Solving Term Inequalities

Gerald E Peterson†

225/105/2/206 mailcode 106 5165
McDonnell Douglas Corporation
P. O. Box 516, St. Louis, MO 63166
peterson@mdc.com

Abstract

This work pertains to the Knuth-Bendix (KB) algorithm which tries to find a complete set of reductions from a given set of equations. In the KB algorithm a term ordering is employed and it is required that every equation be orientable in the sense that the left-hand side be greater than the right. The KB algorithm halts if a non-orientable equation is produced. A generalization of the KB algorithm has recently been developed in which every equation is orientable and which halts only when a complete set is generated. In the generalization a constraint is added to each equation. The constraint governs when the equation can be used as a reduction. The constraint is obtained from the equation by "solving" the term inequality *left-hand side* > *right-hand side*. To understand what it means to solve a term inequality, consider the analogy with algebra in which solving term equalities, i.e. *unification*, is analogous to solving algebraic equalities. Then solving term inequalities is analogous to solving algebraic inequalities. Thus, the solution of term inequalities relates to unification as the solution of algebraic inequalities relates to the solution of algebraic equalities. We show how to solve term inequalities when using the lexicographic path ordering.

Introduction

In a landmark paper, Knuth and Bendix (Knuth & Bendix 1970) described a technique for attempting to generate a complete set of reductions from the axioms of an equational theory. If a complete set of reductions could be found, then it embodied a decision procedure for equations provable from the axioms. In particular they generated a complete set of reductions for groups.

In their theory a term ordering was introduced and it was necessary for every equation to be orientable in the sense that one side be larger than the other in the ordering. Equations such as commutativity which could not be oriented would cause the process to fail. Knuth and Bendix were well aware of this limitation of their method. In fact, they proposed using an approach in which a non-orientable equation would be made into two reductions, one which rewrites the left-hand side to the right-hand side and another which rewrites the right-hand side to the left-hand side and in which a method is available to "make sure that no infinite looping occurs when reducing words to a new kind of irreducible form." This proposed approach has been effected (in (Peterson 1990) and forthcoming publications) by adding a *constraint* to each reduction. For example, commutativity is represented as $x \cdot y \rightarrow y \cdot x$ **if** $x > y$.

By using constraints such as these we are able to generalize the entire Knuth-Bendix process. In fact, complete sets of constrained reductions are now known for common equational theories including Abelian groups, rings, and Boolean algebras, and there is an analogue of the Knuth-Bendix procedure which begins with the axioms of an equational theory and attempts to generate a complete set of reductions. This constrained version of the Knuth-Bendix process will never fail because a non-orientable equation shows up—every equation can be oriented. It may, of course, fail because it runs forever trying to find a complete set. The process reduces to that of Knuth and Bendix when presented with equations which are orientable without the use of constraints.

As an example, the following three reductions taken collectively is a complete set equivalent to associativity and commutativity.

1. $(x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z)$
2. $x \cdot y \rightarrow y \cdot x$ **if** $x > y$
3. $x \cdot (y \cdot z) \rightarrow y \cdot (x \cdot z)$ **if** $x > y$.

In reduction 1 there is no constraint. This means that the left-hand side is always greater than the right. In each of the other two reductions there is a constraint which is a necessary and sufficient condition that the left-hand side be greater than the right. In order to find the constraint we must solve the inequality *left-hand side* > *right-hand side*. Looking back at the example, we see

† This work was supported by the McDonnell Douglas Independent Research and Development Program.

that $x > y$ is the solution to $x \cdot y > y \cdot x$ and $x > y$ is also the solution to $x \cdot (y \cdot z) > y \cdot (x \cdot z)$. The manner in which these kinds of solutions are obtained is the subject of the present work.

It will be helpful to consider how the solution of term inequalities relates to unification. In unification, a term equality, such as $(x \cdot y) \cdot (x \cdot a) = (w \cdot z) \cdot y$ is given and it is required to find a most general substitution for the variables which makes the equality true. In this case, the substitution is $\{x \leftarrow w, y \leftarrow w \cdot a, z \leftarrow w \cdot a\}$. This process could be named *solving term equalities* but *unification* has been chosen instead. The problem of solving term inequalities is similar. Given a term inequality, such as $x \cdot (y \cdot z) > z \cdot y$, find the region in term space where the inequality is true. In this case it turns out to be where $x > z$ or $x = z$ or $y > z$ or $y = z$. Consider an analogy to algebra. The solution in real numbers of a polynomial equation is a finite (possibly empty) set analogous to the single most general unifier of a term equation. The solution of a polynomial inequality, however, consists of a region, possibly with several disjoint intervals, such that the inequality is true if and only if the variable is in the region. For example, $x^2 - 5x + 4 > 0$ if and only if $x > 4$ or $x < 1$. Thus we may expect the solution of a term inequality to be a similar type of region in term space.

So far we have been implicitly assuming that an underlying term ordering existed. Throughout this paper we use the *lexicographic path ordering*. Knuth and Bendix used an ordering which has since become known as the *Knuth-Bendix ordering* (Dershowitz 1987). However, it has a more complex definition than the lexicographic path ordering and is, therefore, presumed more difficult to work with. A possible research problem is to figure out how to solve term inequalities when using the Knuth-Bendix ordering.

Related Work

In (Kirchner & Kirchner 1989) a theoretical overview of constrained equational reasoning is presented. They work modulo an equational theory such as associativity and commutativity. However, they assume that constraints are based on term equalities, wheras we assume constraints are based on term inequalities. In (Comon 1990) it is shown than the question of the existence of solutions to term inequalities based on the lexicographic path ordering, such as those herein, is decidable, but NP-hard.

The automatic solution of algebraic inequalities has a similar goal (i.e. the isolation of variables) to that of solving term inequalities, but is accomplished by completely different methods. A good beginning reference to the solution of algebraic inequalities is (Sacks 1987).

An Example

In order to provide further motivation for the introduction of a technique for solving term inequalities, we present an example of the operation of the constrained Knuth-Bendix algorithm. We emphasize those places in which term inequalities are solved. Details of how the constrained Knuth-Bendix algorithm works in general will be described elsewhere. The reader is assumed familiar with the ordinary Knuth-Bendix algorithm throughout this section.

The following problem is from page 91 of (Monthly 1968).

Suppose S is a semigroup in which $x^2 = x^3$ and $x^2y = yx^2$ for all x, y in S. Then $(xy)^2 = x^2y^2$.

We will describe how a solution can be mechanically obtained using the constrained Knuth-Bendix algorithm. We present only the part that directly leads to the desired equation. Other equations will be generated in an actual implementation.

A semigroup has one associative operation. Thus, the first reduction is

1. $(x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z)$.

The equation $x^2 = x^3$ is orientable because $x \cdot (x \cdot x) > x \cdot x$ for every x. So the second reduction is

2. $x \cdot (x \cdot x) \rightarrow x \cdot x$.

The equation $x^2y = yx^2$, i.e., $x \cdot (x \cdot y) = y \cdot (x \cdot x)$, is not orientable, so the ordinary KB algorithm does not apply to this problem. However, the inequality $x \cdot (x \cdot y) > y \cdot (x \cdot x)$ may be solved and the solution is $x > y$. Thus the equation $x^2y = yx^2$ may be represented as the two reductions

3. $x \cdot (x \cdot y) \rightarrow y \cdot (x \cdot x)$ **if** $x > y$,
4. $y \cdot (x \cdot x) \rightarrow x \cdot (x \cdot y)$ **if** $y > x$.

If we unify the left-hand side of 2 with the $x \cdot y$ subterm of 1 and proceed as in the ordinary KB process, we obtain the new reduction

5. $x \cdot (x \cdot (x \cdot y)) \rightarrow x \cdot (x \cdot y)$.

Unifying the left-hand side of 3 with the subterm $x \cdot (x \cdot y)$ of 5 and applying 3 and 5 once each to the result gives the equality

$$x \cdot (y \cdot (x \cdot x)) = x \cdot (x \cdot y)$$

which is produced only if the constraint of 3, i.e. $x > y$, holds. Now we try to reduce this equality under the assumption $x > y$. Since $x > y$ implies the constraint of reduction 3, the right-hand side can be reduced by 3 giving

$$x \cdot (y \cdot (x \cdot x)) = y \cdot (x \cdot x).$$

No further reductions are possible. At this point, the inequality $x > y$ which came from one of the parents

is discarded and the new equality is oriented on its own merits. In this case the equality is orientable and we add

6. $x \cdot (y \cdot (x \cdot x)) \rightarrow y \cdot (x \cdot x)$

as a new reduction.

Similarly, if we unify the left-hand side of 4 with the $x \cdot y$ subterm of 1, we obtain the equation $x \cdot (x \cdot (y \cdot z)) = y \cdot (x \cdot (x \cdot z))$. We now find $x > y$ as the solution of the term inequality $x \cdot (x \cdot (y \cdot z)) > y \cdot (x \cdot (x \cdot z))$ and add the reductions

7. $x \cdot (x \cdot (y \cdot z)) \rightarrow y \cdot (x \cdot (x \cdot z))$ **if** $x > y$,
8. $y \cdot (x \cdot (x \cdot z)) \rightarrow x \cdot (x \cdot (y \cdot z))$ **if** $y > x$.

Finally, unifying the left-hand side of 7 with the left-hand side of 2 and reducing once with 7 and 2, respectively, yields the equation

$$y \cdot ((y \cdot z) \cdot ((y \cdot z) \cdot z)) = (y \cdot z) \cdot (y \cdot z)$$

under the always valid constraint $y \cdot z > y$. We now reduce this equality to normal form using 1, 6 and 5, and obtain

$$y \cdot (y \cdot (z \cdot z)) = y \cdot (z \cdot (y \cdot z))$$

which is the equality we are trying to prove. Note that if this were now added as a reduction, the solution of a term inequality would again be required.

The Lexicographic Path Ordering

Recall that terms are formed from function symbols (including constants) and variables. The definition of the lexicographic path ordering depends on a given linear ordering on the function symbols. Therefore, we assume there are only a finite number of function symbols and they are put into a linear order. This ordering of function symbols and the lexicographic path ordering on terms will both be denoted by $>$ because either it is possible to figure out from the context whether $>$ is the term ordering or the function symbol ordering, or they are both true simultaneously.

Ground terms contain no variables. Two ground terms are equal if and only if they are identical.

The lexicographic path ordering (lpo) is defined only for ground terms. It is defined recursively as follows (Dershowitz 1987):

Definition. *Suppose*

$$s = f(s_1, \cdots, s_m) \text{ and } t = g(t_1, \cdots, t_n)$$

are ground terms. (We may have $m = 0$ and/or $n = 0$ for a constant.) Then $s > t$ if and only if

L1 *$s_i \geq t$ for some i, $1 \leq i \leq m$, or*

L2 *$f > g$ and $s > t_j$ for every j, $1 \leq j \leq n$, or*

L3 *$f = g$ and for some k, $1 \leq k \leq n$, we have $s_i = t_i$ whenever $1 \leq i < k$, $s_k > t_k$, and $s > t_i$ whenever $k < i \leq n$.*

Note that the first inequality in **L2** is from the function symbol ordering, and in the last part of **L3** we do mean $s > t_i$, not $s_i > t_i$. The two examples given next should help clear up any confusion related to this definition. First, however, we note that it is possible to prove (Dershowitz 1987) that the lpo is a well-ordering of the set of ground terms and it satisfies the *subterm property*, i.e., if s is a strict subterm of t, then $t > s$.

For the first example, suppose the function symbol order is $\cdot > a > b$ and it is required to show that $a \cdot b > b \cdot a$. According to **L1** and **L3** of the definition, this will be so if and only if

$$a \geq b \cdot a \text{ or} \quad (1)$$
$$b \geq b \cdot a \text{ or} \quad (2)$$
$$a > b \text{ and } a \cdot b > a \text{ or} \quad (3)$$
$$a = b \text{ and } b > a. \quad (4)$$

In line (3) the first predicate is true by the given function symbol order and the second predicate is true by the subterm property. Thus we have shown what is required. Incidently, lines (1), (2) and (4) are all false.

As the second example let's determine the larger of $a \cdot (b + c)$ and $a + (b \cdot c)$ when the function symbol order is $+ > \cdot > a > b > c$. Since these two terms are not equal, one must be larger than the other. We can use the definition above to test $a \cdot (b + c) > a + (b \cdot c)$. Since $+ > \cdot$, **L2** and **L3** do not apply. By **L1**, we have $a \cdot (b + c) > a + (b \cdot c)$ if and only if

$$a \geq a + (b \cdot c) \text{ or} \quad (5)$$
$$b + c \geq a + (b \cdot c). \quad (6)$$

But (5) is false by the subterm property. By **L1** and **L3**, (6) is true if and only if

$$b \geq a + (b \cdot c) \text{ or}$$
$$c \geq a + (b \cdot c) \text{ or}$$
$$b > a \text{ and } b + c > b \cdot c \text{ or}$$
$$b = a \text{ and } c > b \cdot c.$$

But all of these are clearly false. Thus we do not have $a \cdot (b + c) > a + (b \cdot c)$. So we must have $a + (b \cdot c) > a \cdot (b + c)$.

Solving Term Inequalities

Using the definition of the lexicographic path ordering, we can determine the larger of any two *ground* terms. Suppose, however, that a given inequality contains variables, and the question is to determine where it is satisfied. This can be done by recursively applying the

definition of the lpo and gathering all the bottom-level expressions which cannot be simplified. To clarify the meaning of this, let's do a couple of examples.

Example 1. *Solve* $x \cdot y > y \cdot x$. By **L1** and **L3** of the definition, $x \cdot y > y \cdot x$ if and only if

$$x \geq y \cdot x \text{ or} \tag{1}$$
$$y \geq y \cdot x \text{ or} \tag{2}$$
$$x > y \text{ and } x \cdot y > x \text{ or} \tag{3}$$
$$x = y \text{ and } y > x. \tag{4}$$

But (1) and (2) are false by the subterm property and the fact that no term equals a subterm of itself. Also (4) is false since it consists of two incompatible statements. In (3), the second predicate is true by the subterm property. This leaves only the first predicate of (3). Thus the answer is $x > y$.

In order to simplify the presentation we will from now on omit the *or* from the end of each line and we will replace each *and* with a comma.

Example 2. *Solve* $x \cdot y > y \cdot a$ *given that* $a > \cdot$ *in the function symbol ordering.* This example will begin to show the complexity that can arise. By **L1** and **L3**, $x \cdot y > y \cdot a$ if and only if

$$x > y \cdot a \tag{5}$$
$$x = y \cdot a \tag{6}$$
$$y > y \cdot a \tag{7}$$
$$y = y \cdot a \tag{8}$$
$$x > y,\ x \cdot y > a \tag{9}$$
$$x = y,\ y > a. \tag{10}$$

By the subterm property, (7) is false. Since y and $y \cdot a$ are not unifiable, (8) is false. Replacing the equalities by equivalent unifiers and expanding the second predicate in (9) using **L1** gives the following form for the solution

$$x > y \cdot a \tag{11}$$
$$\{x \leftarrow y \cdot a\} \tag{12}$$
$$x > y,\ x > a \tag{13}$$
$$x > y,\ x = a \tag{14}$$
$$x > y,\ y > a \tag{15}$$
$$x > y,\ y = a \tag{16}$$
$$\{x \leftarrow y\},\ y > a. \tag{17}$$

However, note that (11), (12), (15) and (16) each implies (13) by the subterm and transitive properties of $>$. Thus (11), (12), (15) and (16) can be eliminated from the above leaving

$$x > y,\ x > a$$
$$\{x \leftarrow a\},\ a > y$$
$$\{x \leftarrow y\},\ y > a.$$

as the solution.

We now present an algorithm which can be used to solve any given term inequality. It is presented as a person would use it with pencil and paper. From a machine standpoint it is not very efficient as presented. It can be made more efficient at the expense of greater complexity and less understandability. Because of limited space, efficiency considerations will be deferred to a later paper.

Begin with the given inequality on a line by itself. As we proceed, each intermediate stage will consist of a table of one or more lines of the form

$$\theta, e_1, e_2, \cdots, e_n$$

where θ is a substitution (possibly empty) and each element e_i is either an equality or an inequality. An inequality is *fully reduced* if it has one of the forms $v > t$ or $t > v$, where v is a variable and t is a term which does not contain v. A line is a *component* if it consists of a substitution θ followed by 0 or more fully reduced inequalities which do not contain the variables that are replaced in θ.

Algorithm. *Repeat steps 1 to 3 below until each line is a component.*

1. *Move down the table to the first line which is not a component and across this line from left to right to the first element which is not a fully reduced inequality. Call this element e and its line ℓ.*
2. *If e is an equality, unify the two sides. If unification is impossible, delete line ℓ. If the table is now empty,* stop*; the answer is* false*. If unification is possible, let σ be the mgu. Delete e in line ℓ, replace the substitution θ at the beginning of ℓ with θ composed with σ, and apply σ to the rest of ℓ.*
3. *If e is an inequality, then*
   - *i. if e has the form $v > t$ where v is a variable which occurs in term t, delete line ℓ; if the table is now empty,* stop*; the answer is* false*;*
   - *ii. if e has the form $t > v$ where v is a variable which occurs in t, delete e from ℓ; if ℓ is now empty,* stop*; the answer is* true*;*
   - *iii. otherwise, the lpo definition will apply to e; delete line ℓ; add to the end of the table several lines corresponding to the results of applying the lpo definition to e; each of these lines will be identical to ℓ except that the position formerly occupied by e will be replaced by one possible way in which e might be true.*

   Now do the following.
4. *Delete each line which is inconsistent.*
5. *For each line ℓ, if ℓ implies another line, delete ℓ.*
6. *Delete the redundant elements in each line. An element e is redundant in ℓ if $\ell - \{e\} \Rightarrow e$.*

Steps 4 to 6 may be interleaved with 1 to 3 in order to keep the table as simple as possible. Steps 4–6 will be automated in the next section.

Example 2 above was performed using essentially these steps. The answers to three other examples are now presented. The reader may go through the above steps to verify these.

Example 3. *Solve* $x+(y\cdot z) > x\cdot(y+z)$ *given that* $+ > \cdot$ *in the function symbol ordering.* The solution is

$$x > y$$
$$\{x \leftarrow y\}.$$

Example 4. *Solve* $x\cdot(y\cdot z) > y\cdot(z\cdot x)$. The solution is

$$x > y,\ x > z$$
$$\{x \leftarrow z\},\ z > y$$
$$\{x \leftarrow y\},\ y > z.$$

Example 5. *Solve* $(y+z)+(y+w) > x+(z+w)$. The solution is

$$y+z > x$$
$$\{x \leftarrow y+z\},\ y > z.$$

Reasoning with Components

In the previous section we have seen that every term inequality can be solved and the solution has the form $C_1 \vee C_2 \vee \cdots \vee C_n$ where each C_i is a component. In order to complete the automation of the solution process we need to have algorithms for determining when a component is inconsistent, when one component implies another, and when an element in a component is redundant.

The purpose of this section is to describe these algorithms.

Note first that even without steps 4–6, the algorithm of the previous section would generate a solution of the given inequality. The purpose of steps 4–6 is to render the solution in a simple form. This simplification is a practical necessity because without it solutions to even simple inequalities would be unwieldy. However, validity is not compromised if we do not find every possible way in which steps 4–6 can be performed. Thus, we do not attempt to find *every* possible way in which simplification can be effected by steps 4–6. We will find the common ones, however.

We first show how to decide if a conjunction of fully reduced inequalities implies another fully reduced inequality. Let $e_1, e_2, \cdots, e_n$ and e be fully reduced inequalities. Let $e_i = (\ell_i > r_i)$ for each i and let $e = (\ell > r)$. To decide whether or not

$$e_1, e_2, \cdots, e_n \Rightarrow e$$

we consider all possible inequality sequences of the form

$$\ell \geq \ell_1 > r_1 \geq \ell_2 > r_2 \geq \cdots \geq \ell_k > r_k \geq r \quad (1)$$

where each $\ell_j > r_j$ is one of the e_i. In the above display, each $\geq$ is shown by the subterm property; each $>$ is present in the e_i. If we can find such a sequence, then we have proved e; otherwise we assume a proof is not possible. For example, suppose we wish to show that $\{x > y,\ y > z\} \Rightarrow x > z$. We have

$$x = x > y = y > z = z$$

as an expression of form (1). Thus it is proved.

Now suppose C and D are components and it is required to determine whether $C \Rightarrow D$. Let

$$C = \{\theta, e_1, \cdots, e_n\} \text{ and } D = \{\phi, f_1, \cdots, f_m\}$$

where $\phi = \{v_1 \leftarrow t_1, \cdots, v_n \leftarrow t_n\}$. Then $C \Rightarrow D$ is equivalent to

$$e_1, \cdots, e_n \Rightarrow \phi\theta, f_1\theta, \cdots, f_k\theta \quad (2)$$

where $\phi\theta = \{v_1\theta = t_1\theta, \cdots, v_n\theta = t_n\theta\}$ because substitution of equals for equals is logically valid, and after θ is used in this way it can no longer contribute to the required implication. Normally inequalities cannot prove equalities, so if the implication is to be true, we must have $\phi\theta$ true. That is, we must have $v_1\theta \equiv t_1\theta, \cdots, v_n\theta \equiv t_n\theta$. We are left with the problem of determining the validity of

$$e_1, \cdots, e_n \Rightarrow f_1\theta, \cdots, f_m\theta.$$

To do this, consider the proof of each $f_i\theta$ separately and conjoin the results. Thus, we must be able to solve the problem

$$e_1, \cdots, e_n \Rightarrow f\theta$$

where f is fully reduced. If $f\theta$ is fully reduced, use the procedure of the previous paragraph. Otherwise, solve the inequality $f\theta$ using the procedure of Section 4 and obtain $C_1 \vee \cdots \vee C_k$, where each C_i is a component. We will determine the validity of $e_1, \cdots, e_n \Rightarrow C_i$ separately for each i and disjoin the results. For $e_1, \cdots, e_n$ to imply the component $\{\phi, e'_1, \cdots, e'_p\}$, ϕ must be empty and $e_1, \cdots, e_n$ must imply each e'_i separately as detailed in the previous paragraph.

For example, in Section 4 it was required to show that $\{x > y\cdot a\} \Rightarrow \{x > y,\ x > a\}$. To do this, we show

$x > y \cdot a \Rightarrow x > y$ and $x > y \cdot a \Rightarrow x > a$ separately. The first follows from

$$x = x > y \cdot a > y$$

and the second from

$$x = x > y \cdot a > a,$$

each of which has the form (1).

In order to show that a component $\{\theta, e_1, \cdots, e_n\}$ is inconsistent, we let i run from 1 to n, $e_i = (\ell_i > r_i)$ and test

$$e_1, \cdots, e_{i-1}, e_{i+1}, \cdots, e_n \Rightarrow r_i > \ell_i.$$

If this is so for some i, then the component is inconsistent. Otherwise it is presumed consistent. For example, suppose $\{x > y,\ y > x\}$ is the component. Try $x > y \Rightarrow x > y$. This is easily proved by the above method. Thus this component is inconsistent.

Similarly, in order to test an element e of a component C for redundancy, check $C - \{e\} \Rightarrow e$.

References

Comon, H. 1990. Solving Inequations in Term Algebras. In Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science.

Dershowitz, N. 1987. Termination of rewriting. *Journal of Symbolic Computation* 3:69-116.

Kirchner, C., and Kirchner, H. 1989. Constrained Equational Reasoning, Technical Report, CRIN 89-R-220, Centre de Recherche en Informatique de Nancy.

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

Monthly 1968. *The American Mathematical Monthly* 75.

Peterson, G. 1990. Complete Sets of Reductions with Constraints. In Proceedings of the Tenth International Conference on Automated Deduction.

Sacks, E. 1987. Hierarchical Reasoning about Inequalities. In Proceedings of the Sixth National Conference on Artificial Intelligence, 649-654. American Association for Artificial Intelligence.