J. Symbolic Computation (1988) **6**, 19–36

Only Prime Superpositions Need be Considered in the Knuth-Bendix Completion Procedure†

Deepak Kapur, David R. Musser, Paliath Narendran

Computer Science Branch
Corporate Research and Development
General Electric Company
Schenectady, NY 12345

(Received 4 October 1985)

The Knuth and Bendix test for local confluence of a term rewriting system involves generating superpositions of the left-hand sides, and for each superposition deriving a critical pair of terms and checking whether these terms reduce to the same term. We prove that certain superpositions, which are called *composite* because they can be split into other superpositions, do not have to be subjected to the critical-pair-joinability test; it suffices to consider only *prime* superpositions. As a corollary, this result settles a conjecture of Lankford that *unblocked* superpositions can be omitted. To prove the result, we introduce new concepts and proof techniques which appear useful for other proofs relating to the Church-Rosser property. This test has been implemented in the completion procedures for ordinary term rewriting systems as well as term rewriting systems with associative-commutative operators. Performance of the completion procedures with this test and without the test are compared on a number of examples in the Rewrite Rule Laboratory (RRL) being developed at General Electric Research and Development Center.

1. INTRODUCTION

Knuth and Bendix (1970) developed a completion procedure which in many cases generates a complete (canonical) term rewriting system from a given term rewriting system. This complete term rewriting system can serve as a decision procedure for the equational theory of the term rewriting system given as the input. Extensions of the Knuth-Bendix procedure have been developed to handle associative and commutative operators by Lankford and Ballantyne (1977) as well as Peterson and Stickel (1981). The Knuth-Bendix completion procedure and its extensions can also be used for checking consistency of a class of theories, algebraic specifications of abstract data types in particular (Musser and Kapur, 1982). This observation also points to the use of the Knuth-Bendix completion procedure and its extensions to prove theorems by induction under certain conditions (Musser, 1980; Huet and Hullot, 1980; Jouannaud and Kounalis, 1986; Kapur, Narendran and Zhang, 1986; Kapur and Musser, 1987). Because of the many applications of these completion procedures, computer implementations have been developed (Knuth and Bendix, 1970; Lankford and Ballantyne, 1977; Musser, 1980; Peterson and Stickel, 1981; Lescanne, 1983; Guttag et al, 1984; Kapur and Sivakumar, 1984; Kapur, Sivakumar, and Zhang, 1986) and in some cases the implementations have been integrated into a verification system (Musser, 1980).

† Partially supported by the National Science Foundation grant no. DCR-8408461.

0747–7171/88/040019+18 $03.00/0

The Knuth-Bendix completion procedure and its extensions involve generating terms, called *superpositions*, from the left-hand sides of rules and checking whether different applications of rules will produce different results; the result obtained from applying two rules on such a superposition is called a *critical pair* (of terms). New rules are generated from those critical pairs whose terms turn out not to be equivalent with respect to the rewriting system so far obtained. The original term rewriting system is augmented with these new rules and the process is repeated until it is no longer possible to generate new rules or until a critical pair is generated which cannot be made into a rule. (Termination is not guaranteed.)

The performance of the Knuth-Bendix completion procedure and its extensions depends mainly upon the number of superpositions generated and the time involved in checking whether or not they lead to a new rule (Kapur and Sivakumar, 1984). In 1979, Buchberger introduced a criterion for detecting unnecessary critical pairs for computing a Gröbner basis of a polynomial ideal over a field; this criterion was subsequently generalized by Winkler and Buchberger (1983) to term rewriting systems. In this paper, we show that certain kinds of superpositions, called *composite* superpositions, do not have to be considered at all in the completion process. As a corollary, this proves a conjecture that, to our knowledge, was first made by Dallas Lankford: that *unblocked* superpositions do not have to be considered, where an unblocked superposition is one whose unifying substitution contains terms that are reducible.

In showing that it is unnecessary to consider composite superpositions, we introduce a new proof technique for proving the Church-Rosser property. This technique and the concepts underlying it may be useful in proving the Church-Rosser property in other situations. We believe that the proof of the Church-Rosser property in case of associative-commutative functions is simpler and more natural than proofs reported in the literature; it especially exhibits the need for extensions of rules as proposed by Lankford and Ballantyne (1977) as well as Peterson and Stickel (1981). The proofs are done by considering *flattened* terms and introducing a new way to identify subterms within a flattened term.

We have implemented the Knuth-Bendix completion procedure and its extension for associative-commutative operators (usually referred to as *AC completion procedure*) developed by Lankford and Ballantyne (1977) and Peterson and Stickel (1981) including a test for blocked superpositions. The implementation is part of *RRL*, a *Rewrite Rule Laboratory* under development at the General Electric Corporate Research and Development Center (Kapur and Sivakumar, 1984; Kapur, Sivakumar, and Zhang, 1986). In this paper, we also discuss our experience with these versions of the completion procedures on a number of examples. Our results show that although fewer superpositions have to be considered than in the usual implementations of the completion procedure, this does not necessarily mean we obtain better performance by including the test for unblocked superpositions for ordinary term rewriting systems. The time for the test itself might be enough to outweigh the savings gained by omitting the processing of composite superpositions. Our experiments confirm that this often is the case. For the AC completion procedure, however, the test for unblocked superpositions results in considerable savings because of (a) unnecessary substitutions being generated during extensions of rules, and (b) the associative and commutative unification algorithm generating

non-minimal unifiers. It should be noted that these results also apply to Gröbner basis algorithms.

The next subsection is a brief discussion of related work. Section 2 introduces the definition of a composite superposition and a new definition of positions as a way to identify subterms within a flattened term as well as other needed definitions and properties in the theory of term rewriting systems. Section 3 is the main result, where it is proved that composite superpositions do not need to be considered for showing confluence of a term rewriting system. No attempt has been made to prove the correctness of completion procedures employing such a criterion; however, we believe that such proofs can be done using an elegant approach proposed by Bachmair and Dershowitz (1986). Section 4 compares composite superpositions with a criterion for eliminating certain critical pairs developed by Winkler and Buchberger (1984). Section 5 is a discussion of the performance of the completion procedures with this check in RRL on a number of examples.

1.1 Related Work

The concept of blocked substitutions was first introduced by Slagle (1974). Lankford (1975) generalized it and reported favorable experiments using the blocking concept; see also Bledsoe (1977). A generalization of blocking methods to theories with permutative axioms (which include associative-commutative axioms) with favorable experiments was reported in (Lankford and Ballantyne, 1979).

Independently Buchberger (1979) developed a criterion for detecting unnecessary reductions in the computation of a Gröbner basis of a polynomial ideal over a field; a Gröbner basis of a polynomial ideal is a complete system for the polynomial ideal when polynomials are viewed as rewrite rules. Subsequently, Winkler and Buchberger (1984) generalized Buchberger's criterion to include term rewriting systems; see also (Winkler, 1984). Küchlin (1985) extended their results and further developed their criterion based on the generalized Newman's lemma proposed by Buchberger as well as the notions of connectedness and subconnectedness. More recently, Bachmair and Dershowitz (1986) have provided a general framework in which various criteria for unnecessary critical pairs can be studied and the correctness of completion procedures using such criteria can be established.

2. DEFINITIONS

Let $>$ be a well-founded partial ordering on a set S; i.e., there is no infinite sequence of the form $a_1 > a_2 > a_3 > \cdots$ consisting of elements from S. A well-founded partial order $>$ on S can be extended to the set of finite *multisets* on S as follows: $M_1 \gg M_2$ if and only if $\forall x \in M_2 - M_1 \; \exists y \in M_1 - M_2$ such that $y > x$ (Dershowitz and Manna, 1979).

Let $\rightarrow$ be a binary relation on a set S, referred to as a *rewriting relation*. The reflexive, transitive closure of a rewriting relation $\rightarrow$ is denoted by $\rightarrow^*$, and is referred to as *reduction*. An element p in S is said to be *in normal form* if and only if there is no q such that $p \rightarrow q$. If $p \rightarrow^* q$ and q is in normal form, then q is said to be a *normal form* of p. Elements p and q are *joinable* if and only if there exists an r such that $p \rightarrow^* r$ and $q \rightarrow^* r$.

A rewriting relation $\rightarrow$ is said to be *finitely terminating* or *Noetherian* if and only if there is no infinite sequence of the form $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots$. The relation $\rightarrow$ is

- *confluent* if and only if for all p, q, r, if $p \rightarrow^* q$ and $p \rightarrow^* r$ then q and r are joinable.
- *locally confluent* if and only if for all p, q, r, if $p \rightarrow q$ and $p \rightarrow r$ then q and r are joinable.

Local confluence is an easier property to test for than confluence, and it can be shown that if $\rightarrow$ is both Noetherian and locally confluent then it is confluent (see (Huet, 1980), for instance, for a proof). A rewriting relation $\rightarrow$ that is both Noetherian and confluent is said to be *complete* (or *canonical*).

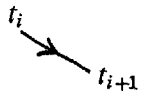
The reflexive, symmetric and transitive closure of $\rightarrow$, denoted by $\leftrightarrow^*$, is an equivalence relation. A rewriting relation $\rightarrow$ is said to have the *Church-Rosser property* if and only if for all p, q such that $p \leftrightarrow^* q$, p and q are joinable. It can be easily shown that $\rightarrow$ has the Church-Rosser property if and only if $\rightarrow$ is confluent.

It is the Church-Rosser property that makes rewriting relations useful in dealing with equational theories of a given set of axiom equations: treat equations as rewrite rules; if the rewriting relation has the Church-Rosser property, then the search for a chain of applications of the axioms connecting a given pair of terms is reduced to the search for a common term to which they are both reducible. If the rewriting relation is also Noetherian, one just has to reduce each term to its normal form and check for identity of the resulting terms.

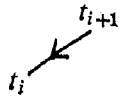
2.1 Mountain Ranges, Peaks, Crevices, and Valleys

The condition $t \leftrightarrow^* u$ is equivalent to the existence of elements $t_0, t_1, \ldots, t_n$ such that $t = t_0$, $t_n = u$ and, for $i = 0, 1, \ldots, n\text{-}1$ either $t_i \rightarrow t_{i+1}$ or $t_{i+1} \rightarrow t_i$. Henceforth, we refer to sequences of elements such as $t_0, t_1, \ldots, t_n$ as *chains*.

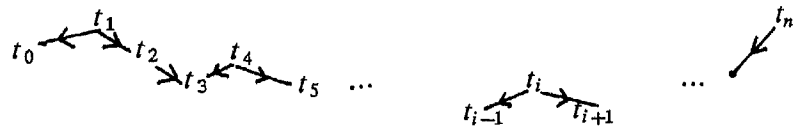
If $\rightarrow$ is Noetherian, then the reduction relation $\rightarrow^*$ is a well-founded partial ordering and we can consistently represent chains pictorially by the following conventions: a pair of elements t_i, t_{i+1} such that $t_i \rightarrow t_{i+1}$ is drawn with t_i above t_{i+1}:



and a pair t_i, t_{i+1} such that $t_{i+1} \rightarrow t_i$ is represented as

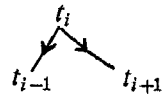


so that the chain $t_0, t_1, \ldots, t_n$ appears as a "mountain range"

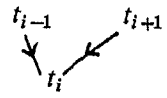


in which each triple t_{i-1}, t_i, t_{i+1} is one of three types:

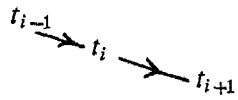
a) a *peak*: $t_i \rightarrow t_{i-1}$ and $t_i \rightarrow t_{i+1}$



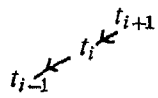
b) a *crevice*: $t_{i-1} \rightarrow t_i$ and $t_{i+1} \rightarrow t_i$



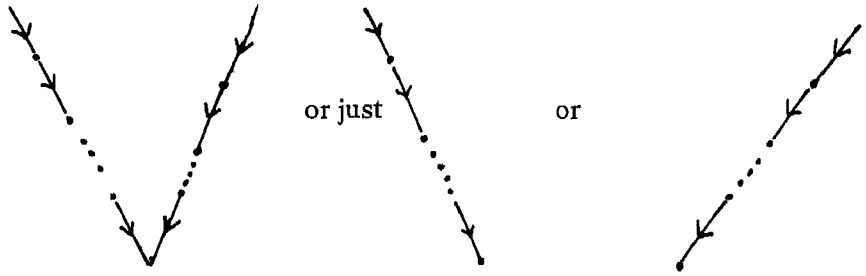
c) a *hillside*: $t_{i-1} \rightarrow t_i$ and $t_i \rightarrow t_{i+1}$



or $t_i \rightarrow t_{i-1}$ and $t_{i+1} \rightarrow t_i$



A chain is called a *valley* if it has no peaks. Thus a valley looks like



(We could also define a *mountain* to be be a chain with no crevices. But we will make no use of this concept in this paper.)

The existence of a valley connecting two elements t and u is equivalent to joinability of t and u. The Church-Rosser property could now be rephrased as:

for every pair of elements $<t, u>$, t and u are connected by a mountain range if and only if they are connected by a valley.

In the proof of the main result in Section 3, we will make extensive use of this terminology.

2.2 Flattened Terms and Positions

Let F be a finite set of function symbols and V be a denumerable set of variables. Some function symbols in F can have the associative and commutative properties; such functions are called AC (associative-commutative) functions. Without any loss of generality, an AC function f can be assumed to be of an arbitrary arity; this allows us to consider terms constructed using AC functions in *flattened* form, that is, no argument to an AC function f is a term whose outermost symbol is f itself.

By $T(F,V)$ we denote the set of all possible flattened terms that can be constructed using F and V. For a term t, $Vars(t)$ denotes the set of all variables that occur in t. For example, $Vars(f(x,y,g(y))) = \{x,y\}$. The *size* of a flattened term s is the number of occurrences of function and variable symbols in s and is denoted by $|s|$.

A *subterm position* within a flattened term is a finite sequence whose members are either positive integers or finite sets of positive integers, separated by ".". Given a term t, we define its subterm positions and the corresponding subterm at each subterm position as follows:

1. To the null sequence, denoted by λ, corresponds the entire term, t.
2. If $f(t_1, \cdots, t_n)$ is the subterm at position I, then
   a. if $1 \leq j \leq n$, then the subterm at position $I.j$ is t_j.
   b. if f is an AC operator and $\{ i_1, \ldots, i_k \} \subseteq \{1, 2, \ldots, n\}$, where $k \geq 2$, then the subterm at position $I.\{ i_1, \ldots, i_k \}$ is $f(t_{i_1}, \ldots, t_{i_k})$.

We write t/i for the subterm of t at position i. Note that $t/i.j = (t/i)/j$. For the term $f(t_1,\ldots,t_n)$, where f is an AC operator, the positions $\{1, 2, \ldots, n\}$ and λ are equivalent.

The above definition permits more than one position to describe the same subtree of the tree structure of t, e.g., if $+$ is AC, the position of the subterm c in $h(+(a,b,c))$ could be given as either 1.3 or as 1.{1, 3}.2 or as 1.{2, 3}.2, but each subterm can be uniquely described by a position in which only the last member could be a set. We call such a position *canonical*. Henceforth, we consider only canonical positions.

We define the operation of replacing a subterm within a term by another term as follows:

$$t[\lambda] \leftarrow s = s,$$

$$f(t_1,\ldots,t_{i-1},t_i,t_{i+1},\ldots,t_n)[i] \leftarrow s = f(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_n)$$
if either f is not an AC function or the root of s is not f,
$$= f(t_1,\ldots,t_{i-1},s_1,\ldots,s_m,t_{i+1},\ldots,t_n) \text{ if } f \text{ is AC and } s = f(s_1,\ldots,s_m).$$

$$f(t_1,\ldots,t_n)[\{i_1,\ldots,i_k\}] \leftarrow s = f(t_{j_1},\ldots,t_{j_{k'}}, s)$$
if f is an AC function and the root of s is not f,
$$= f(t_{j_1},\ldots,t_{j_{k'}}, s_1,\ldots,s_m) \text{ if } f \text{ is AC and } s = f(s_1,\ldots,s_m),$$

where $\{ i_1,\ldots,i_k \} \subset \{1, \ldots, n\}$ and $\{ j_1,\ldots,j_{k'} \} = \{1, \ldots, n\} - \{ i_1,\ldots,i_k \}$.
Note that $f(t_1,\ldots,t_{i-1},t_i,t_{i+1},\ldots,t_n)[i.p] \leftarrow s = f(t_1,\ldots,t_{i-1},t_i\,[p] \leftarrow s,t_{i+1},\ldots,t_n)$ where $p \neq \lambda$.

Given any two positions p_1 and p_2, let p be the longest prefix common to p_1 and p_2, i.e., $p_1 = p.q_1$ and $p_2 = p.q_2$ and q_1 and q_2 do not have a common prefix. Positions p_1 and p_2 are related in one of the following four ways:

1. (i) Either $q_1 = \lambda$, or
   (ii) $q_1 = I$, where I is a finite set of integers, and (a) q_2 is a subset of I or (b) the first element of q_2 is an element of I. In this case, we say that $p_1 \leq p_2$.

2. $p_2 \leq p_1$.
3. Both q_1 and q_2 are sets of integers such that $q_1 \cap q_2$ is neither empty nor equal to q_1 or q_2. In this case p_1 and p_2 are said to be *properly intersecting*.
4. Otherwise, p_1 and p_2 are disjoint. This is the case when neither $q_1 = \lambda$ nor $q_2 = \lambda$, and (a) q_1 and q_2 are sets of integers, and their intersection is empty, (b) q_1 does not contain the first element of q_2 or vice versa, or (c) the first element of q_1 is different from the first element of q_2.

Note that if F does not have any AC function, the case (3) above of properly intersecting positions does not arise as positions are then sequences of positive integers.

Given two terms t and s, $t =_{AC} s$ if and only if

(i) t and s are the same variable or

(ii) $t = f(t_1,\ldots,t_n)$ and $s = f(s_1,\ldots,s_m)$, and $n = m$, and

(a) f is an AC function and the multiset $\{ t_1, \ldots, t_n \}$ is the same multiset as $\{ s_1, \ldots, s_m \}$ using $=_{AC}$ to check equality of elements, or

(b) f is not an AC function and for each $1 \leq i \leq n$, $t_i =_{AC} s_i$.

Henceforth, by equality on terms, we mean $=_{AC}$; by equality on multisets of terms, we mean the equality of multisets using $=_{AC}$ on terms.

2.3 Composite and Prime Superpositions

Let $T = \{ L_i \rightarrow R_i \}$ be a term rewriting system where L_i and R_i are terms. The term rewriting system T induces a rewriting relation denoted as $\rightarrow$, in the following way: $t \rightarrow t'$ if and only if there is a position p in t, a rule $L_i \rightarrow R_i$ in T, and a substitution σ such that $t/p =_{AC} \sigma(L_i)$, where t/p stands for the subterm position p in t, and $t' = t[p] \leftarrow \sigma(R_i)$, the term obtained by replacing the subterm at position p in t by the term $\sigma(R_i)$.

In order to handle certain cases that arise with AC functions, it is useful to define, corresponding to T, an *extended* term rewriting system T^e, obtained by adding certain rules to T: for every rule $f(t_1,\ldots,t_n) \rightarrow t$ in T where f is an AC function, the rule $f(t_1,\ldots,t_n,z) \rightarrow t'$ is in T^e if $f(t_1,\ldots,t_n,z) \rightarrow t'$ is not an instance of $f(t_1,\ldots,t_n) \rightarrow t$, where z is a variable not occurring in $f(t_1,\ldots,t_n)$ and t' is the flattened term corresponding to $f(t,z)$. Clearly (i) if T has no rules having AC function symbols as their outermost function symbols, then $T^e = T$, (ii) T^e has the same rewriting relation as T, and (iii) $(T^e)^e = T^e$. This definition of T^e is similar to the notion of *extension* in Peterson and Stickel (1981) which has properties (i) and (ii), but not (iii) (although (iii) does hold for the notion of extension actually implemented by Peterson and Stickel).

As an illustration, consider a term rewriting system for free abelian groups:

$$\{ x + i(x) \rightarrow 0, \quad x + 0 \rightarrow x \},$$

where '+' is an AC function. The extended term rewriting system for the above is:

$$\{ x + i(x) \rightarrow 0, \quad x + i(x) + z \rightarrow 0 + z, \quad x + 0 \rightarrow x \},$$

where the rule $x + i(x) + z \rightarrow 0 + z$ is an extension of the rule $x + i(x) \rightarrow 0$.

A *generalized superposition* of T (T^e) is a 6-tuple $\langle u, h, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta \rangle$ in which u is a term, h, i are positions in u such that $h \leq i$ or h and i are properly intersecting, $L_1 \rightarrow R_1$ and $L_2 \rightarrow R_2$ are

rewrite rules in T (T^e), and θ is a substitution such that $u/h =_{AC} \theta(L_1)$ and $u/i =_{AC} \theta(L_2)$. Corresponding to a generalized superposition is a *generalized critical pair*, namely, the pair of terms $<u[h] \leftarrow \theta(R_1), u[i] \leftarrow \theta(R_2)>$.

For example, $<y + i(y) + i(i(y))$, $\{1, 2\}$, $x_1 + i(x_1) \rightarrow 0$, $\{2,3\}$, $x_2 + i(x_2) \rightarrow 0$, $\{ x_1 \leftarrow y, x_2 \leftarrow i(y) \}>$ is a generalized superposition in which the positions are properly intersecting;
$<y + i(y) + i(i(y))$, λ, $x_1 + i(x_1) + z \rightarrow 0 + z$, $\{2,3\}$, $x_2 + i(x_2) \rightarrow 0$, $\{ x_1 \leftarrow y, x_2 \leftarrow i(y), z \leftarrow i(i(y)) \}>$ is another generalized superposition in which the positions are not properly intersecting.

A *superposition* is a generalized superposition of the form $<\theta(L_1), \lambda, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$, where L_1/i is a nonvariable subterm of L_1 and θ is the most general unifier of L_1/i and L_2. We also refer to $\theta(L_1)$ as "the superposition" as is usually done in the literature. The generalized critical pair of a superposition is referred to simply as a *critical pair*. These definitions are equivalent to the usual definitions of superpositions and critical pairs.

Two generalized superpositions $<u, h, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$ and $<u', h', L_1' \rightarrow R_1', i', L_2' \rightarrow R_2', \theta'>$ are called *equivalent* if and only if $u =_{AC} u'$ and their generalized critical pairs $<u[h] \leftarrow \theta(R_1), u[i] \leftarrow \theta(R_2)>$ and $<u'[h'] \leftarrow \theta'(R_1'), u'[i'] \leftarrow \theta'(R_2')>$ are equivalent, i.e., $u[h] \leftarrow \theta(R_1) =_{AC} u'[h'] \leftarrow \theta'(R_1')$ and $u[i] \leftarrow \theta(R_2) =_{AC} u'[i'] \leftarrow \theta'(R_2')$. In the example discussed above, the two generalized superpositions are equivalent.

Lemma 1: For every generalized superposition of a term rewriting system T in which the positions are properly intersecting, there is an equivalent generalized superposition of its extended term rewriting system T^e in which the positions are not properly intersecting.

Using this lemma, we do not need to consider generalized superpositions in which positions are properly intersecting, a special situation that arises only because of the presence of AC function symbols in a term rewriting system. Before giving the proof of the lemma, we show that in the example discussed above, for the generalized superposition $<y + i(y) + i(i(y))$, $\{1,2\}$, $x_1 + i(x_1) \rightarrow 0$, $\{2,3\}$, $x_2 + i(x_2) \rightarrow 0$, $\{ x_1 \leftarrow y, x_2 \leftarrow i(y) \}>$ in which the positions are properly intersecting, an equivalent generalized superposition in which the positions are not properly intersecting is obtained using the extension of the rule in the superposition as

$$<y + i(y) + i(i(y)), \lambda, x_1 + i(x_1) + z_1 \rightarrow 0 + z_1, \{2,3\},$$
$$x_2 + i(x_2) \rightarrow 0, \{ x_1 \leftarrow y, x_2 \leftarrow i(y), z_1 \leftarrow i(i(y)) \}>.$$

This lemma also reveals the role played by extensions of rules.

Proof: Let $G = <u, h, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$ be a generalized superposition in which positions are properly intersecting. This implies that the outermost function of both L_1 and L_2 is an AC function, say f. Let $u/h = f(t_1,\ldots,t_k)$ and $u/i = f(s_1,\ldots,s_{k'})$. Let $h = p.h'$ and $i = p.i'$; further, let $h_1 = h' - i'$ and $i_1 = i' - h'$, where $-$ is the set difference.

Let $g' = h' \cup i_1$ and $g = p.g'$. It is easy to see that g is a position in u, and both u/h and u/i are subterms of u/g. In fact, u/g is the smallest subterm of u containing both u/h and u/i. Now we are in a position to construct a generalized

superposition G' equivalent to G as follows: Let $L_1{}^f$ and $R_1{}^f$ be the flattened terms obtained from $f(L_1, z_1)$ and $f(R_1, z_1)$, where z_1 is a new variable not appearing in L_1. Then, $G' = <u, g, L_1' \rightarrow R_1', i, L_2 \rightarrow R_2, \theta'>$; $L_1' \rightarrow R_1'$ is (i) $L_1{}^f \rightarrow R_1{}^f$ if $L_1{}^f \rightarrow R_1{}^f$ is not an instance of $L_1 \rightarrow R_1$, and (ii) otherwise, $L_1 \rightarrow R_1$.

If $L_1' \rightarrow R_1' = L_1 \rightarrow R_1$ meaning that $L_1{}^f \rightarrow R_1{}^f$ is an instance of $L_1 \rightarrow R_1$ by the substitution σ, then for each variable x in $Vars(L_1)$, $\theta'(x) = \theta \cup \{z_1 \leftarrow u/p.i_1\}(\sigma(x))$. Otherwise, if $L_1' \rightarrow R_1'$ is an extension of $L_1 \rightarrow R_1$, i.e., L_1' is a flattened term obtained by flattening $f(L_1, z_1)$, then $\theta'(z_1) = u/p.i_1$ and $\theta'(x) = \theta(x)$ otherwise.

It is easy to verify that the generalized critical pair of G' is equivalent to the generalized critical pair of G. □

Because of the above lemma, we only need to concentrate on generalized superpositions in which positions are not properly intersecting.

A generalized superposition $<u, h, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$ is called *composite* if and only if either (i) h and i are properly intersecting and at least one of u/h and u/i has a proper reducible subterm, or (ii) $h \leq i$ and u/i has a proper reducible subterm.[1] A noncomposite generalized superposition is called *prime*.

Note that in Lemma 1, a generalized superposition G' of T^e equivalent to G of T is so constructed that if G is prime, then G' is also prime.

Lemma 2: Every composite generalized superposition of T has an equivalent generalized superposition of T^e that (i) is either prime or (ii) factors into two prime generalized superpositions of T^e.

Proof: Consider a composite generalized superposition $G = <u, h, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$ of T. If h and i are properly intersecting, by Lemma 1 above there is an equivalent generalized superposition of T^e in which the positions are not properly intersecting; if this equivalent generalized superposition is prime, we are done. Otherwise, we only need to consider composite generalized superpositions of T^e in which positions are not properly intersecting.

Let j be a maximal nontrivial position such that $u/i.j$ is reducible. By renaming, if necessary, the variables of the rule, say $L_3 \rightarrow R_3$, by which $u/i.j$ is reducible, and modifying the substitution θ, we can without loss of generality assume $\theta(L_3) = u/i.j$. Then G, j and $L_3 \rightarrow R_3$ determine two other prime generalized superpositions.

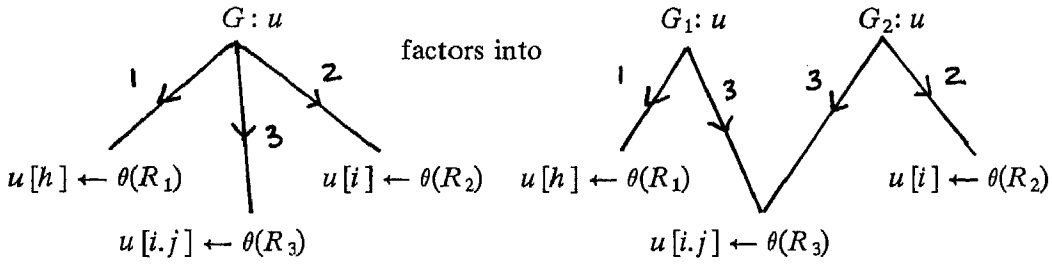
$$G_1 = <u, h, L_1 \rightarrow R_1, i.j, L_3 \rightarrow R_3, \theta>$$

$$G_2 = <u, i, L_2 \rightarrow R_2, i.j, L_3 \rightarrow R_3, \theta>.$$

G_1 and G_2 are called *factors* of G, and we say that G *factors into* G_1 and G_2.

1. The requirement that the subterm be proper is essential, as an example at the end of this section illustrates.

We have the following picture.



in which we have also shown the generalized critical pairs corresponding to G, G_1 and G_2. □

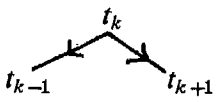
3. MAIN RESULT

Theorem 3. Let T^e be an extended Noetherian term rewriting system corresponding to a term rewriting system T such that that every critical pair corresponding to a prime superposition of T^e is joinable. Then T (T^e) has the Church-Rosser property.

Proof: Note that T and T^e have the same rewriting relation. Let $>_{T^e}$ be a well-founded partial ordering on terms induced by the term rewriting system T^e as follows: $s_1 > s_2$ if and only if $s_1 \rightarrow^+ s_2$. As mentioned before, $>_{T^e}$ can be extended to a well-founded partial ordering $\gg_{T^e}$ on multisets of terms. Given terms t and u, let $t_0, t_1, \ldots, t_n$ be any mountain range connecting t and u. We show how to transform this mountain range into a valley, by showing how to eliminate the peaks.

For any peak, we will show how to replace it by a valley. In some cases this may increase the number of peaks, but nevertheless, the number of peaks must eventually decrease to zero, because the mountain range itself is being reduced in $\gg_{T^e}$.

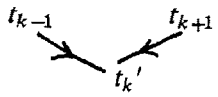
If $t_0, t_1, \ldots, t_n$ has no peaks, we are done. Otherwise, let k be any index, $1 \leq k \leq n-1$, such that t_{k-1}, t_k, t_{k+1} is a peak:



Let $L_1 \rightarrow R_1$ and $L_2 \rightarrow R_2$ be the rules applied to t_k to produce t_{k-1} and t_{k+1}, respectively. There are two cases:

a) The rules are applied at disjoint positions, say i and j:
$t_{k-1} = t_k[i] \leftarrow \theta(R_1)$ and $t_{k+1} = t_k[j] \leftarrow \theta(R_2)$.

Then this peak can be replaced by a crevice:



Both t_{k-1} and t_{k+1} have positions, say j' and i', respectively, such that $t_{k-1}/j' = \theta(L_2)$ and $t_{k+1}/i' = \theta(L_1)$; further we get,

$t_k' = t_{k-1}[j'] \leftarrow \theta(R_2) = t_{k+1}[i'] \leftarrow \theta(R_1)$

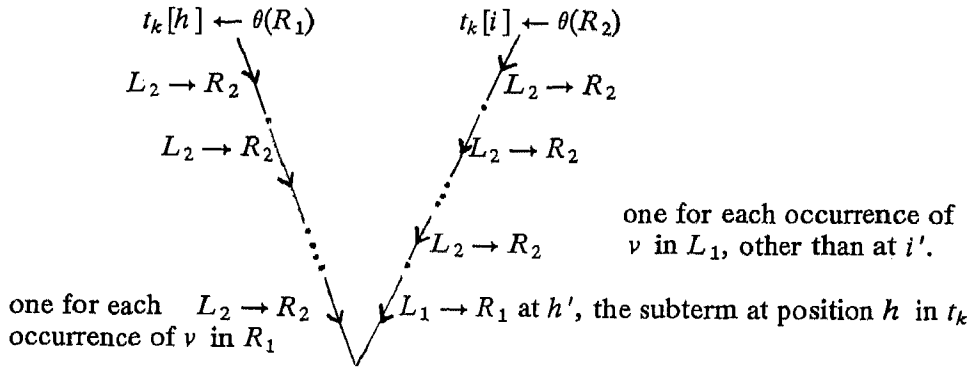
The resulting multiset of terms is lower in the multiset ordering than the original.

b) $<t_{k-1}, t_{k+1}>$ is the generalized critical pair of a generalized superposition $G = <t_k, h, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$ of T, where $t_{k-1} = t_k[h] \leftarrow \theta(R_1)$ and $t_{k+1} = t_k[i] \leftarrow \theta(R_2)$. There are two subcases:

(i) positions are properly intersecting: as shown by Lemma 1, there is an equivalent generalized superposition of T^e, say G', in which positions are not properly intersecting and G' has an equivalent generalized critical pair $<t_{k-1}, t_{k+1}>$. Thus we only need to consider generalized superpositions in which positions are not properly intersecting and then, the second case below applies.

(ii) $h \leq i$, i.e., $i = h.\bar{i}$: There are two subcases in this subcase:

1) L_1/i' is a variable, v, for some prefix i' of position $\bar{i}$ (possibly $i' = \bar{i}$). In this case the peak can be replaced by a valley as follows:



In the above picture, the left side corresponds to repeatedly rewriting t_{k-1} at positions of v in R_1 using $L_2 \rightarrow R_2$; the right side corresponds to rewriting remaining positions of v in L_1 using $L_2 \rightarrow R_2$ so that $L_1 \rightarrow R_1$ can be applied, which is always possible. It is easy to see that the two sides give the same result.

2) $L_1/\bar{i}$ is a nonvariable subterm. In this case there is some superposition $S = <\sigma(L_1), \lambda, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \sigma>$ where σ is a factor of θ, say $\theta = \theta' \sigma$. The reader is reminded that critical pairs among rules are constructed considering all possible non-variable subterms of the left-hand side of a flattened rule (see the definition of a subterm of a flattened term in Section 2.2); this is different from the way critical pairs are constructed in Peterson and Stickel's

approach (1981), where terms are not flattened (see also comments later on implementation in Section 5.2).

There are two subcases:

i) G is prime. Then S must also be prime. By assumption the critical pair $<\sigma(R_1), \sigma(L_1)[\bar{i}] \leftarrow \sigma(R_2)>$ is joinable; in other words it is connected by a valley. By applying the substitution θ' to the terms of this valley, we obtain a valley connecting t_{k-1} and t_{k+1}.

ii) G is composite. In this case, by Lemma 2, we can split G itself into two prime generalized superpositions and then proceed as in the previous case to replace the corresponding pair of peaks by a pair of valleys. This is all done in one step so that the result is lower in the multiset ordering than the original.

In each of these cases, the multiset of terms that results is lower in the multiset ordering than the original. Thus, this process of replacing peaks by valleys must eventually terminate with no peaks; i.e., with one valley connecting t and u. □

A more usual approach to such proofs has been to establish that joinability of critical pairs implies local confluence, and hence confluence under the assumption of the Noetherian property. The Church-Rosser property then comes from its equivalence to confluence. We can use this equivalence to obtain a confluence result:

Corollary 4. Let T be a Noetherian term rewriting system such that every critical pair corresponding to a prime superposition is joinable. Then the rewriting relation of T is confluent.

Note that if we assume joinability of all critical pairs, then in the proof of Theorem 3, we could simplify case 2 to proceed always as in case 2i, but for all superpositions instead of just those that are prime. The result would be a new proof of the theorem that the Noetherian property and joinability of all critical pairs of rules in an extended term rewriting system imply that the Church-Rosser property holds. The concepts of mountain ranges, peaks, and valleys used in the proof may be of use in proofs of the Church-Rosser property under other assumptions. We believe that the above proof is simpler than similar proofs in (Peterson and Stickel, 1981) which use the notion of T-compatibility and show the role of extension rules to ensure
T-compatibility.

The role played by the extension rules for rules involving AC functions is also quite evident in the above proof; it is to ensure that a generalized superposition in which the positions are properly intersecting can be replaced by an equivalent generalized superposition in which the positions are not properly intersecting. From the proof of Lemma 1, it is obvious that

(i) an extension rule is necessary only for a rule satisfying the following two conditions:

(a) the outermost symbol of the left-hand side of the rule is an AC function, and

(b) the extension rule is not an instance of the rule,

(ii) the right-hand sides of an extended term rewriting system can be made irreducible, and

(iii) no critical pairs between two extensions need be considered.

These cases are related to the conditions handled by Peterson and Stickel's approach; see (pp. 255-257 in Peterson and Stickel, 1981). The reason for (i) and (ii) follows from the fact that a term rewriting system is confluent (has the Church-Rosser property) if a reduced system, obtained from it by throwing away rules whose left-hand-sides are reducible and replacing reducible right-hand-sides of rules by their respective normal forms, is confluent. The reason for (iii) follows from the proof of Lemma 1 and the fact that a superposition generated from two extensions is always composite.

As another corollary, we show that "blocked" superpositions are sufficient. A superposition is *blocked* if all terms of the unifier substitution are in normal form, otherwise, a superposition is called *unblocked*. It is easy to see that an unblocked superposition is always composite, for if s and t are two nonvariable terms and θ is their most general unifier one of whose terms is reducible, then $\theta(s)$ $(= \theta(t))$ has a reducible proper subterm. On the other hand, a blocked superposition can either be composite or prime.

Corollary 5. Let T be a Noetherian term rewriting system such that every critical pair corresponding to a blocked superposition is joinable. Then T has the Church-Rosser property (or, equivalently, is confluent).

Proof: Follows directly from the fact that every unblocked critical pair is composite and therefore does not have to be checked for joinability. □

Recall that we defined a generalized superposition $<u, h, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$ to be composite only when u/i has a reducible *proper* subterm. The following example shows that this requirement is essential. Consider the following rewriting system:

1. $g(f(x, x)) \rightarrow i(x)$
2. $f(x, a(y)) \rightarrow j(x)$
3. $f(a(x), y) \rightarrow j(a(x))$

If we only require that for a composite superposition, any subterm of u/i be reducible, then the only prime superposition for the above example is obtained from rules 2 and 3, and is $f(a(x), a(y))$. The corresponding critical pair is $<j(a(x)), j(a(x))>$ which is joinable. The superposition $g(f(a(x), a(x)))$ between rules 1 and 2 and between rules 1 and 3 will be composite by our new definition because rules 2 and 3 can be applied at position 1. However, the rewriting system is not Church-Rosser because the terms $g(j(a(x)))$ and $i(a(x))$ are equivalent but are not joinable. It is obvious from this example that only the critical pair between rules 2 and 3 and either one of the two critical pairs, between 1 and 2 and between 1 and 3 respectively, need be considered. We believe this can be generalized by imposing an order on the pair of rules being considered in the completion procedure as in (Buchberger, 1979).

The correctness of completion procedures using the above criterion for not considering certain critical pairs can be proved using an elegant approach developed by Bachmair and Dershowitz (1986).

4. COMPARISON WITH WINKLER AND BUCHBERGER'S CRITERION

As mentioned earlier, Buchberger (1979) developed a criterion to detect unnecessary critical pairs for the Gröbner basis algorithm which generates a complete basis for polynomial ideals. Winkler and Buchberger (1983) generalized this criterion to apply to term rewriting systems; see also (Winkler, 1984). Their criterion for eliminating unnecessary critical pairs can be briefly summarized as follows:

(C1) The critical pair corresponding to a superposition $p = <\theta(L_1), \lambda, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta>$ can be discarded if there are positions j and k, a rule $(L_3 \rightarrow R_3)$ and superpositions $q = (\sigma(L_1), \lambda, L_1 \rightarrow R_1, j, L_3 \rightarrow R_3, \sigma)$ and $r = <\delta(L_3), \lambda, L_3 \rightarrow R_3, k, L_2 \rightarrow R_2, \delta>$ such that $i = j.k$, $\theta(L_1)$ is an instance of $\sigma(L_1)$, $\theta(L_1)/j$ is an instance of $\delta(L_3)$ and both q and r have already been considered.

Note that if L_3/k is not a non-variable subterm of L_3, for no substitution β will $r' = < \beta(L_3), \lambda, L_3 \rightarrow R_3, k, L_2 \rightarrow R_2, \beta >$ be a superposition. Thus the condition is trivially satisfied and consequently, the critical pair corresponding to p will not be discarded. This shows that their criterion is different from ours.

An example will illustrate this: let $L_1 = f(g(x_1, f(h(x_1))))$, $L_2 = h(a)$ and $L_3 = g(a, y_3)$. The superposition

$$< f(g(a, f(h(a)))), \lambda, L_1 \rightarrow R_1, 1.2.1, L_2 \rightarrow R_2, \{x_1 \leftarrow a\}>$$

will not be discarded because there is no superposition between L_2 and L_3.

4.1 Modification of the Winkler-Buchberger criterion

(C2) The critical pair corresponding to a superposition $p = < \theta(L_1), \lambda, L_1 \rightarrow R_1, i, L_2 \rightarrow R_2, \theta >$ can be discarded if there are positions j and k and a rule $(L_3 \rightarrow R_3)$ such that $i = j.k$, neither j nor k is λ and $\theta(L_1)$ is reducible at position j by rule $(L_3 \rightarrow R_3)$.

This is just a sort of "mirror image" of the criterion discussed earlier in the paper, in that instead of looking "below" position i in the tree for a reducible subterm, we look "above" it. Composite and prime superpositions can be defined this way too, composite superpositions being exactly those that can be discarded using criterion (C2). But proving the correctness of (C2) is trickier, because every composite superposition defined this way need not factor into exactly *two* prime superpositions.

Example: $L_1 = f(g(x_1, g(x_1, g(y_1, x_1))))$, $L_2 = g(y_2, a)$, $L_3 = g(a, g(x_3, y_3))$.

Superposing L_1 and L_2 we get the term $f(g(a, g(a, g(y, a))))$, which is reducible at positions 1 and 1.2 by the rule $(L_3 \rightarrow R_3)$. But it is not hard to see that the superposition

$$< f(g(a, g(a, g(y, a)))), \lambda, L_1 \rightarrow R_1, 1.2.2, L_2 \rightarrow R_2, \{x_1 \leftarrow a, y_1 \leftarrow y, y_2 \leftarrow y\}>$$

cannot be factored into *two* prime generalized superpositions. (It can however be factored into *three* prime superpositions.)

5. IMPLEMENTATION OF THE CHECK FOR UNNECESSARY SUPERPOSITIONS

We have incorporated in RRL (Kapur and Sivakumar, 1984; Kapur, Sivakumar, and Zhang, 1986) the check for unnecessary superpositions in the implementations of the Knuth-Bendix completion procedure as well as of the AC completion procedure.

5.1 Non-AC Completion Procedure

We have observed the performance of the completion procedure with the check for composite superpositions on a number of examples including various axiomatizations of free groups discussed in (Kapur and Sivakumar, 1984). In comparing its performance with the completion procedure without the check, we observed that in most examples the number of rules generated to obtain a complete system was less than the number of rules generated without the check; the total number of critical pairs generated when the check is performed also turned out less. However, the total *time* taken for generating a complete system was usually more than the time taken when the check was not made, thus indicating that the time spent in looking for unnecessary superpositions is often more than the time saved by discarding critical pairs corresponding to them. This was the case also when the check for composite superpositions was replaced by the check for unblocked superpositions.

In the case of the classical three-equations axiomatization of free groups, we found that when we oriented the associativity axiom as $(x * (y * z)) \rightarrow ((x * y) * z)$ (i.e. the status of $*$ in the lexicographic recursive path ordering used for showing termination of rewriting systems was right-to-left) and used E1 strategy for generating critical pairs (where the earliest generated smallest rule is superposed with all earlier generated rules in the completion procedure) (cf. Kapur and Sivakumar, 1984), the number of rules and prime critical pairs generated in the completion procedure with the check for composite superpositions (32 rules, 105 prime critical pairs, 20 composite critical pairs) was more than the number of rules and total number of critical pairs generated when the check was not included in the completion procedure (27 rules, 96 critical pairs).

For the same example, when L1 strategy is used for generating critical pairs (the latest generated smallest rule is superposed with all earlier generated rules), we surprisingly found that the number of rules and prime superpositions generated in the completion procedure with the check for composite superpositions (30 rules, 108 prime critical pairs, 18 composite critical pairs) was the same as the number of rules and superpositions generated by the completion procedure without any check (30 rules, 108 critical pairs).

There is a possibility of the completion procedure encountering an unorientable rule if the check for composite (or unblocked) superpositions is incorporated whereas the unorientable rule may not be encountered for that rewriting system if the check is not incorporated; in our experiments, we observed this on a couple of examples. The reverse case is also possible, i.e., the completion procedure without the check encounters an unorientable rule whereas such a rule is not encountered in the case of the completion procedure with the check; however, we did not see such behavior in our experiments. Neither of these cases poses any problem in generating complete systems using the RRL or the REVE system (Lescanne, 1983;

Forgaard and Guttag, 1984 in Guttag et al, 1984) because of the option to postpone the consideration of a critical pair when such a situation arises.

5.2 AC Completion Procedure

We found that when used in the presence of AC functions, the check for unblocked superpositions pays off a great deal; this check was found easier to implement that the check for composite superpositions. The implementation in RRL for generating blocked superpositions is done differently from an implementation implied by the proof of Theorem 3. Instead of generating superpositions of all non-variable top-level subterms of the left-hand side of an AC rule in T (a rule is called an AC rule if and only if the outermost function symbol of its left-hand side is AC) with the left-hand side of another AC rule, the extension of the first rule is used for generating superpositions with the second rule; this turned out to be an easier and more efficient way to generate all blocked superpositions. For this reason, extensions of rules are used even if they are instances of the original rules.

For examples involving more than one AC-operator or a mixture of AC and non-AC operators, the savings are quite substantial. For instance, for the free distributive lattice example (FDL) in (Peterson and Stickel, 1981), the time spent in normalization to generate a complete system is reduced by a factor of 4 (from 103 seconds to 27 seconds), whereas time spent in testing for blocked superpositions is less than 1/13 of the original normalization time; similarly, for the boolean ring example, the normalization time is reduced to 1/3 (from 2 minutes to 41 sec.) with less than 1/10 of the original normalization spent in the check for blocked superpositions. For the example specifying Milner's nondeterministic machines in (Hullot, 1980), the normalization time is again reduced from approx. 5 minutes to 44 seconds; the time spent in checking blocked superpositions is only 21 seconds. For small examples such as abelian group, the time saved in normalization is approximately the same as the time spent in the check; for a free commutative ring with unity, the check does reduce the total time, but only marginally.

Our observation is that this is so because unification algorithms for terms involving associative and commutative operators (Stickel, 1981; Fages, 1984; Fortenbacher, 1985) often generate unifiers which are either non-minimal or duplicates of others. Further, one sees many such unnecessary unifiers when terms involving more than one AC-operator, or AC as well as non-AC operators, are unifiers. For the AC completion procedure, it is sufficient to consider all minimal unifiers for generating superpositions. There are two alternatives available in such a case in an implementation: (i) filter out unnecessary unifiers using associative-commutative matching, as suggested by Fages, or (ii) filter the superpositions generated by such unifiers using the test for blocked superpositions developed in this paper. We have adopted the latter approach, as the associative-commutative matching operation is also expensive.

Another factor is the generation of unnecessary superpositions when two extension rules are overlapped. We have noted that in many examples, the number of unblocked critical pairs is almost the same as the number of blocked ones and sometimes even more than the number of blocked critical pairs. In all examples we ran, the number of unblocked critical pairs was over 40% of the number of blocked critical pairs. As a result, we have made the check for blocked critical

pairs an integral part of the completion procedure whenever overlaps between rules with AC-operators are considered.

ACKNOWLEDGMENT: We thank G. Sivakumar for implementing the check for blocked critical pairs in RRL and running some experiments for us. We have also benefited a great deal from discussions with him on contrasting our criterion with that of Winkler and Buchberger. We also thank Hantao Zhang and the referees for their helpful comments and suggestions.

REFERENCES

Bachmair, L., and Dershowitz, N. (1986). Critical pair criteria for the Knuth-Bendix completion procedure. Unpublished manuscript, Dept. of Computer Science, University of Illinois at Urbana-Champaign. Extended abstract appeared in the Proc. of *SYMSAC'86*.

Bledsoe, W.W. (1977). Non-resolution theorem proving. *Artificial Intelligence*, 9, 1-35.

Buchberger, B. (1979). A criterion for detecting unnecessary reductions in the construction of Gröbner Bases. Proceedings of *EUROSAM 79*, Lecture Notes in Computer Science 72, Springer Verlag, 3-21.

Dershowitz, N., and Manna, Z. (1979). Proving termination with multiset orderings. *Comm. ACM*, 22/8, 465-476.

Fages, F. (1984). Associative-commutative unification. Proc. of *7th International Conference on Automated Deduction*, Lecture Notes in Computer Science 170, Springer Verlag.

Fortenbacher, A. (1985). An algebraic approach to unification under associativity and commutativity. Proc. of *First International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science , Springer Verlag.

Guttag, J.V., Kapur, D., Musser, D.R. (eds.) (1984). Proceedings of an *NSF Workshop on the Rewrite Rule Laboratory* Sept. 6-9 Sept. 1983, General Electric Research and Development Center Report 84GEN008.

Huet, G. (1980). Confluent reductions: abstract properties and applications to term rewriting systems. *JACM*, 27/4, 797-821.

Huet, G., and Hullot, J.M. (1980). Proofs by induction in equational theories with constructors. *21st IEEE Symposium on Foundations of Computer Science*, 96-107.

Hullot, J.M. (1980). A catalogue of canonical term rewriting systems. Technical Report CSL-13, Computer Science Laboratory, SRI International, Menlo Park.

Jouannaud, J., and Kirchner, H. (1984). Completion of a set of rules modulo a set of equations. In: Proc. of *Principles of Programming Languages (POPL)*.

Jouannaud, J., and Kounalis, E. (1986). Automatic proofs by induction in equational theories without constructors. Proc. of *Symposium on Logic in Computer Science*, 358-366.

Kapur, D., and Musser, D.R. (1987). Proof by consistency.. *Artificial Intelligence*, 31, 125-157.

Kapur, D., Narendran, P., and Zhang, H. (1986). Proof by induction using test sets. *8th Intl. Conf. on Automated Deduction*, Lecture Notes in Computer Science, Springer Verlag.

Kapur, D. and Sivakumar, G. (1984) Architecture of and experiments with RRL, a Rewrite Rule Laboratory. In: Reference 4, 33-56.

Kapur, D., Sivakumar, G., and Zhang, H. (1986). RRL: A Rewrite Rule Laboratory. *8th Intl. Conf. on Automated Deduction*, Lecture Notes in Computer Science, Springer Verlag.

Knuth, D.E. and Bendix, P.B. (1970). Simple word problems in universal algebras. In: *Computational Problems in Abstract Algebras.* (ed. J. Leech), Pergamon Press, 263-297.

Küchlin, W. (1985). A confluence criterion based on the generalized Newman Lemma. In: *EUROCAL'85* (ed. Caviness) 2, Lecture Notes in Computer Science, 204, Springer Verlag, 390-399.

Lankford, D.S. (1975). Canonical inference. Univ. of Texas, Math. Dept. Automatic Theorem Proving

Project, Austin, Texas, Report ATP-32.

Lankford, D.S., and Ballantyne, A.M. (1977). Decision procedures for simple equational theories with commutative-associative axioms: complete sets of commutative-associative reductions. Dept. of Math. and Computer Science, University of Texas, Austin, Texas, Report ATP-39.

Lankford, D.S., and Ballantyne, A.M. (1979). The refutation completeness of blocked permutative narrowing and resolution. Proc. of *Fourth Workshop on Automated Deduction*, Austin, Texas.

Lescanne, P. (1983). Computer experiments with the REVE term rewriting system generator. Proc. of *10th Principles of Programming Languages (POPL)* Conference.

Musser, D.R. (1980). On proving inductive properties of abstract data types. Proc. *7th Principles of Programming Languages (POPL)*.

Musser, D.R. (1980). Abstract data types in the AFFIRM system. *IEEE Trans. on Software Engineering*, SE-6/1.

Musser, D.R., and Kapur, D. (1982). Rewrite rule theory and abstract data type analysis. *Computer Algebra, EUROCAM, 1982* (ed. Calmet), Lecture Notes in Computer Science 144, Springer Verlag, 77-90.

Peterson, G.L., and Stickel, M.E. (1981). Complete sets of reductions for some equational theories. *JACM*, 28/2, 233-264.

Slagle, J. (1974). Automated theorem proving for theories with simplifiers, commutativity and associativity. *J. ACM*, 4, 622-642.

Stickel, M.E. (1981). A unification algorithm for associative-commutative functions. *Journal of the ACM*, 28/3, 423-434.

Winkler, F., and Buchberger, B. (1983). A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm. In: Proc. of the *Coll. on Algebra, Combinatorics and Logic in Computer Science*.

Winkler, F. (1984). The Church-Rosser property in computer algebra and special theorem proving: an investigation of critical pair, completion algorithms. Dissertation der J. Kepler-Universitaet, Linz, Austria.