Associative-Commutative Reduction Orderings

Leo Bachmair

MPI–I–91–209 November 1991

Author's Address

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York 11794, U.S.A., leo@sbcs.sunysb.edu

Publication Notes

The present report has been submitted for publication elsewhere and will be copyrighted if accepted.

Acknowledgements

I wish to thank Harald Ganzinger for his comments on a preliminary version of this paper.
This research was supported in part by the National Science Foundation under grant CCR-8901322.

Abstract

Rewrite systems are sets of directed equations used to compute by repeatedly replacing subterms in a given expression by equal terms until a simplest form possible (a normal form) is obtained. If a rewrite system is terminating (i.e., allows no infinite sequence of rewrites), then every expression has a normal form. A variety of orderings, called reduction orderings, have been designed for proving termination, but most of them are not applicable to extended rewrite systems, where rewrites take into account inherent properties of given functions such as associativity and commutativity. In this paper we show how an ordering represented as a schematic rewrite system—the lexicographic path ordering—can be systematically modified into an ordering compatible with associativity and commutativity.

Keywords

Termination orderings, rewrite systems, automated deduction.

1 Preliminaries

We consider first-order terms built from given function symbols and variables. With each function symbol f we associate a set $\alpha(f) \subseteq \mathbf{N}$ that indicates the number of arguments f may take. A syntactically well-formed term is an expression $f(X)$, where X is a sequence of terms $t_1, \ldots, t_n$ with $n \in \alpha(f)$. The length of a term $f(X)$ is defined by $|f(X)| = 1 + \sum_{t \in X} |t|$. We say that s is a *subterm* of $t = f(X)$ if either $s = f(Y)$, for some sequence $Y \subseteq X$,[1] or else s is a subterm of some term $u \in X$. If $s \neq t$ we speak of a *proper subterm*. For example, if $\{2, 3\} \subseteq \alpha(f)$, then $f(a, c)$ is a subterm of $f(a, b, c)$.

A *rewrite rule* is a pair of terms, written $s \to t$. A *rewrite system* is a set of rewrite rules. A binary relation $\to$ on terms is called a *rewrite relation* if $s \to t$ implies $f(X, s\sigma, Y) \to f(X, t\sigma, Y)$, for all terms s, t, and $f(X, s, Y)$, and substitutions σ. By $\leftarrow$ we denote the inverse of $\to$ and by $\to^+$ and $\to^*$ the transitive and transitive-reflexive closure of $\to$, respectively. By $\to_R$ we denote the smallest rewrite relation containing the rewrite system R. We say that t is *derivable from* s if $s \to_R^+ t$; and write $s \to_R^! t$ if $s \to_R^* t$ and there exists no term t' such that $t \to_R t'$ (then t is also called a *normal form*). A rewrite system R is called *terminating* if there exists no infinite sequence $t_1 \to_R t_2 \to_R t_3 \cdots$.

An (*rewrite*) *ordering* is an irreflexive and transitive (rewrite) relation. Well-founded rewrite orderings are called *reduction orderings*.[2] Evidently, a rewrite system terminates if and only if it is contained in some reduction ordering. A widely used ordering is the *lexicographic path ordering*, which can be defined as the derivability relation induced by the following recursively defined rewrite system LPO:

$$\begin{array}{rcll} f(X) & \to & s & \text{if } s \in X \\ f(X) & \to & g(Y) & \text{if } f \succ g \text{ and } f(X) \to_{LPO} t, \text{ for all } t \in Y \\ f(X, s, Y) & \to & f(X, t, Z) & \text{if } s \to_{LPO} t,\ |Y| = |Z|, \text{ and } f(X, s, Y) \to_{LPO} u, \text{ for all } u \in Z \end{array}$$

where $\succ$ is a well-founded ordering on function symbols (called a *precedence*). Rules of the form $f(X) \to s$ are called *subterm rules*; rules $f(X) \to g(Y)$, *combination rules*; and rules $f(X, s, Y) \to f(X, t, Z)$, *lexicographic rules*.

Rewrite systems are often used to model functions that satisfy certain identities. For instance, the associativity and commutativity of a function symbol f can be described by the rewrite rules

$$\begin{array}{rcl} f(s, f(t, u)) & \to & f(f(s, t), u) \\ f(f(s, t), u) & \to & f(s, f(t, u)) \\ f(s, t) & \to & f(t, s) \end{array}$$

which induce a non-terminating rewrite relation and therefore are dealt with in a special way.

The rewrite system R/S consists of all rules $u \to v$ such that $u \to_S^* u' \to_R v' \to_S^* v$, for some terms u' and v'. We consider the problem of proving termination of rewrite systems R/AC,

[1] We write $Y \subseteq X$ to indicate that X is a sequence $t_1, \ldots, t_n$ and Y a sequence $t_{i_1}, \ldots, t_{i_k}$, where $1 \leq i_1 < \cdots < i_k \leq n$

[2] For a detailed discussion of reduction orderings see the survey by N. Dershowitz, *J. Symbolic Computation* **3**(1987):69-116.

where AC is a set of associativity and commutativity rules. A reduction ordering $\succ$ is called *AC-compatible* if $s \leftrightarrow^*_{AC} u \succ v \leftrightarrow^*_{AC} t$ implies $s \succ t$, for all terms s, t, u, and v. Since the two relations $\rightarrow_{AC}$ and $\leftrightarrow_{AC}$ are identical, a rewrite system R/AC terminates if and only if R is contained in some AC-compatible reduction ordering.

Unfortunately, most reduction orderings are not AC-compatible. For example, if $f \in AC$[3] and $a \succ b$, then $f(b,a) \leftrightarrow_{AC} f(a,b) \rightarrow_{LPO} f(b,a)$, but of course $f(b,a) \not\rightarrow^+_{LPO} f(b,a)$. We shall design a variant of the lexicographic path ordering in which lexicographic rules are restricted to function symbols $f \notin AC$ and terms are compared only after they have been converted to a suitable normal form.

2 Transformation

Let AC be a set of associativity and commutativity rules. Henceforth, we shall assume that $\alpha(f) = \{2,3,4,\ldots\}$, if $f \in AC$, and $\alpha(f)$ is a singleton, otherwise. By F' we denote the set of all rewrite rules of the form

$$f(X, f(Y)) \rightarrow f(X,Y), \qquad f \in AC,\ |X| \geq 1,\ |Y| \geq 2;$$

by $\sim$ the (symmetric) rewrite relation generated by all rules

$$f(X,u,Y,v,Z) \ \leftrightarrow \ f(X,v,Y,u,Z), \qquad f \in AC;$$

and by F the rewrite system $F'/\!\sim$. The rewrite system F is length-decreasing and terminates; its rules are called *flattening rules*. We also say that s' is a *flattened version* of s if $s \rightarrow^!_F s'$. The relation $\sim$ is called the *permutation congruence*. Flattened versions of equivalent terms are unique up to permutation:

Lemma 1 *If $s' \leftarrow^!_F s \leftrightarrow^*_{AC} t \rightarrow^!_F t'$, then $s' \sim t'$.*

If $L = L'/\!\sim$ is a rewrite system, we define the relation $\succ_{L|_F}$ by: $u \succ_{L|_F} v$ if $u \rightarrow^!_F u' \rightarrow^+_{L/F} v' \leftarrow^!_F v$. In general, such "transformed relations" $\succ_{L|_F}$ are *not* rewrite relations.

For example, if $f \succ g$ and $f \in AC$, then $f(a,b) \rightarrow_{LPO} g(a,b)$, but $f(f(a,b),c) \rightarrow^!_F f(a,b,c) \not\rightarrow_{LPO} f(g(a,b),c)$. Observe that the term $f(f(a,b),c)$, but not its flattened version $f(a,b,c)$, can be rewritten by LPO. To address this problem we shall extend LPO to a rewrite system L with which flattened terms can be rewritten, if necessary.

3 Commutation

We say that a rewrite system S *commutes with* T if for all terms u, v, and u' with $u' \leftarrow_T u \rightarrow_S v$, there exists a term v', such that $u' \rightarrow^+_{S/T} v' \leftarrow^*_T v$.[4]

Proposition 1 *Let AC be a set of associativity-commutativity rules, F be the corresponding set of flattening rules, and $L = L'/\!\sim$ be a rewrite system. If L/F terminates and L commutes with F, then $\succ_{L|_F}$ is an AC-compatible reduction ordering.*

[3] We write $f \in AC$ if AC contains the respective rules for f.

[4] Similar commutation properties have been discussed by L. Bachmair and N. Dershowitz, *Proc. Eighth Int. Conf. on Automated Deduction*, Lect. Notes in Computer Science vol. 230, pp. 5–20, Berlin: Springer-Verlag.

Proof. It can easily be seen that $\succ_{L|_F}$ is AC-compatible and well-founded. Furthermore, we may use induction on $\to^+_{L\cup F}$ to prove that for all terms u, u', and v with $u' \leftarrow^!_F u \to^+_{L/F} v$, there exists a term v', such that $u' \to^+_{L/F} v' \leftarrow^*_F v$. (Note that $L \cup F$ is terminating, as both L/F and F are terminating.) ¿From this one can easily derive that $\succ_{L|_F}$ is a rewrite relation. □

For the purpose of extending LPO to a rewrite system L that commutes with F we analyze so-called "critical peaks" between L and F. For instance, a term $f(X, f(Y))$ can be rewritten in two different ways, which produces the peak

$$f(X, Y) \leftarrow_F f(X, f(Y)) \to_{LPO} f(X, y)$$

where $f \in AC$, $|X| \geq 1$, $|Y| \geq 2$, and $y \in Y$. We include the rule

$$f(X, Y) \;\to\; f(X) \qquad \text{if } f \in AC,\ |X| \geq 2, \text{ and } |Y| \geq 1$$

in L to ensure that $f(X, Y) \to_L f(X, y)$. Similarly, the peak

$$f(X, Y) \leftarrow_F f(X, f(Y)) \to_{LPO} f(X, g(Z)),$$

where $f \in AC$, $f \succ g$, $|X| \geq 1$, $|Y| \geq 2$, and $f(Y) \to_{LPO} t$, for all $t \in Z$, results in a rule

$$f(X, Y) \;\to\; f(X, g(Z)) \quad \text{if } f \in AC,\ f \succ g,\ |X| \geq 1,\ |Y| \geq 2, \text{ and } f(Y) \to_{LPO} t, \text{ for all } t \in Z.$$

We shall see that these two new rules already ensure commutation.

Let L be the rewrite system $L'/\sim$, where L' is the following recursively defined set of rules:

$$\begin{array}{rcll}
f(X) & \to & s & \text{if } s \in X \\
f(X) & \to & g(Y) & \text{if } f \succ g \text{ and } f(X) \to_{L'/\sim} t, \text{ for all } t \in Y \\
f(X, s, Y) & \to & f(X, t, Z) & \text{if } f \notin AC,\ s \to_{L'/\sim} t,\ |Y| = |Z|, \text{ and } f(X, s, Y) \to_{L'/\sim} u, \text{ for all } u \in Z \\
f(X, Y) & \to & f(X) & \text{if } f \in AC,\ |X| \geq 2, \text{ and } |Y| \geq 1 \\
f(X, Y) & \to & f(X, g(Z)) & \text{if } f \in AC,\ f \succ g,\ |X| \geq 1,\ |Y| \geq 2, \text{ and } f(Y) \to_{L'/\sim} u, \text{ for all } u \in Z
\end{array}$$

Rules of the form $f(X, Y) \to f(X)$ are called *AC-subterm rules*; rules $f(X, Y) \to f(X, g(Z))$, *AC-combination rules.*

Proposition 2 *The rewrite system L commutes with F.*

Proof. Let s, t, and s' be terms with $s' \leftarrow_F s \to_L t$. We use induction on $|s| + |t| + |s'|$ to prove that there exists a term t', such that $s' \to_L t' \leftarrow^*_F t$. There are several cases according to the rule applied in $s \to_L t$. We discuss one representative case: application of a combination rule. There are several subcases.

(a) Suppose $f(X') \leftarrow_F f(X) \to_L g(Y)$, where $f \succ g$ and $f(X) \to_L u$, for all $u \in Y$. Since $f(X') \leftarrow_F f(X) \to_L u$, for all $u \in Y$, we may apply the induction hypothesis to infer that for each $u \in Y$ there exists a term u' with $f(X') \to_L u' \leftarrow^*_F u$. Let Y' be the sequence obtained from Y

by replacing each term u by the corresponding term u'. Then $f(X') \to_L g(Y') \leftarrow_F^* g(Y)$, which proves this subcase.

(b) If the peak is of the form $f(X, Z) \leftarrow_F f(X, f(Z)) \to_L g(Y)$, where $f \succ g$ and $f(X, f(Z)) \to_L u$, for all $u \in Y$, then we may again use the induction hypothesis to infer that for each term $u \in Y$ there exists a term u', such that $f(X, Z) \to_L u' \leftarrow_F^* u$. Thus $f(X, Z) \to_L g(Y') \leftarrow_F^* g(Y)$, where Y' is obtained from Y by replacing each term u by u'.

(c) For each peak $f(X, Z) \leftarrow_F f(X, f(Z)) \to_L f(X, g(Y))$ we have $f(X, Z) \to_L f(X, g(Y))$ by virtue of an AC-combination rule.

The two remaining subcases–application of a combination rule either within the variable part of a flattening rule or at a disjoint subterm–are trivial. Applications of other rules are dealt with in a similar fashion. □

The following lemmas are useful for proving termination of L/F (or $L \cup F$).

Lemma 2 *If $g(X) \to_{L\cup F} t \to_{L\cup F}^n f(Y)$ and $g \not\succeq f$, then there exist a term $u \in X$ and a number k with $n \geq k$, such that $g(X) \to_L u \to_{L\cup F}^k f(Y)$.*[5]

Proof. Use induction on $(n, |t|)$, considering all possible cases of rewrites $g(X) \to_{L\cup F} t$. □

Lemma 3 *If $f \in AC$, $|Z| \geq 1$, and $f(X) \to_{L\cup F} t \to_{L\cup F}^n f(Y)$, then there exists a term u, such that $f(X, Z) \to_{L\cup F} u \to_{L\cup F}^* f(Y, Z)$ and $|f(t, Z)| \geq |u|$.*

Proof. Similar to the previous lemma, by induction on $(n, |t|)$. □

4 Termination

Suppose $L \cup F$ is not terminating. We define an infinite sequence of rewrites $t_0 \to_{L\cup F} t_1 \to_{L\cup F} t_2 \cdots$ as follows. Let t_0 be a shortest term from which there is an infinite sequence of rewrites. Once the term t_n has been determined, let $t_n \to_{L\cup F} t_{n+1}$ be any minimal rewrite such that there is an infinite sequence of rewrites from t_{n+1}. Here a rewrite $u \to_R v$ is considered to be smaller than a rewrite $u \to_S v'$ if either $|v'| > |v|$ or else $|v| = |v'|$, $R = L$, and $S = F$. By $\mathcal{S}$ we denote the set consisting of all proper subterms of terms t_i, $i \geq 0$, and terms derivable from them.

Lemma 4 *There is no infinite sequence of rewrites from any term in $\mathcal{S}$.*

Proof. Evidently, there is no infinite sequence of rewrites from any proper subterm of t_0. Suppose there is an infinite sequence of rewrites from some proper subterm t of t_{i+1}, but from no term that is (derivable from) a proper subterm of t_i. We consider several cases, depending on which rule is applied in the rewrite $t_i \to_{L\cup F} t_{i+1}$.

If $t_i \sim f(X, f(Y)) \to_F f(X, Y) \sim t_{i+1}$, then $t \sim f(X', Y')$, where $X' \subseteq X$, $Y' \subseteq Y$, and $|X'| \geq 1$. If $X' \neq X$, then

$$f(X', f(Y)) \to_L f(X', Y) \to_L f(X', Y')$$

and hence t is derivable from a proper subterm of t_i, which is a contradiction. Suppose $X' = X$. If $Y' \neq Y$ and $|Y'| \geq 2$, then $f(X, f(Y)) \to_L f(X, f(Y'))$ is a smaller rewrite than $f(X, f(Y)) \to_F$

[5] We write $s \to^n t$ to denote a sequence of n rewrites.

$f(X,Y)$ (because $|f(X,Y)| \geq |f(X,f(Y'))|$), which together with $f(X,f(Y')) \to_L f(X,Y')$ yields a contradiction. Finally, if $|Y'| = 1$, then t can be derived from t_i in one step (because $f(X,f(Y)) \to_L f(X,u)$, for all $u \in Y$), which is a contradiction.

Similar arguments can be applied in other cases. □

The lemma shows that the restriction $\succ_{\mathcal{S}}$ of the rewrite relation $\to^+_{L\cup F}$ to terms in $\mathcal{S}$ is well-founded. The lemma also indicates that no term t_{i+1} is a proper subterm of t_i. Thus, if t_i is written as $f_i(Y_i)$, then $f_i \succeq f_{i+1}$, for all $i \geq 0$. Since the precedence $\succ$ is well-founded, we may infer that $f_j = f_{j+1} = f_{j+2} = \cdots$, for some $j \geq 0$. Let s_i denote the term t_{j+i} and suppose s_i is of the form $f(X_i)$, for all $i \geq 0$.

The sequence $s_0 \to_{L\cup F} s_1 \to_{L\cup F} s_2 \to_{L\cup F} \cdots$ contains infinitely many rewrites where a rule in $L \cup F$ is applied at the top. If $f \notin AC$, then only lexicographic rules can be applied at the top. and the sequence $X_0, X_1, \ldots$ is lexicographically decreasing with respect to $\succ_{\mathcal{S}}$, which is a contradiction. If $f \in AC$, then only flattening and AC-combination rules can be applied at the top. (There have to be infinitely many applications of flattening, as AC-combination rules strictly decrease the number of top-level subterms.)

Now let us assign a status to the elements of X_k, for all $k \geq 0$. All terms in X_0 are *free*, and corresponding terms in X_k and X_{k+1} have the same status. Furthermore, if $X_{k+1} = (X_k \setminus \{f(Y)\}) \cup Y$, then all terms in Y are *free* (application of a flattening rule at the top introduces free terms); if $X_{k+1} = (X_k \setminus Y) \cup \{g(Z)\}$, where $|Y| \geq 2$, then $g(Z)$ is *bound* (application of an AC-combination rule at the top introduces a bound term); and if $X_{j+k+1} = (X_{j+k} \setminus \{u\}) \cup \{v\}$, then v has the same status as u. Observe that bound terms can only be introduced by AC-combination rules.

We claim that whenever there is a rewrite

$$s_k \sim f(X,f(Y)) \to_F f(X,Y) \sim s_{k+1},$$

then $f(Y)$ is free. For if $f(Y)$ is bound, it must have been derived from some term $g(Z')$ that had previously been introduced in a rewrite

$$s_{l-1} \sim f(X',Z) \to_L f(X',g(Z')) \sim s_l,$$

where $k > l$, $|Z| \geq 2$, $f \succ g$, $f(Z) \to_L u$, for all $u \in Z'$, and $f(X',Y) \to^*_{L\cup F} f(X,Y)$. Using Lemmas 2 and 3 we may infer that $f(Z) \to_L u \to^*_{L\cup F} f(Y)$, for some $u \in Z'$, and $f(X',Z) \to_{L\cup F} u' \to^*_{L\cup F} f(X',Y)$, for some term u' with $|f(X',g(Z'))| > |f(X',u)| \geq u'$, which contradicts the minimality of the rewrite $s_{l-1} \to_L s_l$.

Finally, for each term s_k we define a finitely-branching (labeled) tree T_k as follows. The tree T_0 consists of a root labeled with the symbol $\top$ and $|X_0|$ successor nodes labeled by the elements of X_0. The tree T_{k+1} is obtained from T_k as follows: if $X_{k+1} = (X_k \setminus \{f(Y)\}) \cup Y$, then new nodes labeled with the terms in Y are added as successors to the leaf $f(Y)$; if $X_{k+1} = (X_k \setminus Z) \cup \{g(Y)\}$, where $|Z| \geq 2$, then a single successor node labeled with the symbol $\bot$ is added to each leaf representing a term in Z; if $X_{k+1} = (X_k \setminus \{u\}) \cup \{v\}$, where u is free, then v is added as a successor to u. Note that each free term in X_k is represented by some leaf in the tree T_k. Furthermore, $T_k \neq T_{k+1}$ whenever the rewrite $s_k \to_{L\cup F} s_{k+1}$ is by application of a rule at the top. Thus, $T_\infty = \bigcup_i T_i$ is an infinite tree and, by König's Lemma, contains an infinite branch. On the other hand, if v is the label of a successor of a node labeled by u, then $u \succ_{\mathcal{S}} v$ (assuming $\top$ and $\bot$ represent maximum

and minimum elements, respectively). The existence of an infinite branch therefore contradicts the well-foundedness of $\succ_{\mathcal{S}}$.

In summary, we have proved:

Proposition 3 *The rewrite system $L \cup F$ is terminating.*

As a corollary to Propositions 1, 2, and 3 we obtain:

Theorem 1 *The relation $\succ_{L|_F}$ is an AC-compatible reduction ordering.*

It can easily be proved that the ordering also satisfies the *subterm property* (i.e., $t[s] \succ_{L|_F} s$, for all terms t and proper subterms s of t) and hence is a *simplification ordering*. We have thus also established that the lexicographic path ordering is a simplification ordering (this being the special case where $AC = \emptyset$.)

We conclude this section with an example. Let R be the rewrite system

$$\begin{array}{rcl} x + 0 & \rightarrow & x \\ x + y' & \rightarrow & (x + y)' \\ x * 0 & \rightarrow & 0 \\ x * y' & \rightarrow & x + (x * y) \\ (x + y) * z & \rightarrow & (x * z) + (y * z) \end{array}$$

and $\succ$ be a precedence in which $* \succ + \succ {}' \succ 0$. Then R is contained in $\succ_{L|_F}$ and hence R/AC is terminating.

5 Summary

We have illustrated by way of the lexicographic path ordering how to systematically modify a reduction ordering so as to obtain an ordering compatible with associativity and commutativity. The modification is guided by the requirement of establishing certain commutation and termination properties and employs standard techniques of term rewriting, such as an analysis of critical peaks. The specific ordering we have obtained essentially corresponds to an ordering introduced by Kapur, Sivakumar, and Zhang.[6] (The main difference is in the presentation: we describe an ordering via a schematic rewrite system, while Kapur et al. present an algorithm for computing the corresponding rewrite relation. The various operations used in their algorithm—"partitioning," "pseudo-copying," "elevating," etc.—correspond to sequences of rewrites by L/F.)

We believe that the above approach to designing reduction orderings can be applied in other contexts as well, e.g., to rewrite systems $R/AC1$ where $AC1$ is a set associativity, commutativity, and identity axioms. The *associative path ordering*[7] can also be formulated in this framework. This ordering differs from the above ordering in that commutation of the peak

$$f(X, Y) \leftarrow_F f(X, f(Y)) \rightarrow_{LPO} f(X, g(Z))$$

[6] *Proc. Conf. on Foundations of Software Technology and Theoretical Computer Science, 1990.*

[7] L. Bachmair and D. Plaisted, *J. Symbolic Computation* **1**(1985):329-349.

is achieved, not by introducing AC-combination rules, but by enriching F with "distributivity rules"

$$f(X, g(Z)) \to g(f(X, t_1), \ldots, f(X, t_n)),$$

where $f \succ g$ and $Z = t_1, \ldots, t_n$. Certain restrictions on the precedence ensure that the corresponding ordering $\succ_{APO}$ is indeed a reduction ordering. (The main difficulty consists in establishing a suitable commutation property, while termination is straightforward.) The ordering $\succ_{L|_F}$ imposes no such restrictions, but has the disadvantage that certain terms can not be compared, e.g., $f(a, c)$ and $f(b, b)$, where $f \in AC$ and $a \succ b \succ c$. The associative path ordering, on the other hand, allows for more flexible comparisons of arguments of associative-commutative function symbols, so that $f(a, c) \succ_{APO} f(b, b)$. Similar extensions of the rewrite system L that would allow for such comparisons result in non-terminating rewrite systems. It is an open question whether there exist any precedence-based AC-compatible reduction orderings so that two ground terms are either equivalent with respect to AC or else are comparable.[8]

[8]There are orderings with these properties, but they are not precedence-based, see P. Narendran and M. Rusinowitch, *Proc. Fourth Int. Conf. on Rewrite Techniques and Applications*, Lect. Notes in Comp. Scie. vol. 480, pp. 423–434, Berlin: Springer-Verlag.