Maude Summer School: Lecture 2

José Meseguer

University of Illinois at Urbana-Champaign and
Leverhulme visiting professor at King's College, London

Rewrite Proofs

Definition

Denoting by $\rightarrow^{+}_{\vec{E}}$ (resp. $\rightarrow^{*}_{\vec{E}}$) the transitive (resp. reflexive transitive) closure of $\rightarrow_{\vec{E}}$, a $(\Sigma, \vec{E})$-rewrite proof of $t \rightarrow^{*}_{\vec{E}} t'$ is, by definition, either:

Rewrite Proofs

Definition

Denoting by $\to^{+}_{\vec{E}}$ (resp. $\to^{*}_{\vec{E}}$) the transitive (resp. reflexive transitive) closure of $\to_{\vec{E}}$, a $(\Sigma, \vec{E})$-rewrite proof of $t \to^{*}_{\vec{E}} t'$ is, by definition, either:

- a 0-step rewrite $t \to^{*}_{\vec{E}} t$ for Σ-term t, i.e., $t \equiv t'$, or

Rewrite Proofs

Definition

Denoting by $\rightarrow^{+}_{\vec{E}}$ (resp. $\rightarrow^{*}_{\vec{E}}$) the transitive (resp. reflexive transitive) closure of $\rightarrow_{\vec{E}}$, a $(\Sigma, \vec{E})$-rewrite proof of $t \rightarrow^{*}_{\vec{E}} t'$ is, by definition, either:

- a 0-step rewrite $t \rightarrow^{*}_{\vec{E}} t$ for Σ-term t, i.e., $t \equiv t'$, or
- a sequence of $\vec{E}$-rewrite steps of the form

$$t \equiv t_0 \rightarrow_{\vec{E}} t_1 \rightarrow_{\vec{E}} t_2 \ldots t_{n-1} \rightarrow_{\vec{E}} t_n \equiv t'$$

  $n \geq 1$, witnessing $t \rightarrow^{+}_{\vec{E}} t'$,

Rewrite Proofs

Definition

Denoting by $\to^{+}_{\vec{E}}$ (resp. $\to^{*}_{\vec{E}}$) the transitive (resp. reflexive transitive) closure of $\to_{\vec{E}}$, a $(\Sigma, \vec{E})$-rewrite proof of $t \to^{*}_{\vec{E}} t'$ is, by definition, either:

- a 0-step rewrite $t \to^{*}_{\vec{E}} t$ for Σ-term t, i.e., $t \equiv t'$, or
- a sequence of $\vec{E}$-rewrite steps of the form

$$t \equiv t_0 \to_{\vec{E}} t_1 \to_{\vec{E}} t_2 \ldots t_{n-1} \to_{\vec{E}} t_n \equiv t'$$

  $n \geq 1$, witnessing $t \to^{+}_{\vec{E}} t'$, where $u \equiv v$ denotes syntactic equality.

Rewrite Proofs

Definition

Denoting by $\rightarrow^{+}_{\vec{E}}$ (resp. $\rightarrow^{*}_{\vec{E}}$) the transitive (resp. reflexive transitive) closure of $\rightarrow_{\vec{E}}$, a $(\Sigma, \vec{E})$-rewrite proof of $t \rightarrow^{*}_{\vec{E}} t'$ is, by definition, either:

- a 0-step rewrite $t \rightarrow^{*}_{\vec{E}} t$ for Σ-term t, i.e., $t \equiv t'$, or
- a sequence of $\vec{E}$-rewrite steps of the form

$$t \equiv t_0 \rightarrow_{\vec{E}} t_1 \rightarrow_{\vec{E}} t_2 \ldots t_{n-1} \rightarrow_{\vec{E}} t_n \equiv t'$$

$n \geq 1$, witnessing $t \rightarrow^{+}_{\vec{E}} t'$, where $u \equiv v$ denotes syntactic equality.

Remark: The rules R in a term rewriting system (Σ, R) need not be oriented equations $\vec{E}$. Then, a rewrite proof is just written as: $t \rightarrow^{*}_{R} t'$.

Rewrite Proofs

Definition

Denoting by $\to^{+}_{\vec{E}}$ (resp. $\to^{*}_{\vec{E}}$) the transitive (resp. reflexive transitive) closure of $\to_{\vec{E}}$, a $(\Sigma, \vec{E})$-rewrite proof of $t \to^{*}_{\vec{E}} t'$ is, by definition, either:

- a 0-step rewrite $t \to^{*}_{\vec{E}} t$ for Σ-term t, i.e., $t \equiv t'$, or
- a sequence of $\vec{E}$-rewrite steps of the form

$$t \equiv t_0 \to_{\vec{E}} t_1 \to_{\vec{E}} t_2 \ldots t_{n-1} \to_{\vec{E}} t_n \equiv t'$$

$n \geq 1$, witnessing $t \to^{+}_{\vec{E}} t'$, where $u \equiv v$ denotes syntactic equality.

Remark: The rules R in a term rewriting system (Σ, R) need not be oriented equations $\vec{E}$. Then, a rewrite proof is just written as: $t \to^{*}_{R} t'$. Non-equational rules R will be treated in Lectures 3–4.

The Equality Relation and Equational Proofs

The notion of an equational proof, that is, a sequence of steps of replacement of equals by equals using equations E, is a trivial instance of the notion of a rewrite proof.

The Equality Relation and Equational Proofs

The notion of an equational proof, that is, a sequence of steps of replacement of equals by equals using equations E, is a trivial instance of the notion of a rewrite proof.

Given an equational theory (Σ, E), all we need to do is to consider proofs in the term rewriting system $(\Sigma, \overrightarrow{E} \cup \overleftarrow{E})$, where, by definition:

The Equality Relation and Equational Proofs

The notion of an equational proof, that is, a sequence of steps of replacement of equals by equals using equations E, is a trivial instance of the notion of a rewrite proof.

Given an equational theory (Σ, E), all we need to do is to consider proofs in the term rewriting system $(\Sigma, \overrightarrow{E} \cup \overleftarrow{E})$, where, by definition:

- $\overrightarrow{E}$ is the set of left-to-right orientations $\overrightarrow{E} = \{t \to t' \mid t = t' \in E\}$; and

The Equality Relation and Equational Proofs

The notion of an equational proof, that is, a sequence of steps of replacement of equals by equals using equations E, is a trivial instance of the notion of a rewrite proof.

Given an equational theory (Σ, E), all we need to do is to consider proofs in the term rewriting system $(\Sigma, \overrightarrow{E} \cup \overleftarrow{E})$, where, by definition:

- $\overrightarrow{E}$ is the set of left-to-right orientations $\overrightarrow{E} = \{t \to t' \mid t = t' \in E\}$; and
- $\overleftarrow{E}$ is the set of right-to-left orientations $\overleftarrow{E} = \{t' \to t \mid t = t' \in E\}$.

The Equality Relation and Equational Proofs (II)

Definition

Given an equational theory (Σ, E), an E-equality step is, by definition, a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite step $u \rightarrow_{(\overrightarrow{E} \cup \overleftarrow{E})} v$.

The Equality Relation and Equational Proofs (II)

Definition

Given an equational theory (Σ, E), an E-equality step is, by definition, a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite step $u \rightarrow_{(\overrightarrow{E} \cup \overleftarrow{E})} v$.

The relation $\rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})}$ is called the E-equality relation, often abbreviated to $=_E$. It is also called the relation of equality modulo E.

The Equality Relation and Equational Proofs (II)

Definition

Given an equational theory (Σ, E), an E-equality step is, by definition, a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite step $u \rightarrow_{(\overrightarrow{E} \cup \overleftarrow{E})} v$.

The relation $\rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})}$ is called the E-equality relation, often abbreviated to $=_E$. It is also called the relation of equality modulo E.

A (Σ, E)-equality proof of $t =_E t'$ is just a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite proof $t \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t'$.

The Equality Relation and Equational Proofs (II)

Definition

Given an equational theory (Σ, E), an E-equality step is, by definition, a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite step $u \rightarrow_{(\overrightarrow{E} \cup \overleftarrow{E})} v$.

The relation $\rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})}$ is called the E-equality relation, often abbreviated to $=_E$. It is also called the relation of equality modulo E.

A (Σ, E)-equality proof of $t =_E t'$ is just a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite proof $t \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t'$. But then we also have: $t' \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t$ (symmetry).

The Equality Relation and Equational Proofs (II)

Definition

Given an equational theory (Σ, E), an E-equality step is, by definition, a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite step $u \rightarrow_{(\overrightarrow{E} \cup \overleftarrow{E})} v$.

The relation $\rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})}$ is called the E-equality relation, often abbreviated to $=_E$. It is also called the relation of equality modulo E.

A (Σ, E)-equality proof of $t =_E t'$ is just a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite proof $t \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t'$. But then we also have: $t' \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t$ (symmetry).

We write $(\Sigma, E) \vdash t = t'$ iff $t \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t'$, and say that E proves the equality $t = t'$.

The Equality Relation and Equational Proofs (II)

Definition

Given an equational theory (Σ, E), an E-equality step is, by definition, a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite step $u \rightarrow_{(\overrightarrow{E} \cup \overleftarrow{E})} v$.

The relation $\rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})}$ is called the E-equality relation, often abbreviated to $=_E$. It is also called the relation of equality modulo E.

A (Σ, E)-equality proof of $t =_E t'$ is just a $(\overrightarrow{E} \cup \overleftarrow{E})$-rewrite proof $t \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t'$. But then we also have: $t' \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t$ (symmetry).

We write $(\Sigma, E) \vdash t = t'$ iff $t \rightarrow^*_{(\overrightarrow{E} \cup \overleftarrow{E})} t'$, and say that E proves the equality $t = t'$. By definition, $t =_E t'$ is an equivalence relation.

Term Rewriting Modulo Axioms

Certain equations are intrinsically problematic for term rewriting.

Term Rewriting Modulo Axioms

Certain equations are intrinsically problematic for term rewriting. For example, the commutativity equation $x + y = y + x$ is intrinsically problematic for rewriting because:

Term Rewriting Modulo Axioms

Certain equations are intrinsically problematic for term rewriting. For example, the commutativity equation $x + y = y + x$ is intrinsically problematic for rewriting because:

- we do not obtain a simpler term, but only a "mirror image" of the original term; for example, $(x * 7) + (0 * y)$ is rewritten to $(0 * y) + (x * 7)$; and

Term Rewriting Modulo Axioms

Certain equations are intrinsically problematic for term rewriting. For example, the commutativity equation $x + y = y + x$ is intrinsically problematic for rewriting because:

- we do not obtain a simpler term, but only a "mirror image" of the original term; for example, $(x * 7) + (0 * y)$ is rewritten to $(0 * y) + (x * 7)$; and
- even worse, we can easily loop when applying this equation, as in the infinite, alternating sequence

Term Rewriting Modulo Axioms

Certain equations are intrinsically problematic for term rewriting. For example, the commutativity equation $x + y = y + x$ is intrinsically problematic for rewriting because:

- we do not obtain a simpler term, but only a "mirror image" of the original term; for example, $(x * 7) + (0 * y)$ is rewritten to $(0 * y) + (x * 7)$; and
- even worse, we can easily loop when applying this equation, as in the infinite, alternating sequence

$$(x*7)+(0*y) \to_E (0*y)+(x*7) \to_E (x*7)+(0*y) \to_E \dots$$

Term Rewriting Modulo Axioms

Certain equations are intrinsically problematic for term rewriting. For example, the commutativity equation $x + y = y + x$ is intrinsically problematic for rewriting because:

- we do not obtain a simpler term, but only a "mirror image" of the original term; for example, $(x * 7) + (0 * y)$ is rewritten to $(0 * y) + (x * 7)$; and
- even worse, we can easily loop when applying this equation, as in the infinite, alternating sequence

$$(x*7)+(0*y) \to_E (0*y)+(x*7) \to_E (x*7)+(0*y) \to_E \ldots$$

The solution to this problem is to build in certain, commonly occurring equational axioms like commutativity,

Term Rewriting Modulo Axioms

Certain equations are intrinsically problematic for term rewriting. For example, the commutativity equation $x + y = y + x$ is intrinsically problematic for rewriting because:

- we do not obtain a simpler term, but only a "mirror image" of the original term; for example, $(x * 7) + (0 * y)$ is rewritten to $(0 * y) + (x * 7)$; and
- even worse, we can easily loop when applying this equation, as in the infinite, alternating sequence

$$(x*7)+(0*y) \rightarrow_E (0*y)+(x*7) \rightarrow_E (x*7)+(0*y) \rightarrow_E \ldots$$

The solution to this problem is to build in certain, commonly occurring equational axioms like commutativity, so that rewriting takes place modulo such axioms.

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g.,

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$ and an oriented part, e.g., $\vec{E}_0 = \{(x + y) + z \to x + (y + z), x + 0 \to x, x * 1 \to x, x * (y + z) \to (x * y) + (x * z)\}$.

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$ and an oriented part, e.g., $\vec{E}_0 = \{(x + y) + z \to x + (y + z), x + 0 \to x, x * 1 \to x, x * (y + z) \to (x * y) + (x * z)\}$. Then, we can rewrite with the oriented equations in $\vec{E}_0$ applying them, not just to the given term t, but to any other term t' which is provably equal to t by the commutativity equations C.

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$ and an oriented part, e.g., $\vec{E}_0 = \{(x + y) + z \to x + (y + z), x + 0 \to x, x * 1 \to x, x * (y + z) \to (x * y) + (x * z)\}$. Then, we can rewrite with the oriented equations in $\vec{E}_0$ applying them, not just to the given term t, but to any other term t' which is provably equal to t by the commutativity equations C.

This more powerful rewrite relation is called rewriting modulo C, denoted $\to_{E_0/C}$.

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$ and an oriented part, e.g., $\vec{E}_0 = \{(x + y) + z \to x + (y + z), x + 0 \to x, x * 1 \to x, x * (y + z) \to (x * y) + (x * z)\}$. Then, we can rewrite with the oriented equations in $\vec{E}_0$ applying them, not just to the given term t, but to any other term t' which is provably equal to t by the commutativity equations C.

This more powerful rewrite relation is called rewriting modulo C, denoted $\to_{E_0/C}$. For example, we can simplify the expression

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$ and an oriented part, e.g., $\vec{E}_0 = \{(x + y) + z \to x + (y + z), x + 0 \to x, x * 1 \to x, x * (y + z) \to (x * y) + (x * z)\}$. Then, we can rewrite with the oriented equations in $\vec{E}_0$ applying them, not just to the given term t, but to any other term t' which is provably equal to t by the commutativity equations C.

This more powerful rewrite relation is called rewriting modulo C, denoted $\to_{E_0/C}$. For example, we can simplify the expression $((0 + x) * ((1 * y) + 7)) + z$ to $(x * y) + ((x * 7) + z)$

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$ and an oriented part, e.g., $\vec{E_0} = \{(x + y) + z \to x + (y + z), x + 0 \to x, x * 1 \to x, x * (y + z) \to (x * y) + (x * z)\}$. Then, we can rewrite with the oriented equations in $\vec{E_0}$ applying them, not just to the given term t, but to any other term t' which is provably equal to t by the commutativity equations C.

This more powerful rewrite relation is called rewriting modulo C, denoted $\to_{E_0/C}$. For example, we can simplify the expression $((0 + x) * ((1 * y) + 7)) + z$ to $(x * y) + ((x * 7) + z)$ in just four steps with $\to_{E_0/C}$ as follows:

Term Rewriting Modulo Axioms (II)

For example, we can decompose some equations E into a built-in, commutative part, e.g., $C = \{x + y = y + x, x * y = y * x\}$ and an oriented part, e.g., $\vec{E_0} = \{(x + y) + z \to x + (y + z), x + 0 \to x, x * 1 \to x, x * (y + z) \to (x * y) + (x * z)\}$. Then, we can rewrite with the oriented equations in $\vec{E_0}$ applying them, not just to the given term t, but to any other term t' which is provably equal to t by the commutativity equations C.

This more powerful rewrite relation is called rewriting modulo C, denoted $\to_{E_0/C}$. For example, we can simplify the expression $((0 + x) * ((1 * y) + 7)) + z$ to $(x * y) + ((x * 7) + z)$ in just four steps with $\to_{E_0/C}$ as follows:

$$((0{+}x){*}((1{*}y){+}7)){+}z \to_{E_0/C} (x{*}((1{*}y){+}7)){+}z \to_{E_0/C} (x{*}(y{+}7)){+}z \to_{E_0/C} ((x * y) + (x * 7)) + z \to_{E_0/C} (x * y) + ((x * 7) + z)$$

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity?

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x + y) + z = x + (y + z)$ has no looping problems;

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x + y) + z = x + (y + z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x+y)+z = x+(y+z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

For example, we can simplify to 0 the term $((x+y)+z) + -(y+(z+x))$ in one step of rewriting modulo the following set AC of associativity and commutativity axioms for $\_+\_$ and $\_*\_$,

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x + y) + z = x + (y + z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

For example, we can simplify to 0 the term $((x + y) + z) + -(y + (z + x))$ in one step of rewriting modulo the following set AC of associativity and commutativity axioms for $\_+\_$ and $\_*\_$, $AC = \{x + y = y + x, x * y = y * x, (x + y) + z = x + (y + z), (x * y) * z = x * (y * z)\}$,

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x + y) + z = x + (y + z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

For example, we can simplify to 0 the term $((x + y) + z) + -(y + (z + x))$ in one step of rewriting modulo the following set AC of associativity and commutativity axioms for $\_ + \_$ and $\_ * \_$, $AC = \{x + y = y + x, x * y = y * x, (x + y) + z = x + (y + z), (x * y) * z = x * (y * z)\}$, using the single equation $E_0 = \{x + -x = 0\}$ oriented as the rule $\vec{E_0} = \{x + -x \to 0\}$.

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x+y)+z = x+(y+z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

For example, we can simplify to 0 the term $((x+y)+z) + -(y+(z+x))$ in one step of rewriting modulo the following set AC of associativity and commutativity axioms for $\_+\_$ and $\_*\_$, $AC = \{x+y = y+x, x*y = y*x, (x+y)+z = x+(y+z), (x*y)*z = x*(y*z)\}$, using the single equation $E_0 = \{x + -x = 0\}$ oriented as the rule $\vec{E}_0 = \{x + -x \to 0\}$.

$$((x+y)+z) + -(y+(z+x)) \to_{\vec{E}_0,/AC} 0.$$

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x+y)+z = x+(y+z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

For example, we can simplify to 0 the term $((x+y)+z) + -(y+(z+x))$ in one step of rewriting modulo the following set AC of associativity and commutativity axioms for $\_+\_$ and $\_*\_$, $AC = \{x+y = y+x, x*y = y*x, (x+y)+z = x+(y+z), (x*y)*z = x*(y*z)\}$, using the single equation $E_0 = \{x + -x = 0\}$ oriented as the rule $\vec{E_0} = \{x + -x \to 0\}$.

$$((x+y)+z) + -(y+(z+x)) \to_{\vec{E_0},/AC} 0.$$

Rewriting modulo AC:

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x+y)+z = x+(y+z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

For example, we can simplify to 0 the term $((x+y)+z) + -(y+(z+x))$ in one step of rewriting modulo the following set AC of associativity and commutativity axioms for $\_+\_$ and $\_*\_$, $AC = \{x+y = y+x, x*y = y*x, (x+y)+z = x+(y+z), (x*y)*z = x*(y*z)\}$, using the single equation $E_0 = \{x + -x = 0\}$ oriented as the rule $\vec{E}_0 = \{x + -x \to 0\}$.

$$((x+y)+z) + -(y+(z+x)) \to_{\vec{E}_0,/AC} 0.$$

Rewriting modulo AC: (i) the order of the arguments does not matter (by C), and

Term Rewriting Modulo Axioms (III)

But why stopping with commutativity? How about associativity? An associativity (A) equation such as $(x+y)+z = x+(y+z)$ has no looping problems; but parentheses around associative operators are a nuisance and can block the application of many equations.

For example, we can simplify to 0 the term $((x+y)+z) + -(y+(z+x))$ in one step of rewriting modulo the following set AC of associativity and commutativity axioms for $\_+\_$ and $\_*\_$, $AC = \{x+y = y+x, x*y = y*x, (x+y)+z = x+(y+z), (x*y)*z = x*(y*z)\}$, using the single equation $E_0 = \{x + -x = 0\}$ oriented as the rule $\vec{E}_0 = \{x + -x \to 0\}$.

$$((x+y)+z) + -(y+(z+x)) \to_{\vec{E}_0,/AC} 0.$$

Rewriting modulo AC: (i) the order of the arguments does not matter (by C), and (ii) parentheses do not matter (by A).

Rewrite Theories

Likewise, we could also build in the unit element axioms $U = \{x + 0 = x, x * 1 = x\}$.

Rewrite Theories

Likewise, we could also build in the unit element axioms $U = \{x + 0 = x, x * 1 = x\}$. Or any combination of C, and/or A, and/or U axioms could be built in.

Rewrite Theories

Likewise, we could also build in the unit element axioms $U = \{x + 0 = x, x * 1 = x\}$. Or any combination of C, and/or A, and/or U axioms could be built in. Maude supports all such combinations.

Rewrite Theories

Likewise, we could also build in the unit element axioms $U = \{x + 0 = x, x * 1 = x\}$. Or any combination of C, and/or A, and/or U axioms could be built in. Maude supports all such combinations.

In fact, the idea of building in a set B of equational axioms, so that we rewrite with a set of rules R modulo B, is entirely general, and is associated to the notion of a rewrite theory.

Rewrite Theories

Likewise, we could also build in the unit element axioms $U = \{x + 0 = x, x * 1 = x\}$. Or any combination of C, and/or A, and/or U axioms could be built in. Maude supports all such combinations.

In fact, the idea of building in a set B of equational axioms, so that we rewrite with a set of rules R modulo B, is entirely general, and is associated to the notion of a rewrite theory.

Definition

Let Σ be an order-sorted signature. A rewrite theory is a triple (Σ, B, R), where B is a set of Σ-equations, and R is a set of Σ-rewrite rules.

Rewrite Theories

Likewise, we could also build in the unit element axioms $U = \{x + 0 = x, x * 1 = x\}$. Or any combination of C, and/or A, and/or U axioms could be built in. Maude supports all such combinations.

In fact, the idea of building in a set B of equational axioms, so that we rewrite with a set of rules R modulo B, is entirely general, and is associated to the notion of a rewrite theory.

Definition

Let Σ be an order-sorted signature. A rewrite theory is a triple (Σ, B, R), where B is a set of Σ-equations, and R is a set of Σ-rewrite rules.

Rewriting with R modulo B can then be formalized as follows:

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \rightarrow_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that:

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \rightarrow_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$,

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \rightarrow_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$, (ii) $u' \rightarrow_R v'$,

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \to_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$, (ii) $u' \to_R v'$, and (iii) $v' =_B v$.

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \to_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$, (ii) $u' \to_R v'$, and (iii) $v' =_B v$. That is, we have:

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \rightarrow_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$, (ii) $u' \rightarrow_R v'$, and (iii) $v' =_B v$. That is, we have:

$$u =_B u' \rightarrow_R v' =_B v.$$

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \rightarrow_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$, (ii) $u' \rightarrow_R v'$, and (iii) $v' =_B v$. That is, we have:

$$u =_B u' \rightarrow_R v' =_B v.$$

We denote by $\rightarrow^0_{R/B}$ the relation $=_B$, called the 0-step R-rewrite relation modulo B,

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \to_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$, (ii) $u' \to_R v'$, and (iii) $v' =_B v$. That is, we have:

$$u =_B u' \to_R v' =_B v.$$

We denote by $\to^0_{R/B}$ the relation $=_B$, called the 0-step R-rewrite relation modulo B, by $\to^+_{R/B}$ the transitive closure of $\to_{R/B}$,

Rewriting Modulo B

Definition

Let (Σ, B, R) be a rewrite theory. Then the R-rewrite relation modulo B, denoted $u \rightarrow_{R/B} v$, holds between Σ-terms u, v iff there exist Σ-terms u', v' such that: (i) $u =_B u'$, (ii) $u' \rightarrow_R v'$, and (iii) $v' =_B v$. That is, we have:

$$u =_B u' \rightarrow_R v' =_B v.$$

We denote by $\rightarrow^{0}_{R/B}$ the relation $=_B$, called the 0-step R-rewrite relation modulo B, by $\rightarrow^{+}_{R/B}$ the transitive closure of $\rightarrow_{R/B}$, and by $\rightarrow^{\circledast}_{R/B}$ the relation $\rightarrow^{+}_{R/B} \cup =_B$.

Rewrite Proofs Modulo B

Definition

An R-rewrite proof modulo B of $u \rightarrow^{\circledast}_{R/B} v$, is either:

Rewrite Proofs Modulo B

Definition

An R-rewrite proof modulo B of $u \rightarrow^{\circledast}_{R/B} v$, is either:

- a *0-step* R-rewrite modulo B of the form $u \rightarrow^{0}_{R/B} v$, so that, by definition, $u =_B v$, for Σ-terms u, v, or

Rewrite Proofs Modulo B

Definition

An R-rewrite proof modulo B of $u \rightarrow^{\circledast}_{R/B} v$, is either:

- a *0-step* R-rewrite modulo B of the form $u \rightarrow^{0}_{R/B} v$, so that, by definition, $u =_B v$, for Σ-terms u, v, or
- a sequence of R-rewrite steps modulo B of the form:

Rewrite Proofs Modulo B

Definition

An R-rewrite proof modulo B of $u \to^{\circledast}_{R/B} v$, is either:

- a *0-step* R-rewrite modulo B of the form $u \to^{0}_{R/B} v$, so that, by definition, $u =_B v$, for Σ-terms u, v, or
- a sequence of R-rewrite steps modulo B of the form:

$$u \equiv u_0 \to_{R/B} u_1 \to_{R/B} u_2 \ldots u_{n-1} \to_{R/B} u_n \equiv v$$

with $n \geq 1$,

Rewrite Proofs Modulo B

Definition

An R-rewrite proof modulo B of $u \rightarrow^{\circledast}_{R/B} v$, is either:

- a *0-step* R-rewrite modulo B of the form $u \rightarrow^{0}_{R/B} v$, so that, by definition, $u =_B v$, for Σ-terms u, v, or
- a sequence of R-rewrite steps modulo B of the form:

$$u \equiv u_0 \rightarrow_{R/B} u_1 \rightarrow_{R/B} u_2 \ldots u_{n-1} \rightarrow_{R/B} u_n \equiv v$$

with $n \geq 1$, witnessing $u \rightarrow^{+}_{R/B} v$.

Examples of Equational Simplification Modulo B

Lists modulo associativity and identity (AU), with membership:

```
fmod LIST-AU is
  protecting NAT .
  sort List .
  subsort Nat < List .
  op nil : -> List [ctor] .
  op _;_ : List List -> List [assoc id: nil ctor] .
  op _in_ : Nat List -> Bool .
  var N : Nat .  vars L L' : List .
  eq N in L ; N ; L' = true .
  eq N in L = false [owise] .
endfm
reduce in LIST-AU : 7 in 3 ; 4 ; 9 .
result Bool: false
==========================================
reduce in LIST-AU : 7 in 4 ; 3 ; 7 .
result Bool: true
```

Examples of Equational Simplification Modulo B (II)

Lists modulo associativity (A) with membership. More patterns are need.

```
fmod LIST-A is
  protecting NAT .  sort List .  subsort Nat < List .
  op nil : -> List [ctor] .
  op _;_ : List List -> List [assoc ctor] .
  op _in_ : Nat List -> Bool .
  var N : Nat .  vars L L' : List .
  eq nil ; L = L .
  eq L ; nil = L .
  eq N in N = true .
  eq N in N ; L = true .
  eq N in L ; N = true .
  eq N in L ; N ; L' = true .
  eq N in L = false [owise] .
endfm

reduce in LIST-A : 7 in 4 ; 3 ; 7 .
result Bool: true
```

Examples of Equational Simplification Modulo *B* (III)

Multisets modulo associativity, commutativity, and identity (*ACU*).

```
fmod MSET-ACU is
  protecting NAT .
  sort MSet .
  subsort Nat < MSet .
  op nil : -> MSet [ctor] .
  op _;_ : MSet MSet -> MSet [assoc comm id: nil ctor] .
  op _in_ : Nat MSet -> Bool .
  var N : Nat .  var S : MSet .
  eq N in N ; S = true .
  eq N in S = false [owise] .
endfm
reduce in MSET-ACU : 7 in 3 ; 4 ; 9 .
result Bool: false
==========================================
reduce in MSET-ACU : 7 in 4 ; 3 ; 7 .
result Bool: true
```

Examples of Equational Simplification Modulo B (IV)

Multisets modulo associativity and commutativity (AC): more patterns needed.

```
fmod MSET-AC is
  protecting NAT .
  sort MSet .            subsort Nat < MSet .
  op nil : -> MSet [ctor] .
  op _;_ : MSet MSet -> MSet [assoc comm ctor] .
  op _in_ : Nat MSet -> Bool .
  var N : Nat .   var S : MSet .
  eq nil ; S = S .
  eq N in N = true .
  eq N in N ; S = true .
  eq N in S = false [owise] .
endfm
reduce in MSET-AC : 7 in 3 ; 4 ; 9 .
result Bool: false
==========================================
reduce in MSET-AC : 7 in 4 ; 3 ; 7 .
result Bool: true
```

Examples of Equational Simplification Modulo B (V)

Sets of natural numbers using identity and idempotency equations.

```
fmod NAT-SET is protecting NAT .
  sort NatSet .
  subsort Nat < NatSet .
  op mt : -> NatSet [ctor] .
  op _ _ : NatSet NatSet -> NatSet [ctor assoc comm] . *** set union
  op _/\ _ : NatSet NatSet -> NatSet [assoc comm] .    *** intersection
  vars X Y : NatSet .       var N : Nat .
  eq mt X = X .                                        *** identity
  eq X X = X .                                         *** idempotency
  eq N /\ N = N .
  eq N /\ (N X) = N .
  eq (N X) /\ (N Y) = N (X /\ Y) .
  eq X /\ Y = mt [owise] .
endfm
Maude> red (1 2 3 4 5) /\ (3 4 5 6 7) .
result NatSet: 3 4 5
```

Caveats on Equational Simplification Modulo B

Equational simplification modulo identity is trickier. For example, the innocent-looking idempotency equation in

Caveats on Equational Simplification Modulo B

Equational simplification modulo identity is trickier. For example, the innocent-looking idempotency equation in

```
fmod NAT-SET' is protecting NAT .
  sort NatSet .
  subsort Nat < NatSet .
  op mt : -> NatSet [ctor] .
  op _ _ : NatSet NatSet -> NatSet [ctor assoc comm id: mt] .
  var X : NatSet .
  eq X X = X .
endfm
```

Caveats on Equational Simplification Modulo B

Equational simplification modulo identity is trickier. For example, the innocent-looking idempotency equation in

```
fmod NAT-SET' is protecting NAT .
  sort NatSet .
  subsort Nat < NatSet .
  op mt : -> NatSet [ctor] .
  op _ _ : NatSet NatSet -> NatSet [ctor assoc comm id: mt] .
  var X : NatSet .
  eq X X = X .
endfm
```

is nonterminating, since we have,

Caveats on Equational Simplification Modulo B

Equational simplification modulo identity is trickier. For example, the innocent-looking idempotency equation in

```
fmod NAT-SET' is protecting NAT .
  sort NatSet .
  subsort Nat < NatSet .
  op mt : -> NatSet [ctor] .
  op _ _ : NatSet NatSet -> NatSet [ctor assoc comm id: mt] .
  var X : NatSet .
  eq X X = X .
endfm
```

is nonterminating, since we have,

$$\texttt{mt} =_{ACU} \texttt{mt mt} \longrightarrow_E \texttt{mt} =_{ACU} \texttt{mt mt} \longrightarrow_E \ldots$$

Caveats on Equational Simplification Modulo B (II)

Nontermination can be avoided by giving instead a more careful equation, where we restrict idempotency to pairs of elements (yet, with the same effect, sice this ensures that all repeated elements will be eliminated) by means of the (now terminating) equation,

Caveats on Equational Simplification Modulo B (II)

Nontermination can be avoided by giving instead a more careful equation, where we restrict idempotency to pairs of elements (yet, with the same effect, sice this ensures that all repeated elements will be eliminated) by means of the (now terminating) equation,

```
var N : Nat .
eq N N = N .
```

Caveats on Equational Simplification Modulo B (II)

Nontermination can be avoided by giving instead a more careful equation, where we restrict idempotency to pairs of elements (yet, with the same effect, sice this ensures that all repeated elements will be eliminated) by means of the (now terminating) equation,

```
var N : Nat .
eq N N = N .
```

Another alternative is to declare:

Caveats on Equational Simplification Modulo B (II)

Nontermination can be avoided by giving instead a more careful equation, where we restrict idempotency to pairs of elements (yet, with the same effect, sice this ensures that all repeated elements will be eliminated) by means of the (now terminating) equation,

```
var N : Nat .
eq N N = N .
```

Another alternative is to declare:

```
sort NatSet NeNatSet .
subsort Nat < NeNatSet < NatSet .
op mt : -> NatSet [ctor] .
op _ _ : NatSet NatSet -> NatSet [ctor assoc comm id: mt] .
op _ _ : NeNatSet NeNatSet -> NeNatSet [ctor assoc comm id: mt]
var X : NeNatSet .
eq X X = X .
```