Technical Report

UCAM-CL-TR-170
ISSN 1476-2986

Number 170

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Ordered rewriting and confluence

Ursula Martin, Tobias Nipkow

May 1989

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Ordered Rewriting and Confluence

Ursula Martin* and Tobias Nipkow†

Department of Computer Science, RHBNC, University of London
Egham, Surrey TW20 0EX, *UK*
and
University of Cambridge, Computer Laboratory,
Pembroke Street, Cambridge CB2 3QG, *UK*

Abstract

One of the major problems in term rewriting theory is what to do with an equation which cannot be ordered into a rule. Many solutions have been proposed, including the use of special unification algorithms or of unfailing completion procedures.

If an equation cannot be ordered we can still use any instances of it which can be ordered for rewriting. Thus for example $x * y = y * x$ cannot be ordered, but if a, b are constants with $b * a > a * b$ we may rewrite $b * a \longrightarrow a * b$. This idea is used in unfailing completion, and also appears in the Boyer-Moore system. In this paper we define and investigate completeness with respect to this notion of rewriting and show that many familiar systems are complete rewriting systems in this sense. This allows us to decide equality without the use of special unification algorithms. We prove completeness by proving termination and local confluence. We describe a confluence test based on recursive properties of the ordering.

*The author acknowledges support of the UK SERC under grant GR/E 83634.

†This research was supported in part by NYNEX, NSF grant CCR-8706652, and by the Advanced Research Projects Agency of the DoD, monitored by the ONR under contract N00014-83-K-0125.

1 Introduction

One of the major problems in term rewriting theory is what to do with an equation which cannot be ordered into a rule. Many solutions have been proposed, including the use of special unification algorithms [7] or of unfailing completion procedures [1,6].

If an equation cannot be ordered we can still use any instances of it which can be ordered for rewriting. Thus for example

$$x * y = y * x$$

cannot be ordered, but if a, b are constants with $b * a > a * b$ we may rewrite

$$b * a \longrightarrow a * b.$$

This idea is used in unfailing completion, and also appears in Boyer-Moore [2]. In this paper we define and investigate completeness with respect to this notion of rewriting and show that many familiar systems are complete rewriting systems in this sense. This allows us to decide equality without the use of special unification algorithms. We prove completeness by proving termination and local confluence. We describe a confluence test based on recursive properties of the ordering.

1.1 Summary

In this section we summarize our results. Precise definitions are given below. An *ordered rewriting system* consists of a set of equations E and a monotonic ordering on terms $>$ which is total on ground terms. We say a term s rewrites to a term t, denoted by $s \longrightarrow t$, if there is an equation $r = l$ or $l = r$ in E, a substitution σ and a subterm σl of s such that $\sigma l > \sigma r$ and t is s with that subterm replaced by σr. Thus for example if $x * y = y * x$ is in E and $a * b > b * a$ then $a * b \longrightarrow b * a$. We observe that the usual notion of a rewriting system can be regarded as a special case of our concepts in the case when the ordering allows all the equations to be ordered into rules.

A *ground complete ordered rewriting system* is one which is terminating and confluent on ground terms. This means that any ground term can be rewritten to a unique canonical form, and we can decide equality between ground terms, and hence between variable terms by regarding the variables as generalised constants. This process uses only unification in the empty theory. In section 4 we give examples of ground complete ordered rewriting systems including AC, ACI, Boolean rings, Distributivity and Abelian Groups.

Example 1 As an example let E be

$$(x * y) * z = x * (y * z) \quad (1)$$
$$x * y = y * x \quad (2)$$
$$x * (y * z) = y * (x * z) \quad (3)$$

and let $>$ be any monotonic ordering on terms which is total on ground terms and satisfies for all ground terms x, y, z

$$(x * y) * z > x * (y * z) \quad (4)$$

$$x * y > y * x \quad \text{if } x > y \tag{5}$$
$$x * (y * z) > y * (x * z) \quad \text{if } x > y. \tag{6}$$

Then $(E, >)$ is an ground complete ordered rewriting system. For example suppose that $>$ is the lexicographic path ordering (see section 3) and a, b, c are constants with $c > b > a$. Then

$$b*(c*(b*a)) \longrightarrow b*(c*(a*b)) \longrightarrow b*(a*(c*b)) \longrightarrow a*(b*(c*b)) \longrightarrow a*(b*(b*c)).$$

To prove completeness we need as usual to prove termination and confluence. Termination is generally proved by showing that the ordering is well-founded[1], and the orderings we need to do this for our examples are discussed in section 3. To prove confluence we need to prove local confluence, and in section 2.1 we prove the necessary version of the critical pairs lemma

The usual notion of rewriting and confluence allows confluence to be checked automatically by computation of normal forms. At first sight this is not so for ordered rewriting, which appears to require infinitely many calculations. However we shall explain in section 2.2 how the confluence test may indeed be automated in many cases by axiomatising the properties of the orderings and rewritings that we need. This automation works for most of the examples of section 4.

Consider the example above. Computing critical pairs between

$$\begin{aligned} (x * y) * z &\longrightarrow x * (y * z) \\ x * y &= y * x \end{aligned}$$

we obtain

$$z * (x * y) \longleftarrow (x * y) * z \longrightarrow x * (y * z),$$

so we have to prove that $z * (x * y)$ and $x * (y * z)$ are joinable for all ground terms x, y, z. Now since x, y, z are ground terms we may consider the possible relationships between them under $>$. For example if $x > z > y$ then

$$x * (y * z) \longrightarrow y * (x * z) \longrightarrow y * (z * x) \longleftarrow z * (y * x) \longleftarrow z * (x * y)$$

While our technique allows the computation of canonical forms without a special matching algorithm we note that it is not always as powerful as rewriting with an equational matching algorithm.

Example 2 Consider the example above together with the equation $f(x * x) = 1$. Rewriting with an AC matching algorithm shows that $f(a * (a * (b * b))) = 1$. However, there is no equivalent ground complete ordered rewriting system as any such system would have to contain infinitely many equations to deal with

$$f(a * a), \quad f(a * (a * (b * b))), \quad f(a * (a * (a * (a * (b * b)))))$$

and so on.

[1] In fact, it is sufficient to show well-foundedness within equivalence classes. If all equivalence classes are finite, this follows because any ordering an a finite set is well-founded.

On the other hand the advantages of our method are that

- Ground rewriting is possible without E-matching algorithms.
- Ground rewriting is sufficient for theorem proving.
- Completion is possible without E-completion.

It shares these features with unfailing completion as described for example in [6]. Indeed, it is very similar to unfailing completion in two technical aspects: both sides on an equation may give rise to critical pairs, and complete systems need only be confluent for ground terms. In contrast to [6], our method is specially designed to test for confluence of ground terms. Thus we obtain many complete systems which their approach fails to recognize as complete.

2 Critical Pairs, Confluence, and Completion

We assume that all concepts and definitions are as in [5] or [3]. Let Σ be a set of function symbols and V a set of variables. The set of all terms in $\Sigma \cup V$ is denoted by $\mathcal{T} = \mathcal{T}(\Sigma \cup V)$, and the set of all ground terms is the subalgebra $\mathcal{T}_G = \mathcal{T}(\Sigma)$. The function $\mathcal{V}$ returns the set of variables in a term. A term can be represented as subterms in a *context* by writing $C[s_1, \ldots, s_n]$. The context C is a λ-term $\lambda x_1, \ldots, x_n.t$, and $C[s_1, \ldots, s_n]$ denotes application, i.e. the simultaneous the replacement of all x_i by s_i. In particular we assume that every x_i occurs exactly once in t.

An ordering $>$ on a set S is a relation which is irreflexive and transitive, so that it is false that $x > x$, and if $x > y$ and $y > z$ then $x > z$. An ordering on $\mathcal{T}$ is monotonic if for all function symbols f and terms $s_1, \ldots, s_n, s, t$ we have $f(s_1, \ldots, s, \ldots s_n) > f(s_1, \ldots, t, \ldots s_n)$ if $s > t$.

An *ordered rewriting system* is a pair $(E, >)$ where E is a set of equations in $\mathcal{T}$ and $>$ is a monotonic ordering on $\mathcal{T}$ which is total on ground terms. The notation $s \doteq t \in E$ is short for $s=t \in E \vee t=s \in E$. If for some $l \doteq r \in E$ we have $\sigma l > \sigma r$ for all substitutions σ we write $l \longrightarrow r$ and call it a rule.

The ordered rewriting system $(E, >)$ induces a relation $\longrightarrow$ defined as

$$C[\sigma s] \longrightarrow C[\sigma t] \quad \text{if } s \doteq t \in E \ \wedge \ \sigma s > \sigma t$$

Since $>$ is monotonic, $p \longrightarrow q$ implies $p > q$. The restriction of $\longrightarrow$ to ground terms is denoted by $\Longrightarrow$. We use $\longrightarrow^*$ to denote the reflexive transitive closure of $\longrightarrow$. Two terms s and t are called *joinable*, written $s \downarrow t$, iff there is a term u such that $s \longrightarrow^* u$ and $t \longrightarrow^* u$. They are called *ground joinable*, written $s \Downarrow t$, iff any two ground instances σs and σt are joinable. An ordered rewriting system is called *ground terminating* if there is no sequence of ground terms $\{a_i | i \in \mathbf{N}\}$ such that $a_i \Longrightarrow a_{i+1}$ for all i. An ordered rewriting system is called *ground confluent* if whenever r, s, t are ground terms with $r \Longrightarrow^* s$ and $r \Longrightarrow^* t$ then $s \downarrow t$. If the terminating or confluence conditions hold for all terms rather than just ground terms we call $(E, >)$ terminating or confluent respectively. An ordered rewriting system which is terminating and confluent is called *complete*; one which is ground terminating and ground confluent is called

ground complete. It follows from Newman's lemma that if $(E,>)$ is complete then each term s has a unique normal form, and if $(E,>)$ is ground complete this is true for ground terms. Thus if s,t are terms (ground terms) and $(E,>)$ is complete (ground complete) then $s =_E t$ if and only if their normal forms are identical. If s and t are arbitrary terms we may still use a ground complete system to decide equality if we regard the variables occurring in s and t as new constants.

In the sequel let $(E,>)$ denote an ordered rewriting system and let $\longrightarrow$, $\Longrightarrow$ etc. be the rewrite relations it generates. If there is a second ordering, say $\succ$, we write $\longrightarrow_\succ$, $\Downarrow_\succ$ etc. to denote the relations induced by $(E,\succ)$.

2.1 The Critical Pair Lemma

This section deals with the extension of the critical pair lemma to ordered rewriting.

Definition 1 Given two equations $C[u] = t$ and $v = w$, where $u \notin V$, and a most general unifier σ of u and v such that $\sigma(C[u]) \not\leq \sigma t$ and $\sigma v \not\leq \sigma w$, then $(\sigma t, \sigma(C[w]))$ is a *critical pair.*

The set of all critical pairs of E is the set of all critical pairs between any two equations $p \doteq q, s \doteq t \in E$.

Note that because of the symmetry of $\doteq$ both sides of an equation can give rise to critical pairs. If $>$ orders every equation in E into a rule our definition of critical pairs reduces to the usual one.

Lemma 1 *If $>$ is total within equivalence classes of ground terms, both sides of an equation $s \doteq t \in E$ are ground joinable.*

Proof For any ground substitution σ we have $\sigma s > \sigma t$, $\sigma s = \sigma t$ or $\sigma s < \sigma t$, which implies $\sigma s \longrightarrow \sigma t$, $\sigma s = \sigma t$ or $\sigma s \longleftarrow \sigma t$ and hence $\sigma s \downarrow \sigma t$. □

We now come to the proof of the critical pair lemma. We cannot directly appeal to the theorems in [4] because we deal with ordered rewriting. In particular this means that we may have $\sigma l \longrightarrow \sigma r$ but not $\tau l \longrightarrow \tau r$ for two substitutions σ and τ. Fortunately, $\longrightarrow$ is still *compatible*: if $s \longrightarrow t$ then $C[s] \longrightarrow C[t]$ for any context C.

Lemma 2 *An ordered rewriting system $(E,>)$ is locally ground confluent iff all critical pairs are ground joinable.*

Proof The proof is very similar to the one in [4], except that we need to take $>$ into account when rewriting. The $\Rightarrow$-direction of the proposition is trivial. For the other direction let all critical pairs be ground joinable, and let $r \Longrightarrow s$ and $r \Longrightarrow t$. Thus there are equations $p \doteq q, u \doteq v \in E$, matching substitutions σ and τ, and contexts M and N such that $\sigma p \Longrightarrow \sigma q$, $\tau u \Longrightarrow \tau v$, and $r = M[\sigma p] = N[\tau u]$, $s = M[\sigma q]$, and $t = N[\tau v]$. We distinguish 3 cases.

Case 1: the two rewrites are at disjoint occurrences. Then $r = C[\sigma p, \tau u]$, $s = C[\sigma q, \tau u]$, $t = C[\sigma p, \tau v]$, and therefore $s \Longrightarrow C[\sigma q, \tau v] \Longleftarrow t$.

Case 2: the two rewrites overlap each other. W.l.o.g. let $r = C[\sigma p]$ where τu is a subterm of σp. Then $s = C[\sigma q]$.

Case 2a: there is a variable x in p such that $\sigma x = D[\tau u]$. Let $p = A[x^m]$ and $q = B[x^n]$, i.e. p and q contain m and n distinct occurrences of x respectively. Then $\sigma p = A'[\sigma(x)^m] = A'[D[\tau u]^m]$ and $\sigma q = B'[\sigma(x)^n] = B'[D[\tau u]^n]$. By compatibility $\sigma x \Longrightarrow D[\tau v]$ and thus $s \Longrightarrow^* C[B'[D[\tau v]^n]] =: s'$ and $t = C[A'[\sigma x, \ldots, \sigma x, D[\tau v], \sigma x, \ldots, \sigma x]] \Longrightarrow^* C[A'[D[\tau v]^m]] =: t'$. Thus $s' = C[\sigma' q]$ and $t' = C[\sigma' p]$ for $\sigma' = \sigma + [x \to D[\tau v]]$. Because $>$ is total within equivalence classes of ground terms, lemma 1 implies that $\sigma' p \downarrow \sigma' q$. By compatibility $s \downarrow t$ holds as well.

Case 2b: otherwise σp must be the instance of a proper overlap of p and u. Therefore $(\sigma p, t_1)$, where $t = C[t_1]$, is a ground instance of a critical pair between $p = q$ and $u = v$. Thus $\sigma p \downarrow t_1$ which implies $s \downarrow t$ by compatibility. □

The proof of this lemma relies on the totality of $>$ within equivalence classes of ground terms. The following example shows that this requirement cannot be dropped:

Example 3 Let $\Sigma = \{*\} \cup C$, where C is a set of constants, $E = \{x * y = y * x\}$, and $s > t$ iff the leftmost constant in s is $>$ the leftmost constant in t. Clearly $>$ is not total because a and $a * b$ are incomparable. Assume that $a < b$. Then the term $r = (b * a) * a$ can be rewritten to $s = a * (b * a)$ and $t = (a * b) * a$. However, s and t are not joinable because s rewrites only to $a * (a * b)$, which is in normal form, and t is in normal form already.

On the other hand there is only a single critical pair $(y * x, y * x)$ in E which is trivially ground joinable. This shows that for non-total $>$ the consideration of critical pairs does not suffice to determine local ground confluence.

Corollary 1 *A terminating ordered rewriting system is ground confluent iff all critical pairs are ground joinable.*

Proof If all critical pairs are ground joinable we know by lemma 2 that $\Longrightarrow$ is locally ground confluent. Termination implies confluence of $\Longrightarrow$. The other direction is trivial. □

2.2 Automating It

In contrast to ordinary rewriting systems, where critical pairs are required to be joinable, we need the weaker criterion of ground joinability. It is not at all clear how a test of the latter property can be automated since it talks about an infinite set of ground instances. In fact we believe that ground joinability is in general undecidable. The purpose of this section is to give some sufficient criteria which are easily implementable and powerful enough to solve some non-obvious examples. On the other hand they are far from complete. Section 4.9 contains an example which is easily proved to be ground joinable but which is not covered by our method.

The principle idea underlying the automation has already been sketched in the introduction: given two terms s and t, we consider all possible relationships between the variables in s and t under $>$ and $=$ and try to join s and t for each of them. Since there are only finitely many relationships, namely all linear orderings, we only have to consider a finite, albeit possibly very large, number of cases. It remains to be explained how rewriting of terms with variables is to proceed if we do not know what the variables

stand for, only how they are related to each other with respect to $>$. As an example take the term $y * x$ with the constraint $x < y$. It requires some intimate knowledge of $>$ to determine whether this implies that $y * x > x * y$, i.e. whether commutativity is applicable.

Instead of working with a particular ordering and inferring some of its properties, we assume a small set of properties of the ordering which allow us to order enough terms for proving ground confluence. For the AC case we have seen in the introduction that the implications (4)-(6) are sufficient for joining one of the critical pairs under a particular set of constraints. In section 4.2 we show that the equations (1)-(3) together with any ordering satisfying (4)-(6) are ground confluent.

The advantage of this approach is its generality: ground confluence is proved for any ordering satisfying the properties we have assumed. However, it means that one has to be careful in the choice of properties. For example they must not violate well-foundedness.

We will now describe a test for ground joinability based on the above ideas. Formally, the "properties" of the ordering are given as a closure operator C on $\mathcal{T} \times \mathcal{T}$ subject to the restriction

$$(s,t) \in C(>) \quad \Rightarrow \quad (\sigma s, \sigma t) \in C(\sigma(>)) \tag{7}$$

where $\sigma(>) = \{(\sigma u, \sigma v) \mid u > v\}$. The intuition is that C takes a relation on terms and returns the set of consequences implied by the properties we assumed of the ordering. The above restriction ensures that C is well behaved with respect to substitutions. This enforces for example that if $x * y > y * x$ follows from $x > y$, then $x' > y'$ must imply $x' * y' > y' * x'$. We say that an ordering $>$ is *compatible* with C if $C(>) = >$. As a consequence of restriction (7) we obtain:

Lemma 3 *Let E be a set of equations, let $>$ and $\succ$ be two relations on $\mathcal{T}$, and let σ be a substitution such that $\sigma(\succ) \subseteq >$. Then $u \longrightarrow_{C(\succ)} v$ implies $\sigma u \longrightarrow_{C(>)} \sigma v$ for all terms u, v.*

Proof From $u \longrightarrow_{C(\succ)} v$ it follows that $u = C[\tau l]$, $v = C[\tau r]$ and $(\tau l, \tau r) \in C(\succ)$ for some $l \doteq r \in E$. From $(\tau l, \tau r) \in C(\succ)$ it follows by (7) that $(\sigma\tau l, \sigma\tau r) \in C(\sigma(\succ))$. Since C is a closure operator and $\sigma(\succ) \subseteq >$ we also have $(\sigma\tau l, \sigma\tau r) \in C(>)$. Thus $\sigma u = \sigma(C)[\sigma\tau l] \longrightarrow_{C(>)} \sigma(C)[\sigma\tau r] = \sigma v$. □

Ordering the variables in a term with respect to $=$ and $>$ is equivalent to providing a total order on equivalence classes of variables. If ρ is an equivalence on a set of variables, $\hat{\rho}$ denotes a substitution which maps each variable to some fixed representative of its equivalence class. Testing for ground joinability of two terms s and t by considering all total orders on equivalence classes of variables in s and t leads to the following definition. If $\hat{\rho}s \downarrow_{C(\succ)} \hat{\rho}t$ holds for all equivalences ρ on the variables in s and t and all total orders $\succ$ on the range of $\hat{\rho}$, then we write

$$s \Downarrow_C t.$$

Restriction (7) ensures that this definition is independent of the particular choice of representatives of ρ-equivalence classes.

The next lemma shows that $s \Downarrow_C t$ does imply ground joinability:

Lemma 4 *If $s \Downarrow_C t$ then $s \Downarrow_> t$ holds for all orderings $>$ compatible with C.*

Proof Let $>$ be compatible with C and let σ be some ground substitution with $dom(\sigma) = \mathcal{V}(s) \cup \mathcal{V}(t)$. We have to show that $\sigma s \downarrow_> \sigma t$.

Let $\rho = ker(\sigma)$ and define $x \succ y$ iff $\sigma x > \sigma y$ for x, y in the range of $\hat{\rho}$. Then $\succ$ is a total order and $s \Downarrow_C t$ implies $\hat{\rho}s \downarrow_{C(\succ)} \hat{\rho}t$. Since $>$, $\succ$, and σ satisfy the assumptions of lemma 3 it follows that $\sigma s = \sigma\hat{\rho}s \downarrow_{C(>)} \sigma\hat{\rho}t = \sigma t$. Since $>$ is compatible with C we have $\sigma s \downarrow_> \sigma t$. □

From this lemma and the definition of $\Downarrow_C$ it follows directly that

Corollary 2 *If $C(\succ)$ is recursive and well-founded for all recursive and well-founded $\succ$, then $\Downarrow_C$ is a sufficient recursive criterion for ground joinability with respect to all orderings compatible with C.*

This is the first step towards automating the test for ground joinability. The second ingredient is lemma 1. Combining all these criteria we obtain the following set of rules:

$$\begin{aligned} s \Downarrow t &\Leftarrow s = t \\ s \Downarrow t &\Leftarrow s \Downarrow_C t \\ s \Downarrow t &\Leftarrow \exists l \doteq r \in E, \sigma.\ \sigma l = s \ \wedge\ \sigma r = t \\ f(s_1, \ldots, s_n) \Downarrow f(t_1, \ldots, t_n) &\Leftarrow \forall i.\ s_i \Downarrow t_i \end{aligned}$$

The first clause is obvious, the second and third ones are consequences of lemmas 4 and 1 respectively, and the last one follows from compatibility of rewriting.

The prototype implementation of this test is written in Prolog and follows exactly the above four Horn clauses. C is just another predicate. In all our examples C consists of the implications (4)-(6) and further clauses specific to the example.

2.3 Completion

The critical pair lemma in the preceding section leads to a completion algorithm in the usual way: critical pairs which are not ground joinable are added as new equations. Formally this can be expressed as an inference rule between sets of equations:

$$\frac{E}{E \cup \{s = t\}} \quad \text{if } (s,t) \text{ is a critical pair of } E \text{ and not } s \Downarrow t.$$

If this process terminates because all critical pairs are ground joinable, we have obtained a ground complete ordered rewriting system. In addition one may want to obtain a reduced rewriting system by simplifying the right or left hand sides of equations by other equations. This can be achieved by the following rule:

$$\frac{E \cup \{s \doteq t\}}{E \cup \{s = u\}} \quad \text{if } t \longrightarrow_E u \ \wedge\ s \doteq t \notin E$$

Since we are only interested in ground confluence, ground joinable equations can be removed:

$$\frac{E \cup \{s \doteq t\}}{E} \quad \text{if } s \Downarrow t$$

The applications of these three rules may be interleaved arbitrarily.

A prototype implementation of this completion procedure has been written in Prolog and was used for all the examples in section 4.

3 Orderings

Our notations and concepts are taken from Dershowitz [3].

An ordering is called well-founded if there is no set $\{a_i | i \in \mathbf{N}\}$ with $a_i > a_{i+1}$ for each i. We have

Lemma 5 *Let $(E, >)$ be an ordered rewriting system. If $>$ is well-founded then $(E, >)$ is terminating.*

Proof The monotonicity condition ensures that if $s \longrightarrow t$ then $s > t$, so if $>$ is terminating there can be no infinite chain of rewrites. □

The following orderings will be used in the sequel.

Lexicographic Path Ordering

Let $s = f(s_1, \ldots, s_m), t = g(t_1, \ldots, t_n)$. Let $>$ be an ordering on function symbols. Then

$s > t$ if and only if

- $s_i \geq t$ for some $i = 1, \ldots, m$, or
- $f > g$ and $s > t_j$ for all $j = 1, \ldots, n$, or
- $f = g$ (so $n = m$) and $(s_1, \ldots, s_n)$ is greater than $(t_1, \ldots, t_n)$ in the lexicographic ordering from the left on sequences induced by $>$, and $s > t_i$ for $i = 2, \ldots, n$.

Then we have

Lemma 6

1. *The lexicographic path ordering is well-founded, and is total on ground terms if the operator precedence is total.*
2. *If f, g are binary function symbols with $f > g$ and x, y, u, v are any terms and $f(x, y) > u, f(x, y) > v$ then $f(x, y) > g(u, v)$.*

Proof

1. This is just Theorem 22 of [3]
2. Follows from the definitions.

□

Knuth-Bendix Orderings

The essence of the Knuth-Bendix orderings is to compare terms first by weight and then lexicographically by an operator precedence. For details see [8] or [10], where proofs will be found of

Lemma 7 *The Knuth-Bendix ordering is monotonic and well-founded, and is total on ground terms if the operator precedence is total.*

Lexicographic Orderings

Let $\Sigma = \{f, a_1, \ldots, a_k\}$ where f is binary and $a_1, \ldots a_n$ are constants. Assume $a_1 < a_2 \ldots < a_k$. Define $>_t$ for $t = 1, 2$ by

$$\begin{array}{rcll} a_i & >_t & a_j & \text{if and only if} \quad i \geq j \\ f(x,y) & >_t & f(z,u) & \text{if and only if} \quad x > z \text{ or } x = z \text{ and } y > u \\ f(a_i,x) & >_1 & a_j & \text{if and only if} \quad i \geq j \\ a_j & >_1 & f(a_i,x) & \text{if and only if} \quad j > i \\ f(a_i,x) & >_2 & a_j & \text{for all } i,j = 1,\ldots,k \end{array}$$

where x, y, z, u are arbitrary ground terms. Then

Lemma 8 *For each of the orderings $>_1, >_2$*

1. *$>_t$ is a monotonic ordering and total on ground terms*
2. *$f(f(x,y),z) >_t f(x,f(y,z))$ and if $x >_t y$ then $f(x,y) > f(y,x)$ for all ground terms x, y, z.*
3. *$>_t$ is not well-founded.*

Proof The proof is straightforward. For (3) notice that we have

$$f(a_2,a_1) >_t f(a_1,f(a_2,a_1)) >_t f(a_1,f(a_1,f(a_2,a_1))) >_t \ldots.$$

□

Notice that $>_1$ is described by Boyer and Moore [2], where it is expressed in terms of projecting onto strings by

$$s(a_i) = a_i, s(f(x,y)) = fs(x)s(y)$$

where f denotes function application, and ordering the strings lexicographically.

To enable us to use the automatic confluence test described in section 2.2 we need to identify orderings with certain properties.

Definition 2 An ordering is called *AC compatible* for the binary operator f if it is monotonic, well-founded and total on ground terms, and satisfies for all ground terms x, y, z

$$\begin{array}{rcll} f(f(x,y),z) & > & f(x,f(y,z)) & \\ f(x,y) & > & f(y,x) & \text{if } x > y \\ f(x,f(y,z)) & > & f(y,f(x,z)) & \text{if } x > y \end{array}$$

Lemma 9 *Let $>$ be the Knuth-Bendix ordering or the lexicographic path ordering and f any binary function symbol. Then $>$ is AC-compatible for f.*

Proof Follows from the definitions. □

4 Examples

We present here examples of ground complete ordered rewriting systems. 4.1-4.9 are standard algebraic systems. In 4.10 we investigate combination of rewriting systems. In 4.11 we investigate an alternative ordering for which AC has a ground complete system containing two rules only, and in we give a ground complete ordered rewriting system for abelian groups.

Examples 4.2-4.9 are all ground complete for any ordering $>$ which

1. is AC-compatible for all AC operators in the system, and
2. satisfies $s > t$ for all rules $s \longrightarrow t$.

For examples 4.1-4.6 the Knuth-Bendix orderings and the lexicographic path orderings have the required properties. These examples were all proved ground complete using the method of section 2.2. The closure operator C was induced by 1 and 2 above.

Intuitively one reason why all the examples involving AC work is that we we are doing is using a sorting algorithm. Any ground term is equal to a product of irreducibles and the AC rules (1)-(3) are sorting these irreducibles into increasing order using bubble sort. The two rule version is merely using a different sorting algorithm.

4.1 Commutativity

Let E be $\{x * y = y * x\}$ and $>$ any monotonic ordering total within equivalence classes of ground terms. Then $(E, >)$ is a ground complete ordered rewriting system. It is confluent because there are no (non-trivial) critical pairs. It is terminating since each equivalence class is finite, and so any infinite chain of rewrites would contain a loop, which would imply that $>$ was not irreflexive.

4.2 Associativity and Commutativity

This example has been discussed in the introduction. Let E be

$$\begin{array}{rcl} (x * y) * z & \longrightarrow & x * (y * z) \\ x * y & = & y * x \\ x * (y * z) & = & y * (x * z) \end{array}$$

$(E, >)$ is also ground complete if $>$ is either of the lexicographic orderings $>_i$.

4.3 Associativity and Commutativity — Another Version

In the introduction we observed that one of the critical pairs generated by 4.2 was $(z * (x * y), x * (y * z))$. We may use this to obtain another three rule ground complete ordered rewriting system for AC. Let E be

$$\begin{array}{rcl} (x * y) * z & \longrightarrow & x * (y * z) \\ x * y & = & y * x \\ z * (x * y) & = & x * (y * z) \end{array}$$

and $>$ any ordering which satisfies (5) and $z * (x * y) > x * (y * z)$ if $z > x$. The lexicographic path ordering and the Knuth-Bendix ordering have this property.

4.4 Associativity, Commutativity, and Idempotence

Let E be

$$\begin{aligned}(x * y) * z &\longrightarrow x * (y * z)\\ x * y &= y * x\\ x * (y * z) &= y * (x * z)\\ x * x &\longrightarrow x\\ x * (x * y) &\longrightarrow x * y.\end{aligned}$$

4.5 Groups of Exponent Two

Let E be

$$\begin{aligned}(x * y) * z &\longrightarrow x * (y * z)\\ x * y &= y * x\\ x * (y * z) &= y * (x * z)\\ x * x &\longrightarrow 1\\ x * (x * y) &\longrightarrow y\\ x * 1 &\longrightarrow x\\ 1 * x &\longrightarrow x\end{aligned}$$

Then $(E, >)$ is a ground complete ordered rewriting system for groups of exponent two.

4.6 Groups of Exponent Two in Disguise

We want to prove that the two laws

$$\begin{aligned}(x * x) * y &= y\\ (x * y) * z &= (y * z) * x \qquad (8)\end{aligned}$$

axiomatize groups of exponent two. Starting from this system, the completion procedure generated the following list of critical pairs, ordering some of them into rules:

$$\begin{aligned}(x * y) * x &\longrightarrow y\\ (x * y) * y &\longrightarrow x\\ x * x &= y * y \qquad (9)\\ x * x &\longrightarrow 1 \qquad (10)\\ 1 * x &\longrightarrow x\\ x * 1 &\longrightarrow x\\ x * (x * y) &\longrightarrow y\end{aligned}$$

$$\begin{aligned} x*y &= y*x \\ (x*y)*z &\longrightarrow x*(y*z) \\ x*(y*z) &= y*(x*z) \end{aligned}$$

Notice that (10) is the result of "dividing" (9), i.e. introducing the new constant 1. The final set of equations (all the ones below and including (10)) is the same as in section 4.5. All the other equations are now joinable.

In [9] the same problem is attacked with the help of the term rewriting system Reve. Because (8) cannot be oriented into a rule, Reve cannot deal with it directly. Martin obtained the result by working with consequences of (8) that can be ordered.

4.7 Distributivity

Let E be

$$\begin{aligned} (x*y)*z &\longrightarrow x*(y*z) \\ x*y &= y*x \\ x*(y*z) &= y*(x*z) \\ x*(y+z) &\longrightarrow x*y+x*z \\ (x+y)*z &\longrightarrow x*z+y*z \end{aligned}$$

and let $>$ be any ordering which is AC-compatible for both $+$ and $*$. For example the lexicographic path ordering fits the bill. Then $(E,>)$ is a ground complete ordered rewriting system.

4.8 Boolean Rings

The following is a ground complete set of ordered rewrite rules for Boolean rings.

$$\begin{aligned} x+y &= y+x & \qquad x*y &= y*x \\ x+(y+z) &= y+(x+z) & \qquad x*(y*z) &= y*(x*z) \\ x+x &\longrightarrow 0 & \qquad x+0 &\longrightarrow x \\ 0+x &\longrightarrow x & \qquad x*x &\longrightarrow x \\ 1*x &\longrightarrow x & \qquad x*1 &\longrightarrow x \\ x*0 &\longrightarrow 0 & \qquad 0*x &\longrightarrow 0 \\ (x*y)*z &\longrightarrow x*(y*z) & \qquad (x+y)+z &\longrightarrow x+(y+z) \\ x*(y+z) &\longrightarrow x*y+x*z & \qquad (x+y)*z &\longrightarrow x*z+y*z \\ x*(x*y) &\longrightarrow x*y & \qquad x+(x+y) &\longrightarrow y \end{aligned}$$

The ordering must be AC-compatible for both $+$ and $*$. The lexicographic path ordering has these properties. Ground confluence can be checked by the technique of of section 2.2.

4.9 Another System

The equation

$$(x*x)*y = y*(x*x) \tag{11}$$

is an example of a system that is ground confluent for any ordering total and well-founded on ground terms. The reason is that the only nontrivial critical pair

$$y * ((x * x) * (x * x)) = ((x * x) * (x * x)) * y$$

is an instance of (11). By lemma 1 this implies ground joinability.

If (11) is generalized slightly to

$$(x * y) * z = z * (x * y) \tag{12}$$

and we assume that $x * y > z$ implies $(x * y) * z > z * (x * y)$, the criteria of section 2.2 fail to prove ground confluence, although there is a very simple proof. The two critical pairs are

$$\begin{aligned} (z * (x * y)) * u &= u * ((x * y) * z) \\ ((x * y) * z) * u &= u * (z * (x * y)). \end{aligned} \tag{13}$$

Let us just consider the first one. If $(x * y) = z$, (13) is an instance of (12). If $(x * y) > z$ or $(x * y) < z$, (13) can be rewritten to $(z * (x * y)) * u = u * (z * (x * y))$ or $((x * y) * z) * u = u * ((x * y) * z)$, both of which are instances of (12). Again lemma 1 implies ground joinability. The proof for the second critical pair is practically identical.

The tests in section 2.2 cannot cope with these critical pairs because the proof of ground joinability is based on a case distinction which compares not just variables but whole subterms, namely $x * y$ and z.

4.10 Combination of Systems

In this section we discuss how a ground complete ordered rewriting system may be combined with a ground complete rewriting system in the usual sense.

Lemma 10 *Suppose that*

1. *R is a ground complete rewriting system in the usual sense over a set of function symbols Σ and $R' = \{l = r \mid l \longrightarrow r \in R\}$,*
2. *$(E, >)$ is a ground complete ordered rewriting system over a set of function symbols Γ,*
3. *there is a well-founded monotonic total ordering $\succ$ on $\mathcal{T}(\Sigma \cup \Gamma)$ such that $\succ \supseteq >$ and $\sigma l \succ \sigma r$ for each rule $l \longrightarrow r \in R$ and ground substitution σ, and*
4. *there are no critical pairs in the sense of definition 1 between E and R' w.r.t. $\succ$.*

Then $(E \cup R', \succ)$ is a ground complete ordered rewriting system.

Proof Condition 3 ensure that $(E \cup R', \succ)$ is terminating. Due to condition 4 the only critical pairs of $(E \cup R', \succ)$ are those of E or of R and hence are ground joinable. Thus $(E \cup R', \succ)$ is ground complete. □

As a corollary we see immediately that the combination of any of the theories we have considered above with new free function symbols (so R is empty) gives a ground complete ordered rewriting system.

If R and E are both proved terminating using the lexicographic path ordering or Knuth-Bendix ordering, and assuming that the operator precedences are consistent on $\Sigma \cap \Gamma$, we may obtain a total ordering $\succ$ by constructing a total operator precedence on $\Sigma \cup \Gamma$ which subsumes the two partial precedences. Thus we may combine any of the examples above with any such R.

4.11 AC with Two Rules

In this section we show how with a suitable choice of ordering two rules suffice for a ground complete AC rewriting system. Let E be

$$\begin{aligned} (x * y) * z &\longrightarrow x * (y * z) \\ x * y &= y * x \end{aligned}$$

and $>$ any monotonic ordering which is total on ground terms and satisfies for all ground terms x, y, z and all constants a, b

$$\begin{aligned} (x * y) * z &> x * (y * z) \\ x * y &> y * x \qquad \text{if } x > y \\ a &> b * x \qquad \text{if } a > b. \end{aligned}$$

We show that $(E, >)$ is ground complete. Notice that the ordering $>_2$ of the previous section has the required properties. (In fact it is not hard to see that any ordering with these properties is not well-founded).

We must first prove termination. Suppose that $s_1 \Longrightarrow s_2 \Longrightarrow \cdots$ is an infinite chain of rewrites. Since each equivalence class is finite it must contain a loop $s_i \Longrightarrow s_{i+1} \Longrightarrow \cdots \Longrightarrow s_{i+k} \Longrightarrow s_i$. But since $s \Longrightarrow t$ implies $s > t$ we have $s_i > s_{i+1} > \cdots > s_i$, which contradicts the definition of $>$. Thus $(E, >)$ is terminating.

To prove ground confluence we first observe

Lemma 11 *Let x and y be ground terms and let S_x be the multiset of all constants occurring in X. If $x =_E y$ then $S_x = S_y$.*

Proof S_x is invariant when applying the equations of E. □

Then we can prove

Theorem 1 *Let $(E, >)$ be as above. Then*

1. *If $w \in \mathcal{T}_G$ then*

$$w \Longrightarrow^* a_1 * (a_2 * \cdots * (a_{n-1} * a_n) \ldots)$$

where $S_w = \{a_1, \ldots, a_n\}$ and $a_1 \leq a_2 \leq \cdots \leq a_n$, and this expression is irreducible.

2. *$(E, >)$ is a ground complete ordered rewriting system.*

Proof

1. The proof is by induction on $n = |S_w|$. If $n \leq 2$ the result is clear. Now by applying associativity we may assume $w \Longrightarrow^* a * v$ where $S_v = S_w - \{a\}$. By induction we may assume that v has the required form, so that in particular $v = b * u$, where $b \leq c$ for each $c \in S_u = S_v - \{b\}$. Now if $a \leq b$ we are done, so assume that $a > b$. Then

$$w \Longrightarrow^* a * (b * u) \Longrightarrow^* (b * u) * a \Longrightarrow^* b * (u * a).$$

Now by induction $u * a$ rewrites to the required form, $a_2 * (a_3 \cdots * a_n)$, and since $b < a$ and $b \leq c$ for each $c \in S_u$ we have $b \leq a_i$ for each $i = 2, \ldots, n$. So

$$w \Longrightarrow^* a_1 * (a_2 * (\cdots * a_n)),$$

where $a_1 = b$. It is clear that this expression is irreducible.

2. Suppose that v, w are ground and $v =_E w$. Then $S_v = S_w$, so by part 1

$$v \Longrightarrow^* a_1 * (a_2 * (\cdots * a_n))$$

and

$$w \Longrightarrow^* a_1 * (a_2 * (\cdots * a_n))$$

where $S_v = \{a_1, \ldots, a_n\}$ and $a_1 \leq \cdots a_n$. Thus v and w are joinable. Hence $(E, >)$ is ground confluent, and as it is terminating it is ground complete.

□

4.12 Abelian Groups

Let E be

$$\begin{aligned}
x * y &= y * x \\
x * (y * z) &= y * (x * z) \\
(x * y) * z &= x * (y * z) \\
x * i(x) &= 1 \\
1 * x &= x \\
x * 1 &= x \\
x * (i(x) * y) &= y \\
i(x * y) &= i(x) * i(y) \\
i(i(x)) &= x \\
i(1) &= 1
\end{aligned}$$

and $>$ any AC compatible ordering which orders the last seven equations from left to right for all ground terms x, y, and has

$$a_1 < i(a_1) < a_2 < i(a_2) < \cdots < a_n < i(a_n)$$

where the constants are $a_1, \ldots, a_n$. The lexicographic path ordering with precedence $* < i < a_1 < \ldots < a_n$ will do this. We shall show that $(E, >)$ is ground complete.

Unfortunately our automated ground confluence checking procedure fails in this case as it has to reduce arbitrary terms of the form $x * (y_1 * (y_2 * (\cdots (y_n * (i(x) * z)\ldots)$. But we may prove ground completeness using the technique of the previous example.

If a is a constant and t is a term we define the polarity of a in t as $p(a, \_) : \mathcal{T}_G \longrightarrow \mathbf{Z}$ inductively as

$$\begin{aligned} p(a,a) &= 1 \\ p(a,b) &= 0 \\ p(a,i(x)) &= -p(a,x)) \\ p(a,s*t) &= p(a,s) + p(a,t) \end{aligned}$$

where s and t are terms and b is any constant distinct from a. Thus for example $p(a, i(i(a) * (t * a))) = -p(t)$.

Lemma 12 *If x and y are ground terms with $x =_E y$ then for each i $p(a_i, x) = p(a_i, y)$.*

Proof It is easy to check that $p(a_i, x)$ is invariant under application of any of the equations. □

Now we have

Theorem 2 *Let $(E, >)$ be as above. Then*

1. *If $w \in \mathcal{T}_G$ and $w \neq_E 1$ then*

$$w \Longrightarrow^* e_1^{p_1} * e_2^{p_2} \cdots * e_n^{p_n},$$

(assumed associated to the right) where for each i we have $p_i = |p(a_i, w)|$, and $e_i = a_i$ if $p(a_i, w) \geq 0, e_i = i(a_i)$ if $p(a_i, w) < 0$. Furthermore each such expression is irreducible.

2. *$(E, >)$ is a ground complete ordered rewriting system.*

Proof

1. If $w \in \mathcal{T}_G$ then by applications of the last seven equations it is easy to see that

$$w \Longrightarrow^* e_1 * (e_2 * \cdots * e_n)\ldots)$$

where each $e_i \in \{a_j, i(a_j) \mid j = 1, \ldots, n\}$. Now as $a_1 < i(a_1) < a_2 < i(a_2) \cdots < a_n < i(a_n)$ an argument similar to the previous theorem shows that we may assume $e_1 \leq e_2 < \cdots < e_n$. Now applying (7) we see that each expression of the form $u * (a_i * (i(a_i) * z))$ reduces to $u * v$, and thus

$$w \Longrightarrow^* e_1^{p_1} * e_2 p_2 * \cdots * e_n^{p_n},$$

where each a_j is a_j or $i(a_j)$. Now by the lemma each $p_i = |p(a_i, w)|$. It is clear that this expression is irreducible.

2. Since the ordering is well-founded, $(E, >)$ is terminating.
To prove confluence suppose that $u =_E v$ with u, v ground. We have by the lemma that $p(a_j, u) = p(a_j, v)$ for each j. But then by the first part u and v are joinable. Thus $(E, >)$ is ground confluent. □

Acknowledgements

This paper was written while the second author was at the Laboratory for Computer Science at MIT and the first author was visiting there. We would both like to acknowledge the generous hospitality of John Guttag.

References

[1] L. Bachmair, N. Dershowitz, D. Plaisted: *Completion Without Failure*, Proc. Coll. on Resolution of Equations in Algebraic Structures (1987).

[2] R.S. Boyer, J.S. Moore: *A Computational Logic Handbook*, Academic Press (1988).

[3] N. Dershowitz: *Termination of Rewriting*, Journal of Symbolic Computation (1987) 3, 69-116.

[4] G. Huet: *Confluent Reductions: Abstract properties and Applications to Term Rewriting Systems*, Journal ACM 27, 4 (1980), 797-821.

[5] G. Huet, D.C. Oppen: *Equations and Rewrite Rules - A Survey*, in: Formal Languages: Perspectives and Open Problems, R. Book (ed.), Academic Press (1982).

[6] J. Hsiang, M. Rusinowitch: *On Word Problems in Equational Theories*, Proc. ICALP'87, LNCS 267 (1987), 54-71.

[7] J.-P. Jouannaud, H. Kirchner: *Completion of a Set of Rules Modulo a Set of Equations*, SIAM Journal of Computing 15 (1986), 1155-1194.

[8] D.E. Knuth, P.B. Bendix: *Simple Word Problems in Universal Algebras*, in: Computational Problems in Abstract Algebra, ed J. Leech, Pergamon 1970, 263-297

[9] U. Martin: *Doing Algebra with Reve*, Report UMCS-86-10-4, Dept. of Comp. Sci., University of Manchester (1986).

[10] U. Martin: *How to Choose the Weights in the Knuth Bendix Ordering*, in: Rewriting Techniques and Applications, LNCS 256, 1987, 42-53.