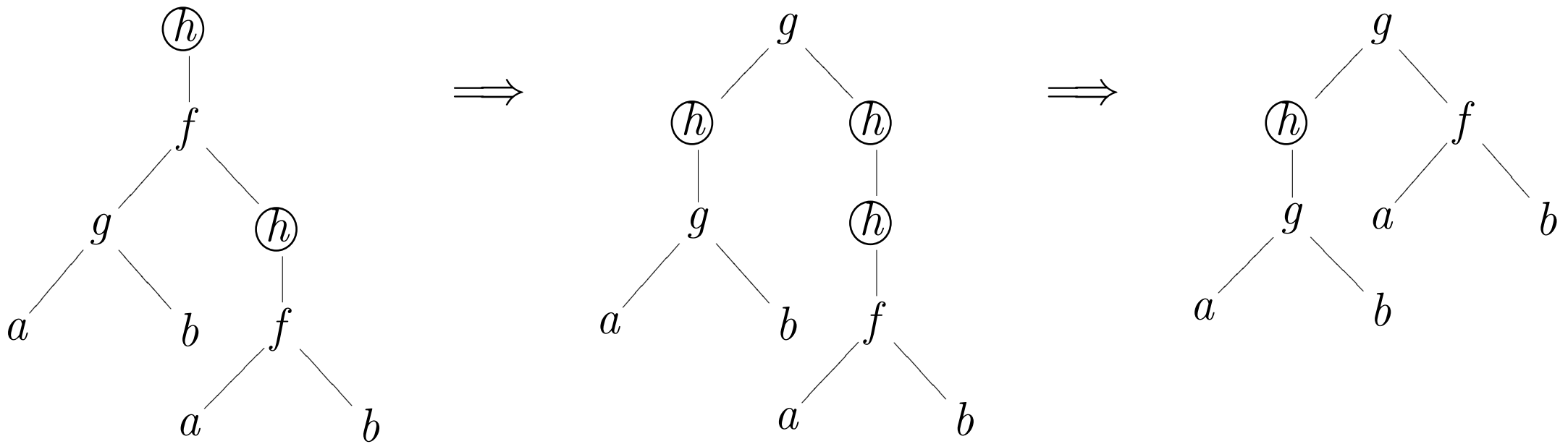
$$h(h(x)) \Rightarrow x$$

“Height” of  $h$

$$\begin{array}{l} \{3, 5\} \\ \{3, 4, 4\} \\ \{3\} \end{array} \quad \begin{array}{l} h(f(g(a, b), h(f(a, b)))) \Rightarrow \\ g(h(g(a, b)), h(h(f(a, b)))) \Rightarrow \\ g(f(h(a), h(b)), f(a, b)) \end{array}$$

# Term Rewrite Systems

In pictures:



## Term Rewrite Systems

How about this term rewrite system?

$$\text{R1 } 0 + x \Rightarrow x$$

$$\text{R2 } x + 0 \Rightarrow x$$

$$\text{R3 } -x + x \Rightarrow 0$$

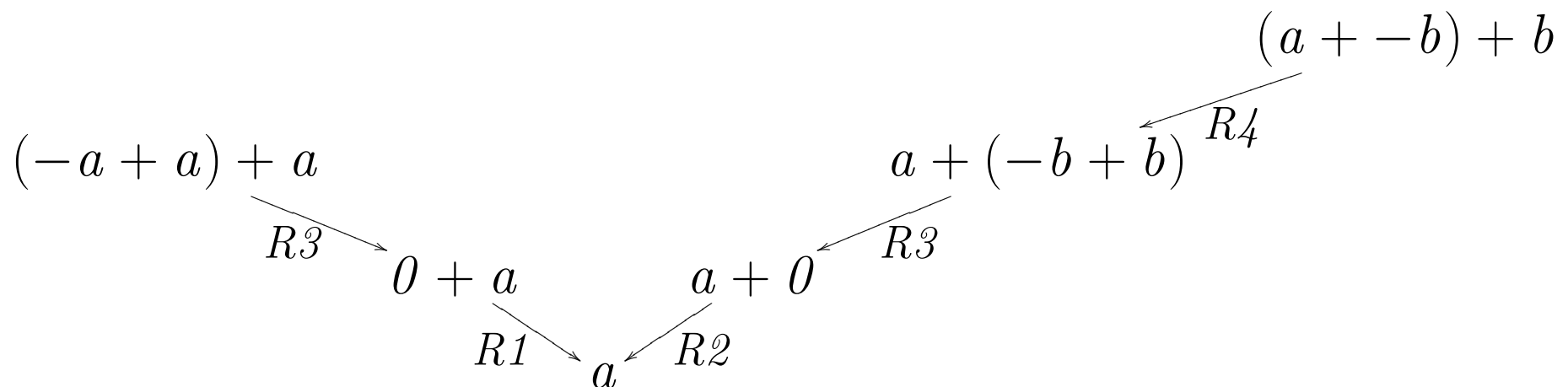
$$\text{R4 } (x + y) + z \Rightarrow x + (y + z)$$

It turns out that it stops on all inputs – but what is it good for? We can prove  $a + 0 = a$ . Can we prove  $--a = a$ ? (Is that true?)

## Term Rewrite Systems

Something we can prove:

$$(-a + a) + a = (a + -b) + b .$$



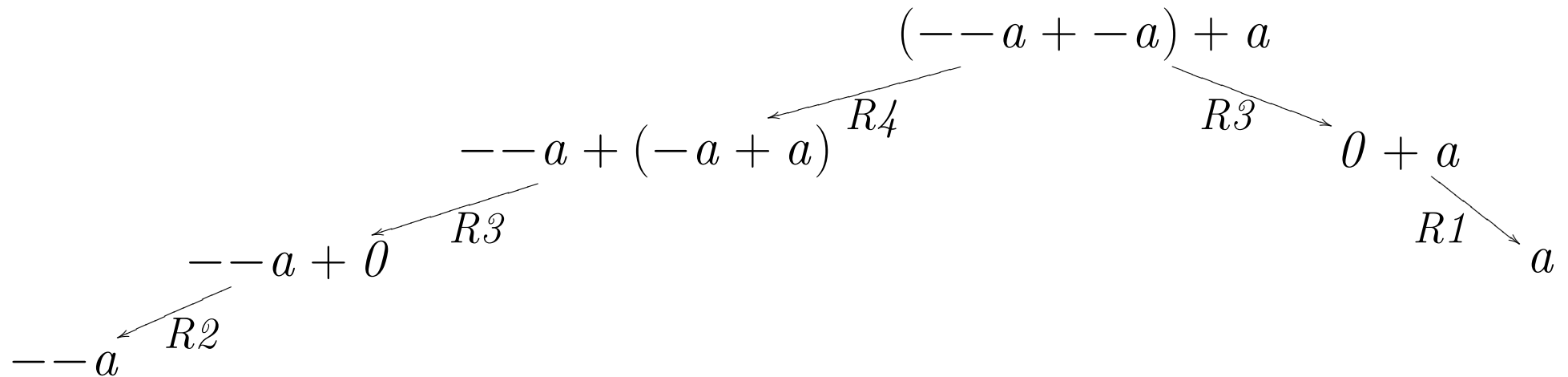
We were able to apply our rewrite rules to both sides, until they were the same.

# Term Rewrite Systems

Something we can't prove,

$$---a = a$$

unless we nondeterministically *guessed* an intermediate term, such as,



But we can't get a proof by applying our existing rewrite rules, *starting with* the left and right sides.

## Term Rewrite Systems

What to do? Make up *new rules* as we go along! 😊 (But, principled rules.)

To make up a new rule, we start by “superposing” pairs of our original rules.

Superposing means unifying *one left side* with *all or part* of another left side.

Let’s superpose:

$$\mathbf{R1} \quad 0 + x \Rightarrow x$$

$$\mathbf{R2} \quad x + 0 \Rightarrow x$$

## Term Rewrite Systems

### Superposing

$$\mathbf{R1} \quad 0 + x \Rightarrow x$$

$$\mathbf{R2} \quad x + 0 \Rightarrow x$$

produces a *unified* left side of  $0 + 0$ . Then we apply the two rules we used *to the result*.

$$0 + 0 \Rightarrow 0 \text{ by R1}$$

$$0 + 0 \Rightarrow 0 \text{ by R2}$$

In this case, the results are the same – so we *could* introduce a new rule, based on the two results:

$$\mathbf{RZ} \quad 0 \Rightarrow 0 \quad \times$$

but it hardly seems worth it. Let's try again...

## Term Rewrite Systems

### Superposing

$$\mathbf{R1} \quad 0 + x \Rightarrow x$$

$$\mathbf{R4} \quad (x + y) + z \Rightarrow x + (y + z)$$

produces a left side of  $(0 + y) + z$ . Apply rules to result:

$$(0 + y) + z \Rightarrow y + z \text{ by R1}$$

$$(0 + y) + z \Rightarrow 0 + (y + z) \text{ by R2}$$

In this case, the results are not the same, *but*,

$$0 + (y + z) \Rightarrow y + z \text{ by R1}$$

they are still the same “modulo” our existing rules. So a new rule based on these two right sides can’t help us:

$$\mathbf{RZ} \quad 0 + (y + z) \Rightarrow y + z \quad \times$$

## Term Rewrite Systems

### Superposing

$$\text{R2 } x + 0 \Rightarrow x$$

$$\text{R3 } -x + x \Rightarrow 0$$

produces a left side of  $-0 + 0$ . Applying the two rules to the result produces

$$-0 + 0 \Rightarrow -0 \text{ by R2}$$

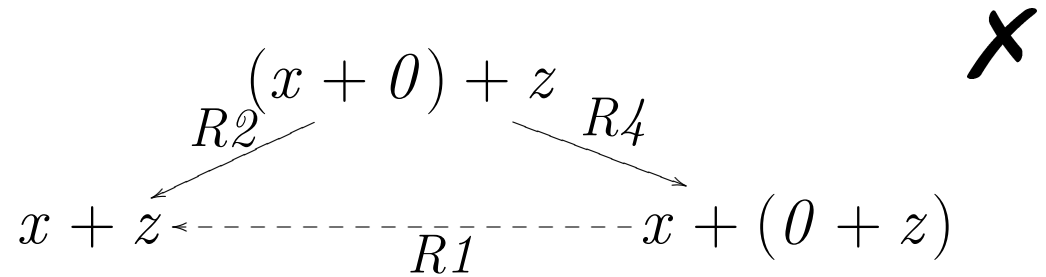
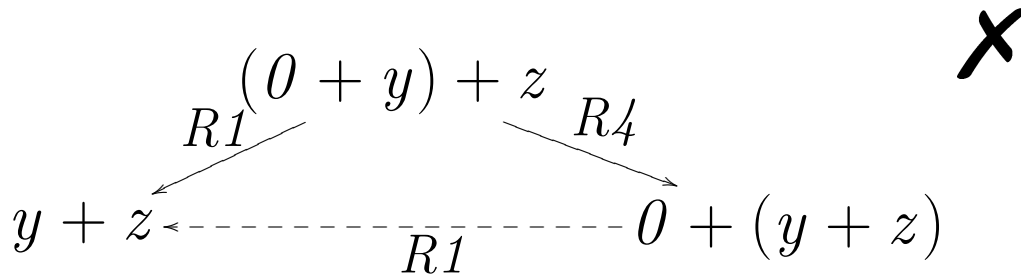
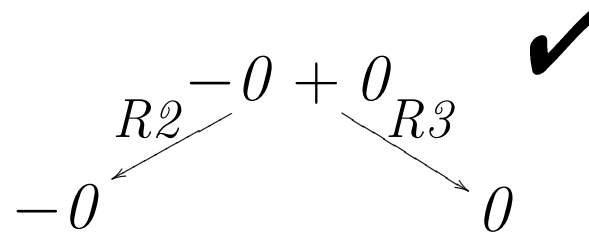
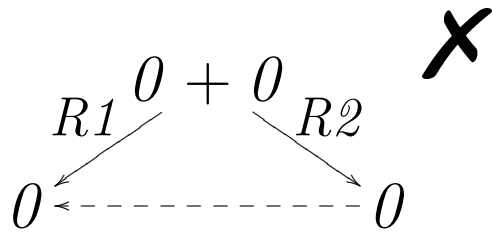
$$-0 + 0 \Rightarrow 0 \text{ by R3}$$

No existing rules will equate  $-0$  with  $0$ , so we can introduce new rule:

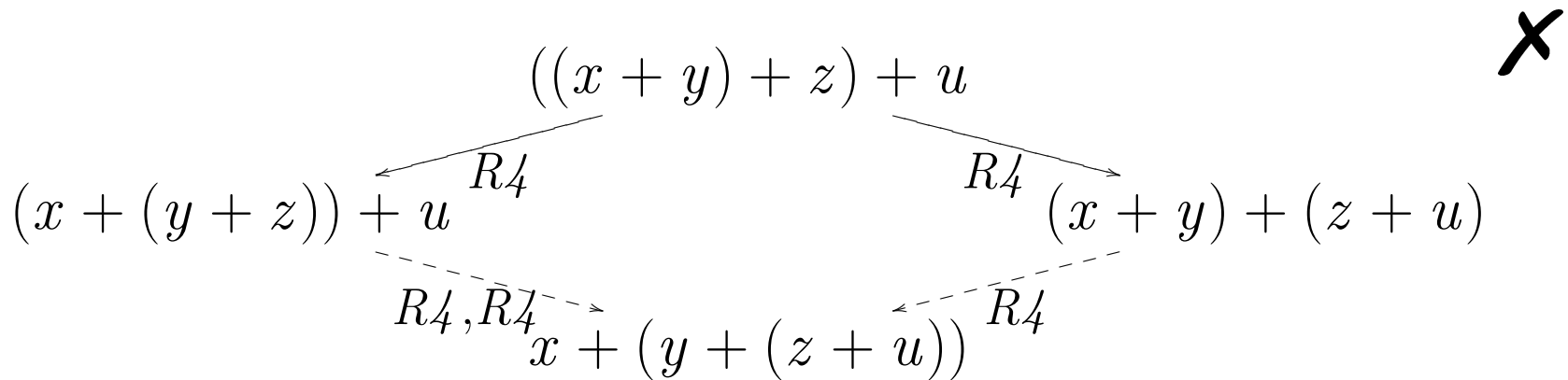
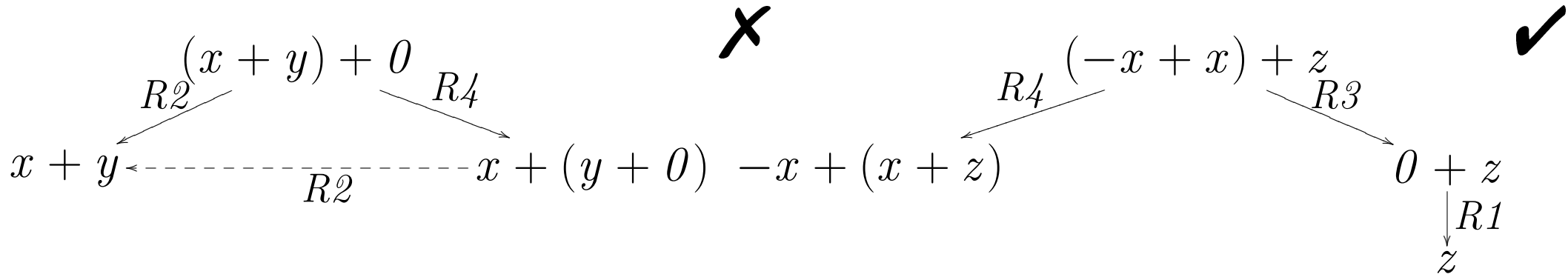
$$\text{RA } -0 \Rightarrow 0 \checkmark$$

Yay! 😊

# Term Rewrite Systems



# Term Rewrite Systems



## Term Rewrite Systems

$$\mathbf{R1} \quad 0 + x \Rightarrow x$$

$$\mathbf{R2} \quad x + 0 \Rightarrow x$$

$$\mathbf{R3} \quad -x + x \Rightarrow 0$$

$$\mathbf{R4} \quad (x + y) + z \Rightarrow x + (y + z)$$

$$\mathbf{RA} \quad -0 \Rightarrow 0$$

$$\mathbf{RB} \quad -x + (x + z) \Rightarrow z$$



## Term Rewrite Systems

Results of all possible superpositions using our new rules:

$$-0 + z = z$$

$$-x + x = 0 + 0$$

$$\boxed{--x + 0 = x}$$

$$-(x + y) + (x + (y + z)) = z$$

$$0 + z = z$$

$$x + y = --x + y$$

Let's try the third one.



## Term Rewrite Systems

$$\mathbf{R1} \quad 0 + x \Rightarrow x$$

$$\mathbf{R2} \quad x + 0 \Rightarrow x$$

$$\mathbf{R3} \quad -x + x \Rightarrow 0$$

$$\mathbf{R4} \quad (x + y) + z \Rightarrow x + (y + z)$$

$$\mathbf{RA} \quad -0 \Rightarrow 0$$

$$\mathbf{RB} \quad -x + (x + z) \Rightarrow z$$

$$\mathbf{RC} \quad --x \Rightarrow x$$

## Term Rewrite Systems

Very short proof, applying rules only to the starting terms:

$$\begin{array}{c} \text{---} \text{---} a \\ \downarrow RC \\ a \end{array}$$

## Theory Unification

If AC-Unification left you hungry for more...

$\emptyset(f)$	$:= \{ \}$	Empty Theory
$A(f)$	$:= \{f(x, f(y, z)) = f(f(x, y), z)\}$	Associativity
$C(f)$	$:= \{f(x, y) = f(y, x)\}$	Commutativity
$I(f)$	$:= \{f(x, x) = x\}$	Idempotency
$D_l(f, g)$	$:= \{f(g(x, y), z) = g(f(x, z), f(y, z))\}$	Left Distributivity
$D_r(f, g)$	$:= \{f(x, g(y, z)) = g(f(x, y), f(x, z))\}$	Right Distributivity
$D(f, g)$	$:= D_l \cup D_r$	Distributivity
$Inv_l(f, i, e)$	$:= \{f(i(x), x) = e\}$	Left Inverse
$Inv_r(f, i, e)$	$:= \{f(x, i(x)) = e\}$	Right Inverse
$Inv(f, i, e)$	$:= Inv_l \cup Inv_r$	Inverse
$Id_l(f, e)$	$:= \{f(e, x) = x\}$	Identity Left
$Id_r(f, e)$	$:= \{f(x, e) = x\}$	Identity Right
$Id(f)$	$:= Id_l \cup Id_r$	Identity
$Inv(f)$	$:= \{f(f(x)) = x\}$	Involution
$E(h, f)$	$:= \{h(f(x_1, \dots, x_n)) = f(h(x_1), \dots, h(x_n))\}$	Endomorphism
$AE(h, f)$	$:= \{h(f(x_1, \dots, x_n)) = f(h(x_n), \dots, h(x_1))\}$	Anti-Endomorphism

## Universal Unification

1. Remove each equation  $t = t$  from  $\Gamma$ .
2. Replace each equation  $f(p_1, \dots, p_n) = f(q_1, \dots, q_n)$  by the equations  $p_1 = q_1, \dots, p_n = q_n$ .
3. For each equation  $x = t$  replace all other occurrences of  $x$  in  $\Gamma$  by  $t$ , unless  $x$  occurs in  $t$ . Do the same for all equations  $x = y$ .
4. Replace each equation  $t = x$  by  $x = t$ , if  $t$  is not a variable.

5. Replace any equation of the form  $f(p_1, \dots, p_n) = t$  by  $f(p_1, \dots, p_n) = f(q_1, \dots, q_n)$  and  $s = t$  if  $f(q_1, \dots, q_n) = s$  or  $s = f(q_1, \dots, q_n)$  is a variant of an axiom in the theory. (Note that this rule may not again be applied to  $f(p_1, \dots, p_n) = f(q_1, \dots, q_n)$ .)
6. If  $x$  is among the variables in  $q_i$ , replace  $x = f(q_1, \dots, q_n)$  by  $x = f(v_1, \dots, v_n)$  and  $v_1 = q_1, \dots, v_n = q_n$ , and replace all other occurrences of  $x$  in  $\Gamma$  by  $t$ .

## Term-Ordering; Sidebar – Multisets

Intuitively, a multiset (or “bag”) is an unordered collection of possibly non-unique elements.

Example:  $\{a, b, b, c\}$ .

Formally, a multiset  $S$  is specified by a mapping on some base set  $C$  to the non-negative integers,

$$S : C \rightarrow \mathbb{N} .$$

A multiset  $S$  is *empty* if  $S(x) = 0$  for every  $x \in C$ . A multiset  $S$  is *finite* if  $S(x) > 0$  for only a finite number of  $x \in C$ .

## Sidebar – Multisets

We say:

$$x \in S \quad \text{iff} \quad S(x) > 0 \quad ,$$

$$S_1 \subset S_2 \quad \text{iff} \quad \text{forall } x \text{ in } C, S_1(x) \leq S_2(x) \quad ,$$

$$(S_1 \cap S_2)(x) = \min(S_1(x), S_2(x)) \quad ,$$

$$(S_1 \setminus S_2)(x) = \max(S_1(x) - S_2(x), 0) \quad .$$

$$(S_1 \cup S_2)(x) = S_1(x) + S_2(x) \quad !$$

## Sidebar – Multisets

Why is

$$(S_1 \cup S_2)(x) = S_1(x) + S_2(x) \quad ?$$

It *does* correspond intuitively to the idea of combining two bags of objects. *But*, in regular set theory, the union of  $S_1$  and  $S_2$  is the *smallest set that contains them both*. Generalizing this to multisets should give

$$(S_1 \cup S_2)(x) = \max(S_1(x), S_2(x)) \quad ,$$

and so  $\{a, b, b, b, c, c\} \cup \{a, a, c, c, c\}$  would just need to be  $\{a, a, b, b, b, c, c, c\}$ . However, the standard definition gives  $\{a, a, a, b, b, b, c, c, c, c, c\}$ .

(Note: the standard definition *also* breaks DeMorgan's laws.)

## Sidebar – Multiset Orders

If  $>$  is a partial ordering on  $C$ , then  $>^*$  is the *multiset extension* of  $>$ ,

$$S_1 >^* S_2 \text{ iff } \left( \begin{array}{l} \text{for every } x \text{ in } C \text{ s.t. } S_1(x) < S_2(x), \\ \text{for some } y \text{ in } C, \\ y > x \text{ and } S_1(y) > S_2(y) \end{array} \right)$$

What this *actually says* is to find the highest element in each multiset according to  $>$ , and see which multiset has more of it. If it's a tie, check the next highest element.

Example: with  $a > b > c$ ,

$$\begin{aligned} \{b, a, a, c, c, c\} &>^* \{c, a, b, b\} \text{ ?} \\ \{a\} &>^* \{c, c, c, b, b, c\} \text{ ?} \\ \{b, b, a, c\} &>^* \{a, c, c, b\} \text{ ?} \end{aligned}$$

## Recursive Path Ordering

Let  $>$  be a partial ordering on a set of operators  $F$ . The *recursive path ordering*  $\triangleright$  on the set  $T(F)$  of terms over  $F$  is defined recursively as follows:

$$s = f(s_1, \dots, s_m) \triangleright g(t_1, \dots, t_n) = t \text{ ,}$$

if and only if

$$(i) \ f = g \text{ and } \{s_1, \dots, s_m\} \triangleright^* \{t_1, \dots, t_n\}$$

or

$$(ii) \ f > g \text{ and } \{s\} \triangleright^* \{t_1, \dots, t_n\}$$

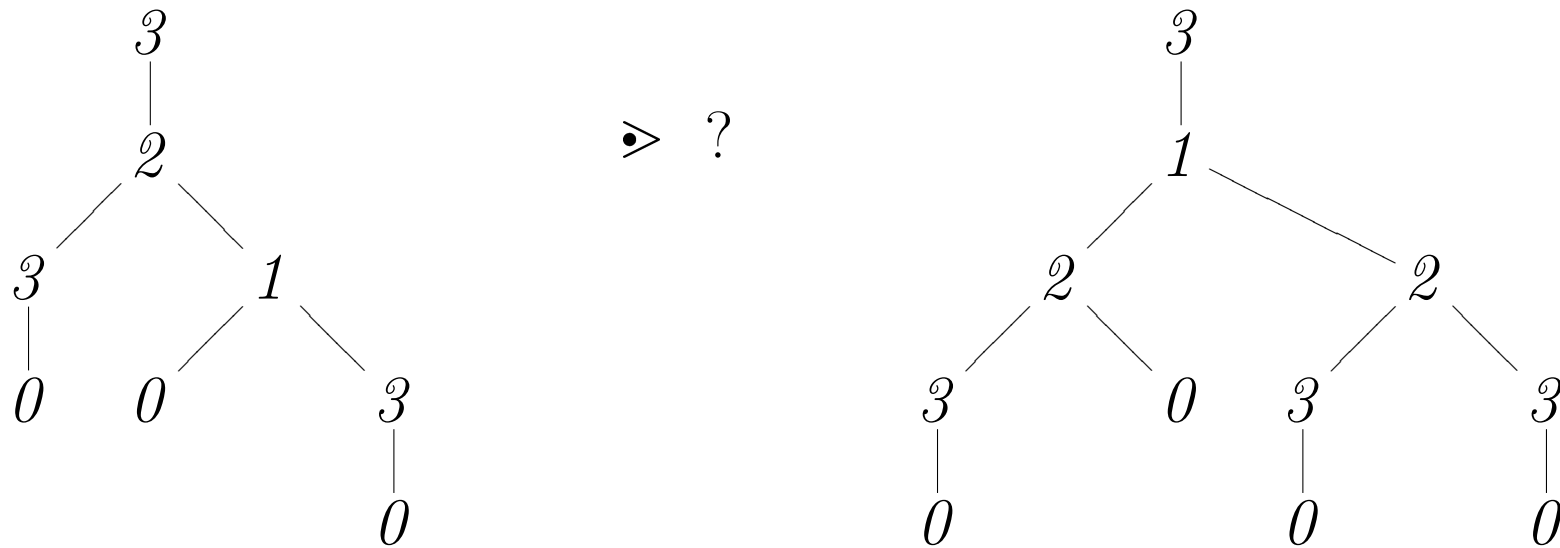
or

$$(iii) \ f \not> g \text{ and } \{s_1, \dots, s_m\} \succeq^* \{t\} \text{ ,}$$

where  $\triangleright^*$  is the extension of  $\triangleright$  to multisets, and  $\succeq^*$  means  $\triangleright^*$  or  $=$ .

# Recursive Path Ordering

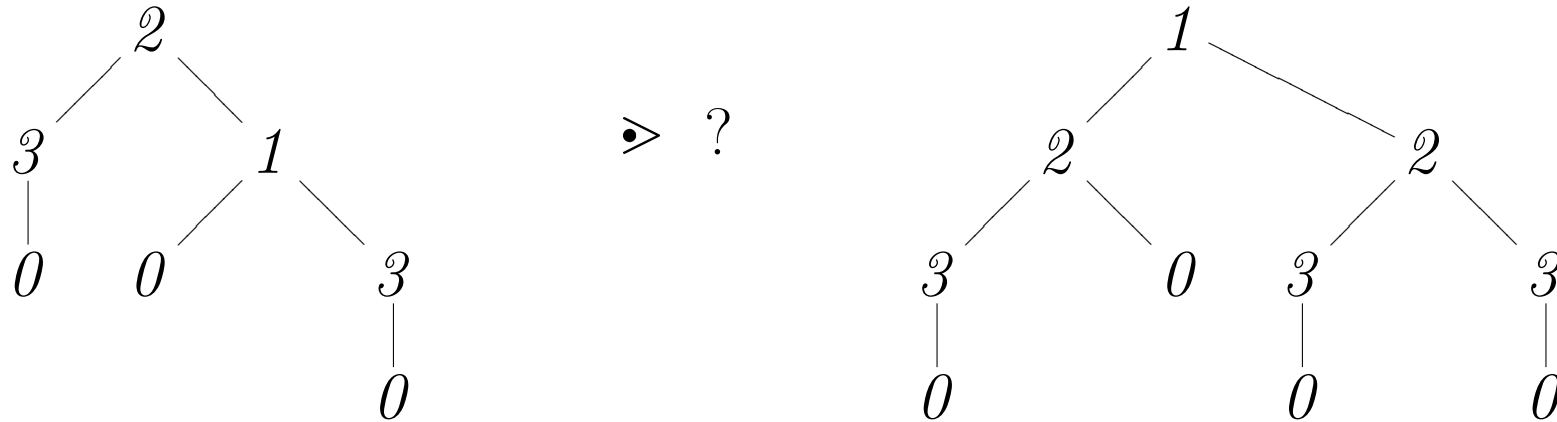
Given  $3 > 2 > 1$ ,



Use

(i)  $f = g$  and  $\{s_1, \dots, s_m\} \triangleright^* \{t_1, \dots, t_n\}$

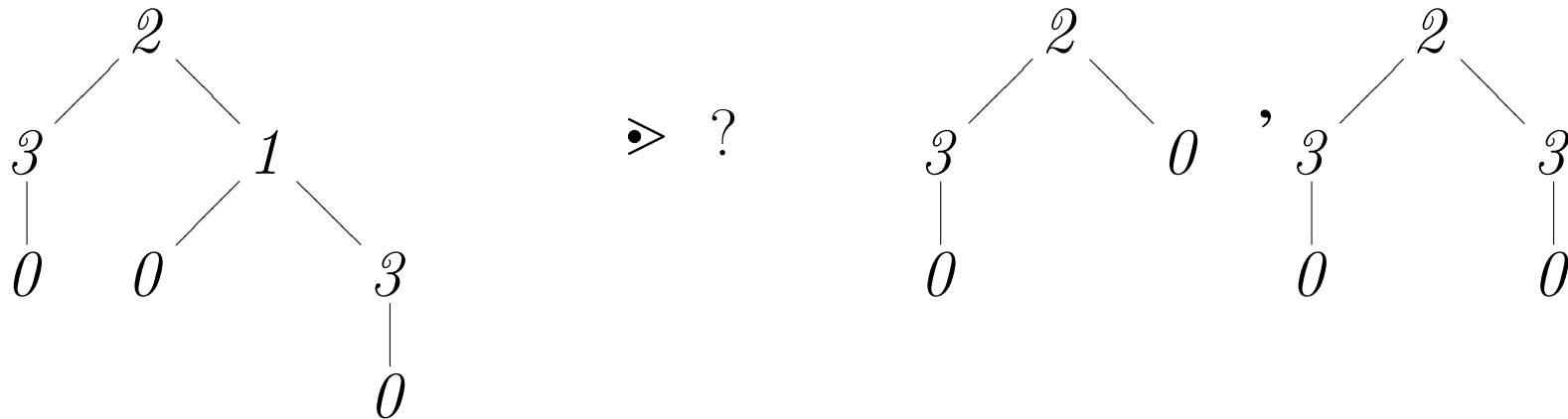
# Recursive Path Ordering



Use

$$(ii) f > g \text{ and } \{s\} \triangleright^* \{t_1, \dots, t_n\}$$

# Recursive Path Ordering



Use

$$(i) f = g \text{ and } \{s_1, \dots, s_m\} \triangleright^* \{t_1, \dots, t_n\}$$

## Recursive Path Ordering

$$\left\{ \begin{array}{l} 3 \\ | \\ 0 \end{array} , \begin{array}{c} 1 \\ / \quad \backslash \\ 0 \quad 3 \\ | \\ 0 \end{array} \right\} \succ^* ? \left\{ \begin{array}{l} 3 \\ | \\ 0 \end{array} , 0 \right\} , \left\{ \begin{array}{l} 3 \\ | \\ 0 \end{array} , \begin{array}{l} 3 \\ | \\ 0 \end{array} \right\}$$

$3 - 0$  is in all the multisets; okay to cancel it out.

Use

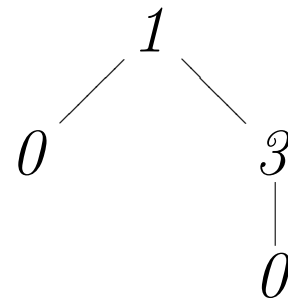
$$(ii) f > g \text{ and } \{s\} \succ^* \{t_1, \dots, t_n\}$$

and

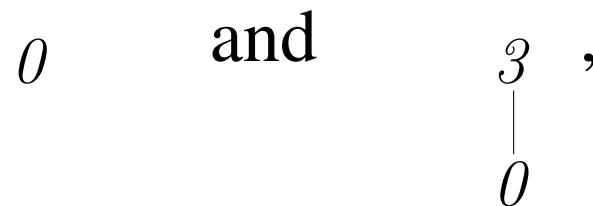
$$(iii) f \not> g \text{ and } \{s_1, \dots, s_m\} \succeq^* \{t\}$$

## Recursive Path Ordering

So finally, since



is greater than both



we have  $s \triangleright t$ .