

# Term Rewriting and Simplification

Jeffrey Clark  
Elon University

April 29, 2008

# Don't Take Too Many Notes

This talk is available from my web-site:

`http://frodo.elon.edu`

under the link **Presentations**.

# Introduction: Expressions

We are used to having different ways of expressing the same concept:

- $3/6 = 2/4 = 1/2$

# Introduction: Expressions

We are used to having different ways of expressing the same concept:

- $3/6 = 2/4 = 1/2$
- $x + x = 2x$

# Introduction: Expressions

We are used to having different ways of expressing the same concept:

- $3/6 = 2/4 = 1/2$
- $x + x = 2x$
- $x \cdot x \cdot x = x^3$

# Introduction: Preferred Forms

Some ways are preferred:

- $3/6 = 2/4 = 1/2$

# Introduction: Preferred Forms

Some ways are preferred:

- $3/6 = 2/4 = 1/2$
- $x + x = 2x$

# Introduction: Preferred Forms

Some ways are preferred:

- $3/6 = 2/4 = 1/2$
- $x + x = 2x$
- $x \cdot x \cdot x = x^3$

# Introduction: Simplification

- Some expressions that represent the same concept are deemed equivalent.

# Introduction: Simplification

- Some expressions that represent the same concept are deemed equivalent.
- The class of equivalent expressions can be ordered by complexity.

# Introduction: Simplification

- Some expressions that represent the same concept are deemed equivalent.
- The class of equivalent expressions can be ordered by complexity.
- If out of all the equivalent expressions for expressing the same concept there is a simplest one, that form is to be preferred.

## Introduction: Examples

- Positive fractions  $a/b$  and  $c/d$  are equivalent if  $ad = bc$ ; for equivalent fractions, the smaller the magnitude of the numerator and/or denominator the better. The simplest form of the fraction has the smallest magnitude for both the numerator and denominator.

## Introduction: Examples

- Positive fractions  $a/b$  and  $c/d$  are equivalent if  $ad = bc$ ; for equivalent fractions, the smaller the magnitude of the numerator and/or denominator the better. The simplest form of the fraction has the smallest magnitude for both the numerator and denominator.
- Two polynomial expressions are equivalent if the total of the coefficients for each power of the variable are equal; for equivalent expressions the smaller the number of terms the simpler the expression, and if the terms are the same then ordering them from highest to lowest power is simpler; the standard form for a polynomial is the simplest.

# Terms

- We will be using **terms** as the objects we act upon.

# Terms

- We will be using **terms** as the objects we act upon.
- Terms are formed by a countably infinite supply of variables  $\{x_1, x_2, \dots\}$  and a finite collection of operators, each with a fixed degree.

# Terms

- We will be using **terms** as the objects we act upon.
- Terms are formed by a countably infinite supply of variables  $\{x_1, x_2, \dots\}$  and a finite collection of operators, each with a fixed degree.
- Every variable is a term.

# Terms

- We will be using **terms** as the objects we act upon.
- Terms are formed by a countably infinite supply of variables  $\{x_1, x_2, \dots\}$  and a finite collection of operators, each with a fixed degree.
- Every variable is a term.
- If  $f$  is an operator of degree  $n$  and  $t_1, \dots, t_n$  are terms, then  $f(t_1, \dots, t_n)$  is a term. Constants are operators of degree 0.

# Terms

- We will be using **terms** as the objects we act upon.
- Terms are formed by a countably infinite supply of variables  $\{x_1, x_2, \dots\}$  and a finite collection of operators, each with a fixed degree.
- Every variable is a term.
- If  $f$  is an operator of degree  $n$  and  $t_1, \dots, t_n$  are terms, then  $f(t_1, \dots, t_n)$  is a term. Constants are operators of degree 0.
- Terms are written many different ways:  $x_1 + x_2$ ,  $D(f(x)/g(x))$ , with mixtures of prefix, infix, and postfix notation. The theory is easiest to express using prefix notation:  $+(x_1, x_2)$ ,  $D(/(f(x), g(x)))$ .

# Equivalence Relations

- A relation  $R$  on a set  $S$  is a function that takes two inputs  $a, b \in S$  and returns True or False as an output. It is usually written  $aRb$ .

# Equivalence Relations

- A relation  $R$  on a set  $S$  is a function that takes two inputs  $a, b \in S$  and returns True or False as an output. It is usually written  $aRb$ .
- $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \neq b$  are all examples of relations on the real numbers.  $a \mid b$  is an example of a relation on the integers.

# Equivalence Relations

- A relation  $R$  on a set  $S$  is a function that takes two inputs  $a, b \in S$  and returns True or False as an output. It is usually written  $aRb$ .
- $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \neq b$  are all examples of relations on the real numbers.  $a \mid b$  is an example of a relation on the integers.
- A relation  $R$  is **reflexive** if  $aRa$  is always true.  $a \leq b$  and  $a \mid b$  are reflexive.

# Equivalence Relations

- A relation  $R$  on a set  $S$  is a function that takes two inputs  $a, b \in S$  and returns True or False as an output. It is usually written  $aRb$ .
- $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \neq b$  are all examples of relations on the real numbers.  $a \mid b$  is an example of a relation on the integers.
- A relation  $R$  is **reflexive** if  $aRa$  is always true.  $a \leq b$  and  $a \mid b$  are reflexive.
- A relation  $R$  is **symmetric** if  $aRb$  implies that  $bRa$ .  $a = b$  and  $a \neq b$  are symmetric.

# Equivalence Relations

- A relation  $R$  on a set  $S$  is a function that takes two inputs  $a, b \in S$  and returns True or False as an output. It is usually written  $aRb$ .
- $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \neq b$  are all examples of relations on the real numbers.  $a \mid b$  is an example of a relation on the integers.
- A relation  $R$  is **reflexive** if  $aRa$  is always true.  $a \leq b$  and  $a \mid b$  are reflexive.
- A relation  $R$  is **symmetric** if  $aRb$  implies that  $bRa$ .  $a = b$  and  $a \neq b$  are symmetric.
- A relation  $R$  is **transitive** if  $aRb$  and  $bRc$  together imply that  $aRc$ .  $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \mid b$  are all transitive.

# Equivalence Relations

- A relation  $R$  on a set  $S$  is a function that takes two inputs  $a, b \in S$  and returns True or False as an output. It is usually written  $aRb$ .
- $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \neq b$  are all examples of relations on the real numbers.  $a \mid b$  is an example of a relation on the integers.
- A relation  $R$  is **reflexive** if  $aRa$  is always true.  $a \leq b$  and  $a \mid b$  are reflexive.
- A relation  $R$  is **symmetric** if  $aRb$  implies that  $bRa$ .  $a = b$  and  $a \neq b$  are symmetric.
- A relation  $R$  is **transitive** if  $aRb$  and  $bRc$  together imply that  $aRc$ .  $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \mid b$  are all transitive.
- A relation  $R$  is an **equivalence relation** if it is reflexive, symmetric, and transitive.

## Equivalence: Ignoring Irrelevant Details

- Equivalence relations allow us to focus on the crucial part of an expression, and to ignore irrelevant details.

## Equivalence: Ignoring Irrelevant Details

- Equivalence relations allow us to focus on the crucial part of an expression, and to ignore irrelevant details.
- Different expressions are useful in different contexts, and equivalence relations give us choices.

## Equivalence: Ignoring Irrelevant Details

- Equivalence relations allow us to focus on the crucial part of an expression, and to ignore irrelevant details.
- Different expressions are useful in different contexts, and equivalence relations give us choices.
- The collection of all expressions equivalent to a given expression is its **equivalence class**.

## Equivalence: Ignoring Irrelevant Details

- Equivalence relations allow us to focus on the crucial part of an expression, and to ignore irrelevant details.
- Different expressions are useful in different contexts, and equivalence relations give us choices.
- The collection of all expressions equivalent to a given expression is its **equivalence class**.
- Simplification involves trying to find the simplest member of an equivalence class if it exists.

# Order Relations

- A relation  $R$  is **irreflexive** if  $aRa$  is always false.  $a < b$  and  $a \neq b$  are irreflexive.

# Order Relations

- A relation  $R$  is **irreflexive** if  $aRa$  is always false.  $a < b$  and  $a \neq b$  are irreflexive.
- A relation  $R$  is **transitive** if  $aRb$  and  $bRc$  together imply that  $aRc$ .  $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \mid b$  are all transitive.

# Order Relations

- A relation  $R$  is **irreflexive** if  $aRa$  is always false.  $a < b$  and  $a \neq b$  are irreflexive.
- A relation  $R$  is **transitive** if  $aRb$  and  $bRc$  together imply that  $aRc$ .  $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \mid b$  are all transitive.
- A relation  $R$  is an **order relation** if it is irreflexive and transitive.

# Order Relations

- A relation  $R$  is **irreflexive** if  $aRa$  is always false.  $a < b$  and  $a \neq b$  are irreflexive.
- A relation  $R$  is **transitive** if  $aRb$  and  $bRc$  together imply that  $aRc$ .  $a < b$ ,  $a \leq b$ ,  $a = b$ , and  $a \mid b$  are all transitive.
- A relation  $R$  is an **order relation** if it is irreflexive and transitive.
- Simplification (in various contexts) is an order relation.

# Simplifying Examples

- Reduce denominators in fractions:  $3/6 \rightarrow 1/2$

# Simplifying Examples

- Reduce denominators in fractions:  $3/6 \rightarrow 1/2$
- Collect like terms:  $2x^2 + 3x^2 \rightarrow 5x^2$

# Simplifying Examples

- Reduce denominators in fractions:  $3/6 \rightarrow 1/2$
- Collect like terms:  $2x^2 + 3x^2 \rightarrow 5x^2$
- Distribute differentiation in sums:  
 $D(x^2 + x^3) \rightarrow D(x^2) + D(x^3)$

# Termination

- In theory this process need never stop.

# Termination

- In theory this process need never stop.
- If defined simplification for fractions as moving towards larger denominators, then the simplification would never end.

# Termination

- In theory this process need never stop.
- If defined simplification for fractions as moving towards larger denominators, then the simplification would never end.
- We require for simplification that there be no infinite chains of equivalent, simpler expressions.

# Termination

- In theory this process need never stop.
- If defined simplification for fractions as moving towards larger denominators, then the simplification would never end.
- We require for simplification that there be no infinite chains of equivalent, simpler expressions.
- Such an ordering relation on equivalence classes is said to be **terminating**.

# Termination

- In theory this process need never stop.
- If defined simplification for fractions as moving towards larger denominators, then the simplification would never end.
- We require for simplification that there be no infinite chains of equivalent, simpler expressions.
- Such an ordering relation on equivalence classes is said to be **terminating**.
- Given any expression, its simplest equivalent form is called its **canonical form**.

# Confluence

- In theory this process could produce different canonical forms from the same starting expression.

# Confluence

- In theory this process could produce different canonical forms from the same starting expression.
- We require for simplification that there be no inequivalent canonical forms.

# Confluence

- In theory this process could produce different canonical forms from the same starting expression.
- We require for simplification that there be no inequivalent canonical forms.
- Such an ordering relation on equivalence classes is said to be **confluent**.

# Substitutions

- Terms can serve as patterns, if we define substitutions for the variables contained in them.

# Substitutions

- Terms can serve as patterns, if we define substitutions for the variables contained in them.
- The Sum Rule refers to expressions of the pattern  $D(+ (x_1, x_2))$ . It is only useful when we are able to replace  $x_1$  and  $x_2$  by arbitrary expressions.

# Substitutions

- Terms can serve as patterns, if we define substitutions for the variables contained in them.
- The Sum Rule refers to expressions of the pattern  $D(+ (x_1, x_2))$ . It is only useful when we are able to replace  $x_1$  and  $x_2$  by arbitrary expressions.
- A substitution is a function  $\sigma$  defined recursively on terms.

# Substitutions

- Terms can serve as patterns, if we define substitutions for the variables contained in them.
- The Sum Rule refers to expressions of the pattern  $D(+ (x_1, x_2))$ . It is only useful when we are able to replace  $x_1$  and  $x_2$  by arbitrary expressions.
- A substitution is a function  $\sigma$  defined recursively on terms.
- $\sigma(x_k)$  is a term, and differs from  $x_k$  for only a finite number of variables.

# Substitutions

- Terms can serve as patterns, if we define substitutions for the variables contained in them.
- The Sum Rule refers to expressions of the pattern  $D(+ (x_1, x_2))$ . It is only useful when we are able to replace  $x_1$  and  $x_2$  by arbitrary expressions.
- A substitution is a function  $\sigma$  defined recursively on terms.
- $\sigma(x_k)$  is a term, and differs from  $x_k$  for only a finite number of variables.
- If  $f$  is an operator of degree  $n$ , then  $\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n))$  for arbitrary terms  $t_1, t_2, \dots, t_n$ .

# Substitutions

- Terms can serve as patterns, if we define substitutions for the variables contained in them.
- The Sum Rule refers to expressions of the pattern  $D(+ (x_1, x_2))$ . It is only useful when we are able to replace  $x_1$  and  $x_2$  by arbitrary expressions.
- A substitution is a function  $\sigma$  defined recursively on terms.
- $\sigma(x_k)$  is a term, and differs from  $x_k$  for only a finite number of variables.
- If  $f$  is an operator of degree  $n$ , then  $\sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n))$  for arbitrary terms  $t_1, t_2, \dots, t_n$ .
- For example, define  $\sigma(x_1) = +(x_1, x_2)$  and  $\sigma(x_2) = x_3$ , with  $\sigma(x_k) = x_k$  for all other variables. Then  $\sigma(D(+ (x_1, x_2))) = D(\sigma(+ (x_1, x_2))) = D(+ (\sigma(x_1), \sigma(x_2))) = D(+ (+ (x_1, x_2), x_3))$ .

# Invariance

- We will want to work with patterns that are ordered. Patterns can be instantiated many different ways with substitutions.

# Invariance

- We will want to work with patterns that are ordered. Patterns can be instantiated many different ways with substitutions.
- We will add a very strict requirement to our order relations: they should be invariant under substitutions.

# Invariance

- We will want to work with patterns that are ordered. Patterns can be instantiated many different ways with substitutions.
- We will add a very strict requirement to our order relations: they should be invariant under substitutions.
- If  $t_1 < t_2$ , then for any substitution  $\sigma$ , we will require that  $\sigma(t_1) < \sigma(t_2)$ .

# Invariance

- We will want to work with patterns that are ordered. Patterns can be instantiated many different ways with substitutions.
- We will add a very strict requirement to our order relations: they should be invariant under substitutions.
- If  $t_1 < t_2$ , then for any substitution  $\sigma$ , we will require that  $\sigma(t_1) < \sigma(t_2)$ .
- A **reduction order** is an order relation that is terminating, confluent, and invariant under substitutions.

# Rewrite Rules

- The specific steps in simplification are encoded in **rewrite rules**.

# Rewrite Rules

- The specific steps in simplification are encoded in **rewrite rules**.
- A rewrite rule is of the form  $t_L \rightarrow t_R$ .  $t_L$  is referred to as the left side and  $t_R$  as the right side of the rule.

# Rewrite Rules

- The specific steps in simplification are encoded in **rewrite rules**.
- A rewrite rule is of the form  $t_L \rightarrow t_R$ .  $t_L$  is referred to as the left side and  $t_R$  as the right side of the rule.
- The Sum Rule (when its hypotheses are satisfied) can be written as  $D(+ (x_1, x_2)) \rightarrow + (D(x_1), D(x_2))$ .

# Rewrite Rules

- The specific steps in simplification are encoded in **rewrite rules**.
- A rewrite rule is of the form  $t_L \rightarrow t_R$ .  $t_L$  is referred to as the left side and  $t_R$  as the right side of the rule.
- The Sum Rule (when its hypotheses are satisfied) can be written as  $D(+ (x_1, x_2)) \rightarrow + (D(x_1), D(x_2))$ .
- A rewrite rule can be applied to a term  $t$  if any sub-term  $t_0$  of  $t$  is of the same form as the left side of the rewrite rule, with some substitution for the variables appearing in  $t_L$ . Then applying that same substitution to the variables in the right side of the rule yields a replacement for the sub-term  $t_0$ .

## Rewrite Rules: Example

- Applying the Sum Rule to  $D(+(+x_1, x_2), x_3)$  requires the substitution  $x_1 \rightarrow +(x_1, x_2)$  and  $x_2 \rightarrow x_3$ ; applying that substitution to  $+(D(x_1), D(x_2))$  yields  $+(D(+x_1, x_2), D(x_3))$ .

## Rewrite Rules: Example

- Applying the Sum Rule to  $D(+(+x_1, x_2), x_3)$  requires the substitution  $x_1 \rightarrow +(x_1, x_2)$  and  $x_2 \rightarrow x_3$ ; applying that substitution to  $+(D(x_1), D(x_2))$  yields  $+(D(+x_1, x_2), D(x_3))$ .
- A second application of the Sum Rule yields  $+(+(D(x_1), D(x_2)), D(x_3))$ .

# Rewrite System

- A collection of rewrite rules is called a **rewrite system**.

# Rewrite System

- A collection of rewrite rules is called a **rewrite system**.
- Rewrite systems, when terminating and confluent, are an efficient way to take an expression and simplify it to an equivalent, canonical form.

# Creating Rewrite Systems

- Knuth and Bendix created a procedure for taking a set of axioms expressed in terms of equations between terms, along with a reduction order.

# Creating Rewrite Systems

- Knuth and Bendix created a procedure for taking a set of axioms expressed in terms of equations between terms, along with a reduction order.
- The idea is for each equation, to order the terms and create a rewrite rule from the equation.

# Creating Rewrite Systems

- Knuth and Bendix created a procedure for taking a set of axioms expressed in terms of equations between terms, along with a reduction order.
- The idea is for each equation, to order the terms and create a rewrite rule from the equation.
- It's possible that the resulting rewrite system may not be confluent; when it isn't, use the differing canonical forms to create a new axiom that will eventually become a rewrite rule.

# Creating Rewrite Systems

- Knuth and Bendix created a procedure for taking a set of axioms expressed in terms of equations between terms, along with a reduction order.
- The idea is for each equation, to order the terms and create a rewrite rule from the equation.
- It's possible that the resulting rewrite system may not be confluent; when it isn't, use the differing canonical forms to create a new axiom that will eventually become a rewrite rule.
- There is no guarantee that this procedure will end; it does in practice enter into an infinite loop sometimes.

# Creating Rewrite Systems

- Knuth and Bendix created a procedure for taking a set of axioms expressed in terms of equations between terms, along with a reduction order.
- The idea is for each equation, to order the terms and create a rewrite rule from the equation.
- It's possible that the resulting rewrite system may not be confluent; when it isn't, use the differing canonical forms to create a new axiom that will eventually become a rewrite rule.
- There is no guarantee that this procedure will end; it does in practice enter into an infinite loop sometimes.
- If at any point it is impossible to order the terms in an axiom, the procedure fails.

## Example: Groups

- A standard example that Knuth and Bendix explored was the set of axioms defining a group.

## Example: Groups

- A standard example that Knuth and Bendix explored was the set of axioms defining a group.
- Let  $M$  be an operator of degree 2 (multiplication),  $I$  an operator of degree 1 (left inverse), and  $E$  an operator of degree 0 (left identity).

## Example: Groups

- A standard example that Knuth and Bendix explored was the set of axioms defining a group.
- Let  $M$  be an operator of degree 2 (multiplication),  $I$  an operator of degree 1 (left inverse), and  $E$  an operator of degree 0 (left identity).
- A reduction ordering is defined using weights for the operators so that the number of multiplications in a product is minimized, as well as the number of copies of the identity appearing in a term.

## Example: Groups

- A standard example that Knuth and Bendix explored was the set of axioms defining a group.
- Let  $M$  be an operator of degree 2 (multiplication),  $I$  an operator of degree 1 (left inverse), and  $E$  an operator of degree 0 (left identity).
- A reduction ordering is defined using weights for the operators so that the number of multiplications in a product is minimized, as well as the number of copies of the identity appearing in a term.
- We start with the following three axioms:  $M(E, x_1) = x_1$  (left identity),  $M(I(x_1), x_1) = E$  (left inverse), and  $M(M(x_1, x_2), x_3) = M(x_1, M(x_2, x_3))$  (associativity).

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$
  - $M(x_1, E) \rightarrow x_1$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$
  - $M(x_1, E) \rightarrow x_1$
  - $I(E) \rightarrow E$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$
  - $M(x_1, E) \rightarrow x_1$
  - $I(E) \rightarrow E$
  - $I(I(x_1)) \rightarrow x_1$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$
  - $M(x_1, E) \rightarrow x_1$
  - $I(E) \rightarrow E$
  - $I(I(x_1)) \rightarrow x_1$
  - $M(x_1, I(x_1)) \rightarrow E$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$
  - $M(x_1, E) \rightarrow x_1$
  - $I(E) \rightarrow E$
  - $I(I(x_1)) \rightarrow x_1$
  - $M(x_1, I(x_1)) \rightarrow E$
  - $M(x_1, M(I(x_1), x_2)) \rightarrow x_2$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$
  - $M(x_1, E) \rightarrow x_1$
  - $I(E) \rightarrow E$
  - $I(I(x_1)) \rightarrow x_1$
  - $M(x_1, I(x_1)) \rightarrow E$
  - $M(x_1, M(I(x_1), x_2)) \rightarrow x_2$
  - $I(M(x_1, x_2)) \rightarrow M(I(x_2), I(x_1))$

## Example: Groups (Continued)

- After running the procedure on these three axioms, the result is a rewriting system of ten rules (the first three come from directly from the axioms).
  - $M(E, x_1) \rightarrow x_1$
  - $M(I(x_1), x_1) \rightarrow E$
  - $M(M(x_1, x_2), x_3) \rightarrow M(x_1, M(x_2, x_3))$
  - $M(I(x_1), M(x_1, x_2)) \rightarrow x_2$
  - $M(x_1, E) \rightarrow x_1$
  - $I(E) \rightarrow E$
  - $I(I(x_1)) \rightarrow x_1$
  - $M(x_1, I(x_1)) \rightarrow E$
  - $M(x_1, M(I(x_1), x_2)) \rightarrow x_2$
  - $I(M(x_1, x_2)) \rightarrow M(I(x_2), I(x_1))$
- Note that  $I$  is also a right inverse by the eighth rule, and  $E$  is also a right identity by the fifth rule.

# Automated Theorem Proving

- Algebraic systems are frequently expressed in terms of a finite set of axioms that are equations between terms.

# Automated Theorem Proving

- Algebraic systems are frequently expressed in terms of a finite set of axioms that are equations between terms.
- If it is possible to define a reduction order on the terms that yields a rewrite system, then any theorem that claims equality between two terms can be proven using the rewrite systems.

# Automated Theorem Proving

- Algebraic systems are frequently expressed in terms of a finite set of axioms that are equations between terms.
- If it is possible to define a reduction order on the terms that yields a rewrite system, then any theorem that claims equality between two terms can be proven using the rewrite systems.
- If the two terms have the same canonical form under the rewrite system, then the terms are always equal in that type of algebraic system.

# Automated Theorem Proving

- Algebraic systems are frequently expressed in terms of a finite set of axioms that are equations between terms.
- If it is possible to define a reduction order on the terms that yields a rewrite system, then any theorem that claims equality between two terms can be proven using the rewrite systems.
- If the two terms have the same canonical form under the rewrite system, then the terms are always equal in that type of algebraic system.
- Sadly, axioms about commutativity are not subject to this process; since there is a substitution that will switch the two sides of an axiom about commutativity, there is no reduction order that can be defined on such an axiom.

## Example: Monoid Presentations

- A **monoid** is an algebraic system with an associative multiplication and an identity.

## Example: Monoid Presentations

- A **monoid** is an algebraic system with an associative multiplication and an identity.
- A **monoid presentation** is a list of generators and a set of equations that the terms (expressed as products of generators) must satisfy.

## Example: Monoid Presentations

- A **monoid** is an algebraic system with an associative multiplication and an identity.
- A **monoid presentation** is a list of generators and a set of equations that the terms (expressed as products of generators) must satisfy.
- The generators are constant operators in this term system, with multiplication being the only non-constant operator.

## Example: Monoid Presentations

- A **monoid** is an algebraic system with an associative multiplication and an identity.
- A **monoid presentation** is a list of generators and a set of equations that the terms (expressed as products of generators) must satisfy.
- The generators are constant operators in this term system, with multiplication being the only non-constant operator.
- Group presentations are a special case of monoid presentations, where all generators are invertible, and the axioms include the products of generators with their inverses yielding the identity.

## Example: Monoid Presentations

- A **monoid** is an algebraic system with an associative multiplication and an identity.
- A **monoid presentation** is a list of generators and a set of equations that the terms (expressed as products of generators) must satisfy.
- The generators are constant operators in this term system, with multiplication being the only non-constant operator.
- Group presentations are a special case of monoid presentations, where all generators are invertible, and the axioms include the products of generators with their inverses yielding the identity.
- Rewrite systems can be formed from monoid presentations that facilitate computations in those systems; that is where my main interest has been for a number of years.

# Conclusion

- Simplification is a large part of the mathematical algorithms that we use.

# Conclusion

- Simplification is a large part of the mathematical algorithms that we use.
- We often work at the level of simplifying terms to equivalent canonical forms.

# Conclusion

- Simplification is a large part of the mathematical algorithms that we use.
- We often work at the level of simplifying terms to equivalent canonical forms.
- We can encode our rules for doing so in rewriting systems.

# Conclusion

- Simplification is a large part of the mathematical algorithms that we use.
- We often work at the level of simplifying terms to equivalent canonical forms.
- We can encode our rules for doing so in rewriting systems.
- Term rewriting is at the heart of most simplification.

# Conclusion

- Simplification is a large part of the mathematical algorithms that we use.
- We often work at the level of simplifying terms to equivalent canonical forms.
- We can encode our rules for doing so in rewriting systems.
- Term rewriting is at the heart of most simplification.
- Term rewriting can be used for automated theorem proving in algebraic systems.

# References

- Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

# References

- Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- Donald E. Knuth and P.B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

# References

- Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- Donald E. Knuth and P.B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
- Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, 2002.

# References

- Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- Donald E. Knuth and P.B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
- Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, 2002.
- Charles C. Sims, *Computation with Finitely Presented Groups*, Cambridge University Press, 1994.