Acta Informatica 28, 95–119 (1990)

Automating the Knuth Bendix ordering[*]

Jeremy Dick[1,], John Kalmus[1], and Ursula Martin[2,***]**
[1] Informatics Department, Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, UK
[2] Department of Computer Science, Royal Holloway and Bedford New College, University of London, Egham Hill, Egham, Surrey TW20 0EX, UK

Received March 2, 1989/Mai 29, 1990

Abstract. Knuth and Bendix proposed a very versatile technique for ordering terms, based upon assigning weights to operators and then to terms by adding up the weights of the operators they contain. Our purpose in this paper is as follows. First we give some examples to indicate the flexibility of the method. Then we give a simple and practical algorithm, based on solving systems of linear inequalities, for determining whether or not a set of rules can be ordered by a Knuth Bendix ordering. We also describe how this algorithm may be incorporated in a completion procedure which thus considers all possible choices of weights which orient a given equation.

1. Introduction

Well-founded orderings on terms arise in several areas of computer science: proving termination [De]; devising implementation strategies for functional languages [O'D 77]; and doing inductive proofs with an automated theorem prover [BM 79]. This paper is particularly concerned with proving the termination of term rewriting systems. We are given a set of rules, or pairs of terms, and we want to prove termination by constructing a well-founded ordering in which the first term of each pair is greater than the second.

In 1952 Higman [Hi 52] showed that any ordering on terms which is what is now called a simplification ordering is well-founded (see also [De 82].) Many simplification orderings have been described in print, see [HO 80] or [De] for

---

* A preliminary version of this paper [Ma 87] appeared in the Proceedings of RTA-87
** Supported by the Science and Engineering Research Council
*** Supported in part by the Science and Engineering Research Council under grant GR/D 81718

Offprint requests to: J. Dick

a survey. A term rewriting system such as REVE [Le 83] or ERIL [Di 85] and [DK 88] will typically implement a family of simplification orderings and the user or the machine sets certain parameters (for example a partial ordering on the operators) which, with luck, will pick out one ordering which proves termination.

Termination of a set of rules is undecidable in general [HL 78] so the best we can hope for is to have several simplification orderings available which between them cope with many of the kinds of rule that occur in practice. Thus no one family of simplification orderings is necessarily 'better' than another. Each may order some rules that others cannot, although some may be easier to implement, faster to execute or simpler to write down than others. As Huet and Oppen point out [HO 80, Sect. 11], "Generating a canonical term rewriting system is similar to compilation; you run the algorithm only once for the theory in which you are interested, even if it is very costly." One reason for this cost is the difficulty of proving termination.

In their original implementation of the critical pair completion algorithm Knuth and Bendix [KB 70] used a simple but powerful ordering. Non-negative integer weights are assigned to each function symbol, and then to each term by adding up the weights of the function symbols it contains. Two terms are compared by comparing their weights and, if the weights are the same, by comparing their subterms in the same way. A precise definition is given in the next section. Several variations of the ordering are possible, for example to deal with AC operators, but we do not consider them here.

This ordering was able to deal with most of the examples described by Knuth and Bendix in their paper. We give some more examples in Sects. 3 and 4. It is easy to implement, and fast to execute. It is very flexible; the choice of weights means that there may be infinitely many Knuth Bendix orderings on a given set of terms. However, there are two main problems.

The first is that the number of occurrences of a variable must not increase when passing from a larger to a smaller term. Thus no Knuth Bendix ordering can order the distributive law

$$x * (y+z) \rightarrow x * y + x * z,$$

since x occurs twice on the right and once on the left.

The second problem is that it is very difficult to know what weights to choose! If the Knuth Bendix ordering fails to order our rules with one choice of weights we can try another, but without some help we may fail again and indeed it may be that no Knuth Bendix ordering can order them. This paper sets out to solve this problem. We give in Sect. 5 a simple and practical decision procedure, based on solving systems of linear inequalities, for determining whether or not a set of rules can be ordered by a Knuth Bendix ordering[1]. Essentially, although there are some subtleties involved, the rules can be ordered if and only if the solution space of an appropriate system is non-empty. We do not, in this paper, explore variations of the Knuth Bendix ordering, although our method would also apply in such cases.

[1] The possibility of using such a decision procedure was suggested by Dershowitz in [De 82], although no explicit algorithm is given there

The idea of ordering terms by associating to each a polynomial rather than a weight appears in [MN 70], [L] and [H]. It is also used in the Boyer Moore theorem prover [BM 79]. It is more flexible than the Knuth Bendix ordering, and can prove the termination of the distributive law, for example. However, choosing a polynomial is much harder than choosing a weight, and proving that two terms are ordered is no longer straightforward, since it involves verifying an inequality between two polynomials. Some progress in automating this is described in [CL].

Many recursively defined orderings, based on recursive definitions for ordinal notations in [Fe 68], have been constructed. See [De 82], [De 85] and [Ru 85] for example. These orderings depend on an operator precedence; roughly speaking one term is bigger than another if the outermost operator of the first is greater in the precedence than that of the second, and if the first is bigger than all the subterms of the second. Unlike the polynomial orderings these orderings are very suitable for machine implementation; for example REVE 2.4 contains three different ones. However in contrast to the Knuth Bendix orderings there are only $n!$ recursive path ordering on terms in n function symbols.

These recursive orderings satisfy an incremental property; that is if the operator precedence is extended then the ordering it defines on terms is extended as well. Thus in an implementation we can build up the operator precedence step by step. Each time a new rule is considered we can check if the operator precedence can be extended to order it, and if it can we extend it and hence the previous ordering on terms. A procedure for proving termination by constructing extensions in this way automatically was described in [DF 85].

The algorithm that we shall describe below has been adapted to treat the Knuth Bendix ordering in a similar way. At each stage (roughly speaking) we have a set of rules, and a system of inequalities which have a solution, proving termination of the rules. When a new rule is to be considered, the system of inequalities is extended appropriately. If the extended system has a solution, then there exists a KBO for the set of rules which includes the new rule. We shall describe below how this has been implemented in the ERIL system [DK 88].

The paper is organised as follows. Sections 2, 3 and 4 contain a description of the Knuth Bendix ordering and examples. Sections 5 and 6 give the algorithm and an example, and Sect. 7 a description of a method for solving linear inequalities. In Sect. 8 we describe how our technique can be incorporated in a completion algorithm.

2. The Knuth Bendix ordering

In this section we define the Knuth Bendix ordering.

We assume we have a finite set of operators Σ, each of fixed arity, and that there is at least one operator of zero arity (i.e. a constant)[2]. The arity of the element f of Σ is denoted by $\alpha(f)$. The elements of the term algebra $\mathcal{T}(\Sigma, X)$ are built up from Σ and a set of variables X in the usual way. If

[2] This is so that pure words (terms involving operators but no variables) may be considered, see [KB 70, p. 265]

a is an operator or a variable and t is a term we write $n(a, t)$ for the number of occurrences of a in t.

A partial ordering on terms is an irreflexive transitive binary relation. If $>$ is a partial ordering, we use the symbol $\geqq$ to denote $a > b$ or $a = b$.

Now we choose a partial ordering $\gg$ on Σ, and we assign a non-negative integer, or weight, $w(f)$, to each operator $f \in \Sigma$, and a positive weight w to each variable, subject to

A If $\alpha(f) = 0$ then $w(f) \geqq w$

B If $\alpha(f) = 1$ and $w(f) = 0$ *then* $f \gg g$ for all operators g.

For each such choice of weights and partial ordering we can define an ordering on terms, known as the Knuth Bendix ordering (KBO), which we denote by $>(\gg, w)$, or just $>$. To do this we assign a weight to each term as follows. For any term t we let

$$w(t) = w \sum_{x \in X} n(x, t) + \sum_{g \in \Sigma} n(g, t)\, w(g).$$

Thus, for example, the weight of any variable is w and the weight of a term is just the sum of the weights of all the symbols appearing in it.

Now if s and t are two terms, we shall say $s > t$ if and only if

$n(x, s) \geqq n(x, t)$ for all variables $x \in X$
and either
1 $w(s) > w(t)$,
or
2 $w(s) = w(t)$
and either
(a) $s = f^n(x)$ and $t = x$ for some $n \geqq 1$,
or
(b) $s = f(s_1, \ldots, s_{\alpha(f)})$ and $t = g(t_1, \ldots, t_{\alpha(g)})$ and $f \gg g$,
or
(c) $s = f(s_1, \ldots, s_{\alpha(f)})$ and $t = f(t_1, \ldots, t_{\alpha(f)})$ and $s_1 = t_1, \ldots, s_{k-1} = t_{k-1}$ and $s_k > t_k$ for some k with $1 \leqq k \leqq \alpha(f)$.

We have

Theorem 1. *Let $>$ be defined as above. Then $>$ is a simplification ordering.*

Proof. See Apendix A.

Thus any rewriting system $R = \{L_i \rightarrow R_i \mid 1 \leqq i \leqq n\}$ over T with $L_i > R_i$ for each i is terminating [HO 80].

The intuition behind the Knuth Bendix ordering is simple, although the final definition looks rather complicated. Terms are compared by comparing lexicographically first the weights, then the outermost operators and then the subterms. The complications arise because we allow function symbols of weight zero. If we restrict to positive weights conditions **B** and **2a** are unnecessary.

Our definition is slightly different from that given in Knuth and Bendix's paper – they require that the ordering $\geqslant$ be total, and that $n(x,s)=n(x,t)$ at **2**. A lexicographic formulation of a simplified version of the Knuth Bendix ordering is given in [De 82].

3. Examples

Here are some examples.

Example 1

The following rewriting system, a complete set for the theory of free groups, appears in [KB 70].

```
R1   x * e => x
R2   e * x => x
R3   i (x) * x => e
R4   x * i (x) => e
R5   (x * y) * z => x * (y * z)
R6   i (e) => e
R7   i (x) * (x * y) => y
R8   x * (i(x) * y) => y
R9   i (i (x)) => x
R10  i (y * x) => i (x) * i (y)
```

Thus we have $\Sigma=\{*, i, e\}$, and $\alpha(*)=2, \alpha(i)=1$ and $\alpha(e)=0$. Now let the ordering on the operators be $i \geqslant * \geqslant e$, and let $w(*)=w(i)=0$ and $w(e)=1$. Thus $w=1$. These weights and orderings satisfy conditions A and B. Rules 1, 2, 3, 4, 7 and 8 can be ordered by comparing the weights. Both sides of each of the other rules have the same weight. Rule 9 is ordered by condition 2a, rules 6 and 10 by condition 2b and rule 5 by condition 2c. Thus this set of rules is terminating.

Example 2

The following example is given on page 201 of [De85]

```
--x => x
-(x + y) => (---x) * (---y)
-(x * y) => (---x) + (---y)
```

We have $\Sigma=\{-, +, *, e\}$, where $\alpha(+)=\alpha(*)=2, \alpha(-)=1$. **e** is the nullary (constant) operator required by the theory, hence $\alpha(e)=0$. Let $w(+)=w(*)=w(e)=1$, and let $w(-)=0$. So once again $w=1$. Let the ordering on the operators be $- \geqslant + \geqslant *$. These satisfy conditions A and B.

The weight of the left hand side of each rule is the same as that of the right hand side. The first rule is ordered by condition 2a, and the second and third by condition 2b. Thus the system terminates.

Example 3

The following example was considered by Bellegarde [Be] and BenCherifa and Lescanne [BL]. It arose in the study of transformations of functional programs.

```
f (x) * f (y) => f (x * y)
(x * y) * z => x * (y * z)
f (x) * (f (y) * z) => f (x * y) * z
```

This cannot be ordered by the recursive path ordering; Lescanne and Cherifa proved termination with a polynomial ordering. It is easy to see that it can be ordered in any Knuth Bendix ordering in which $w(f) > 0$.

Example 4

The following example comes from a paper of Lescanne [Le 86].

```
(x * y) + (x * z) => x * (y + z)
(x + y) + z => x + (y + z)
(x * y) + ((x * z) + u) => (x * (y + z)) + u
```

The recursive path ordering orders the third rule in the reverse direction, and then the Knuth Bendix algorithm runs for ever when we try to complete the resulting system. Lescanne shows termination using a polynomial ordering. It is clear that the system terminates in any Knuth Bendix ordering (since $w(x) > 0$ by assumption).

4. An extended example

In this section we describe how the Knuth Bendix ordering orders terms in two unary function symbols a and b and one nullary operator e, for different choices of weight and operator precedence. Since there are no variables involved, we are considering ground terms (pure words).

In the examples below we list some of the small terms in order. To save space we omit brackets and e so that ab^2 stands for $a(b(b(e)))$, and so on.

Case 1 – a and b have positive weight

It is easy to show that if a and b both have positive weight then there are only finitely many terms of any given weight n, that is precisely those terms u for which

$$w(a)\,n(a,u) + w(b)\,n(b,u) = n.$$

Terms of the same weight are ordered lexicographically from the left. The ordering depends only on the ratio of $w(a)$ to $w(b)$, and the operator precedence $a >_1 b$ or $b >_2 a$, and not on the values of $w(a)$ and $w(b)$ themselves. Thus for each positive real r there are two Knuth Bendix orderings, namely $>(>_1, w)$ and $>(>_2, w)$ where $w(a) = r, w(b) = 1$, and all these orderings are distinct. If r is irrational, two terms have the same weight if and only if they contain the same number of a's and b's.

Example 1: $\mathbf{a \gg b, w(a) = w(b)}$

Terms are ordered by weight, and then lexicographically.

$$b < a < b^2 < ba < ab < a^2 < b^3 < b^2a < bab < ba^2 < ab^2 < aba < a^2b < a^3 < \ldots$$

Example 2: $\mathbf{b \gg a, w(a) = w(b)}$

The weight of a term is the same as in the previous example, but the order on terms of the same weight has been reversed.

$$a < b < a^2 < ab < ba < b^2 < a^3 < a^2b < aba < ab^2 < ba^2 <$$
$$< bab < b^2a < b^3 \ldots$$

Example 3: $\mathbf{a \gg b, w(a) = 2, w(b) = 1}$

$$b < b^2 < a < b^3 < ba < ab < b^4 < b^2a < bab < ab^2 < a^2$$
$$< b^5 < b^3a < b^2ab < bab^2 < ba^2 < ab^3 < aba < a^2b < b^6 < \ldots$$

Example 4: $\mathbf{b \gg a, w(a) = 2, w(b) = 1}$.

This is the same as the previous example, except that the order on terms of the same weight has been reversed.

$$b < a < b^2 < ab < ba < b^3 < a^2 < ab^2 < bab < b^2a < b^4 < \ldots$$

Example 5: $\mathbf{a \gg b, w(a) = 3, w(b) = 2}$.

$$b < a < b^2 < ba < ab < b^3 < a^2 < b^2a < bab < ab^2 < b^4 < \ldots$$
$$ba^2 < aba < a^2b < \ldots$$

Case 2 – a has zero weight

Now suppose that $w(a)=0$, in which case $w(b)>0$ and $a \gg b$. There are now infinitely many terms of given weight. It is easy to see that $u>v$ if and only if

either $n(b,u)>n(b,v)$, so $w(u)>w(v)$

or $n(b,u)=n(b,v)$, and hence $w(u)=w(v)$, but u is greater than v in a lexicographic ordering from the left, that is,

if $u=f_1 f_2 \ldots f_n$ and $v=g_1 g_2 \ldots g_m$, then

either $n>m$ and $f_1=g_1, \ldots, f_m=g_m$.

or there is an i with $1 \leqq i \leqq min(m,n)$ and $f_1=g_1, \ldots, f_{i-1}=g_{i-1}$, where $f_i=a$ and $g_i=b$

Thus we have

$$\begin{aligned} & a<a^2<a^3 \ldots <a^i< \ldots <b<ba<ba^2< \ldots <ba^i< \ldots \\ & <ab<aba<aba^2< \ldots <aba^i< \ldots <b^2<b^2a< \ldots <b^2a^i \\ & < \ldots <bab<bata< \ldots <baba^i< \ldots \end{aligned}$$

5. The decision procedure

The Knuth Bendix ordering operates by assigning weights to symbols and placing a precedence on operators. As mentioned in the Introduction, a major difficulty in any practical implementation involving a set of rewrite rules lies in finding the right combination of weights and precedences for ordering those particular rules. In this section we describe an algorithm which avoids this difficulty by simply determining whether or not such a combination of weights and precedences exists, rather than having to select particular weights.

Our procedure works by maintaining a system of inequalities, which the weights must satisfy, and an operator precedence. A convenient method for finding the general solution of such a system of inequalities, the Method of Complete Description, will be given in Section 7 below. A subsystem of the inequalities, called the degenerate subsystem, is used to determine when the operator precedence should be extended.

Thus we begin by describing linear inequalities.

Linear inequalities

Let T be a system of m linear inequalities in n variables $x_1, \ldots, x_n$ over $\mathbf{R}$, the real numbers, of the form

$$a_{i1}x_1 + \ldots + a_{in}x_n \geqq 0, \qquad 1 \leqq i \leqq m.$$

If $\mathbf{x}$ is a vector we write $\mathbf{x} \geqq 0$ if each entry of $\mathbf{x}$ is non-negative, and $\mathbf{x}>0$ if each entry of $\mathbf{x}$ is positive. Thus $\mathbf{x}$ is a solution of T if and only if

$$A\mathbf{x} \geqq 0,$$

where A is the matrix (a_{ij}) with m rows and n columns.

Gale's Theorem. *Let C be the set of all solutions of T. Then C is a finite cone [Ga 60, Theorem 2.13], that is*

1. *If* $\mathbf{x}$ *and* $\mathbf{y}$ *are in* C *then* $\mathbf{x}+\mathbf{y}$ *is in* C.
2. *If* $\mathbf{x}$ *is in* C *and* $r \geqq 0$ *then* $r\mathbf{x}$ *is in* C.
3. *There are vectors* $\mathbf{u}_1, \ldots, \mathbf{u}_k$ *such that*

$$C = \{r_1 \mathbf{u}_1 + \ldots + r_k \mathbf{u}_k \mid r_i \in \mathbf{R}, r_i \geqq 0, 1 \leqq i \leqq k\}.$$

Note the special case $C = \{0\}$, when 0 is the only solution.

The vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$, are called the extreme vectors of the finite cone C. Our method depends on being able to find these extreme vectors $\mathbf{u}_i$ (see Sect. 7). However, we shall also need to know some general properties of the solution of T, and these are given in the next lemma.

Lemma 1. *Let T be a system of linear inequalities as above. Let C be the set of all solutions of T. Then there is a subsystem I_T of T such that*

1. *If*

$$b_1 x_1 + \ldots + b_n x_n \geqq 0$$

is in I_T *then*

$$b_1 y_1 + \ldots + b_n y_n = 0$$

for all $(y_1, \ldots, y_n) \in C$.

2. *If* $C = \{(0, \ldots, 0)\}$, *then* $I_T = T$.
3. *If* $I_T \neq T$ *then there is an element* $(z_1, \ldots, z_n)$ *of* C *such that*

$$d_1 z_1 + \ldots + d_n z_n > 0$$

whenever

$$d_1 x_1 + \ldots + d_n x_n \geqq 0$$

is in $T \setminus I_T$.

If $S \subset T$ *then* $I_S \subseteq I_T$.

The subsystem I_T will be called the degenerate subsystem of T.

Proof. See Appendix B.

Partial orderings

Let $>$ be a partial ordering on a finite set S and H a set of pairs of elements of S. If there is a unique minimal partial ordering $>_*$ on S with $> \subseteq >_*$, and $s >_* t$ for all pairs $(s, t) \in H$, we denote it by *extend*$(>, H)$; if not we set *extend*$(>, H) = \bot$. Then *extend*$(>, H) \neq \bot$ if and only if there is some ordering $\gg$ on S with $f \gg g$ for all (f, g) satisfying $f > g$ or $(f, g) \in H$. If $H = \{(s, t)\}$ and $s \not\gtreqless t$ then *extend*$(>, H)$ exists if and only if $s \not< t$. We can determine if *extend*$(>, H)$ exists by starting with $\bot$ and extending $>$ by one element of H at a time. Notice that *extend*(*extend*$(>, A), B) =$ *extend*$(>, A \cup B)$ if none of the expressions involved is $\bot$.

Reduced pairs

If s is not a variable and $s=f(s_1, \ldots, s_{\alpha(f)})$ we define $hd(s)$ to be f. If s and t are two non-variable terms with $hd(s)=hd(t)$ we want to replace them by two smaller terms, the reduced pair $(red(s), red(t))$, defined as follows. If $s=f(s_1, \ldots, s_{\alpha(f)})$, $t=f(t_1, \ldots, t_{\alpha(f)})$, $s \neq t$, and $\alpha(f) \geqq 1$ let $(red(s), red(t))=(s_i, t_i)$ where $1 \leqq i \leqq \alpha(f), s_1=t_1, \ldots, s_{i-1}=t_{i-1}$ and $s_i \neq t_i$. Otherwise, let $(red(s), red(t))=(s, t)$.

The automated Knuth Bendix ordering

In this section we present an algorithm to determine whether or not a finite set of rewrite rules $R=\{L_i \rightarrow R_i \mid i \in I\}$ can be ordered by a Knuth Bendix ordering.

The data. At any stage in the algorithm we have

Q a set of pairs of terms, consisting of $Q_0=\{(L_i, R_i) \mid L_i \rightarrow R_i\}$ together with $(red(s), red(t))$ for certain (s, t) in Q, and
$>$ a partial ordering on the function symbols.

Associated to these we have

E(Q) a system of linear inequalities in the variables $w(f)$, for each function symbol $f \in \Sigma$, and the variable w, derived as follows:
$\mathbf{e_{st}}$ each pair (s, t) in Q contributes

$$\sum_{f \in \Sigma} w(f)[n(f, s)-n(f, t)]+w \sum_{x \in X}[n(x, s)-n(x, t)] \geqq 0$$

$\mathbf{e_f}$ each non-nullary function symbol f contributes

$$w(f) \geqq 0$$

$\mathbf{e_a}$ each nullary function symbol a contributes

$$w(a)-w \geqq 0$$

$\mathbf{e_w}$ the variable w is non-negative[3]

$$w \geqq 0,$$

I(Q) the degenerate subsystem $I(Q)=I_{E(Q)}$ of $E(Q)$ defined in Lemma 1, obtained using the Method of Complete Description or otherwise, and

H(Q) $\{(hd(s), hd(t)) \mid e_{st} \in I(Q), s, t \notin X, hd(s) \neq hd(t)\}$, a subset of $\Sigma \times \Sigma$.

The algorithm. The outline of the algorithm is rather simple; at each pass we check if the inequalities have a solution which gives a Knuth Bendix ordering, and add new inequalities to cope with $(red(s), red(t))$ for any pairs (s, t) which have $w(s)=w(t)$ under all solutions to the current set. The key to the whole

[3] After determining the finite cone of solutions (see Sect. 7), we reject any solution having $w=0$ – thereby fulfilling the requirement that w must be strictly positive in any KBO. See fail **2** in the algorithm below

process is Lemma 1, part 3, which ensures that these are the only new pairs (s, t) that we have to consider.

```
$Q := Q_0$ and $> := \{\}$
while $(Q \neq Q \cup \{(red(s), red(t)) \mid e_{st} \in I(Q)\})$ do
begin
    $Q := Q \cup \{(red(s), red(t)) \mid e_{st} \in I(Q)\}$
    if $(e_{st} \in E(Q)$ and $\exists x \in X : n(x, s) < n(x, t))$ then $fail$ **1**;
    if $e_w \in I(Q)$ then $fail$ **2**;
    if $(e_f, e_g \in I(Q)$ for unary $f$ and $g, f \neq g)$ then $fail$ **3**;
    if $(e_f \in I(Q)$ for $f$ unary) then
        if $(extend(>, \{(f, h) \mid \neq h, h \in \Sigma\}) = \perp)$ then $fail$ **4**
        else $(> =: extend(>, \{(f, h) \mid f \neq h, h \in \Sigma\}))$;
    if $(\exists e_{st} \in I(Q) : s \in X)$ then $fail$ **5**;
    if $(extend(>, H(Q)) = \perp)$ then $fail$ **6**
    else $(> := extend(>, H(Q)))$;

end
```

We have the following theorem,

Theorem 2. *The algorithm always terminates. R can be ordered by a Knuth Bendix ordering if and only if it does not fail.*

Proof. See Appendix C.

6. Example

In this section we shall describe what the algorithm does on a simple example. The following equations R are part of example 1 in Sect. 3.

$$R = \left\{ \begin{matrix} (x * y) * z \to x * (y * z) \\ i(y * x) \to i(x) * i(y) \end{matrix} \right\}.$$

Thus we have

$$Q_0 = \left\{ \begin{matrix} ((x * y) * z, x * (y * z)) \\ (i(y * x), i(x) * i(y)) \end{matrix} \right\}.$$

Let

$$s = (x * y) * z$$
$$t = x * (y * z)$$
$$u = i(y * x)$$
$$v = i(x) * i(y).$$

Then we have

$$E(Q_0) = \left\{ \begin{array}{rrcl} e_{st}: & 0 & \geqq & 0 \\ e_{uw}: & -w(i) & \geqq & 0 \\ e_*: & w(*) & \geqq & 0 \\ e_i: & w(i) & \geqq & 0 \\ e_w: & w & \geqq & 0 \end{array} \right\}.$$

It is easy to see in this case that

$$I(Q_0)=\left\{\begin{array}{lrcl} e_{st}: & 0 & \geqq & 0 \\ e_{uv}: & -w(i) & \geqq & 0 \\ e_i: & w(i) & \geqq & 0 \end{array}\right\},$$

and

$$H(Q_0)=\{(i,*)\}.$$

At the beginning of the first pass we get

$$Q_1=\left\{\begin{array}{cc} ((x*y)*z, & x*(y*z)) \\ (i(y*x), & i(x)*i(y)) \\ (x*y, & x) \end{array}\right\},$$

and so

$$E(Q_1)=E(Q_0)\cup\{w+w(*)\geqq 0\},$$

and

$$I(Q_1)=I(Q_0).$$

This means that $Q_1=Q_2$, and so the algorithm terminates at the end of the first pass, and R can be ordered, provided it does not fail during the first pass. It is easy to see that it does not fail at 1, 2, 3 or 5. Now *extend*$(\{\ \},(i,*))$ exists; it is just the ordering $\gg$ with $i \gg *$. So it does not fail at 4. Also *extend* $(\gg, H(Q_1))=\gg$, so the algorithm does not fail at 6. Thus R can be ordered.

7. The method of complete description

In this section we describe a method which is capable of solving all linear programming problems which have a solution[4]. It is based on a theorem concerning the extreme points of convex polyhedral cones (see [Uz 58]), and will allow us to calculate all the optimal solutions of a system T of linear inequalities[5]. We have chosen it in preference to the Simplex Method (which is described in detail in [Ch 83], Chapters 16 and 18) or the Dual Simplex Method, because we know of no way in which such methods can be simply employed to yield the general solution of T, rather than just its optimal solution.

The Method of Complete Description (MCD) is a method of computing the solutions, and is ideal for our purposes because it is intrinsically incremental in nature. This means that adding a new inequality to the existing system does not necessitate a complete recalculation: rather, the new solution space is obtained from the existing one by a single calculation amounting to a simple matrix construction and two matrix multiplications. Finding $I(Q)$ is then a sim-

[4] Our approach follows closely that in Sect. 15.2, pp. 263–267 of [K 68]. Notice, however, that while Kreko deals with systems of homogeneous linear inequalities of the form $\mathbf{a}_i^T\mathbf{x}\leqq 0$, we wish to consider systems where $\mathbf{a}_i^T\mathbf{x}\geqq 0$. The theory remains unchanged, except that we must interchange the roles of the indexing sets, N_+ and N_-, described below

[5] The first method of this kind was due to Uzawa, see [Uz 58] for greater detail and proofs

ple matter of inspecting the solution space for degeneracy, achieved by performing another multiplication. Our implementation (in Prolog as part of the ERIL system [Di 85] and [DK 88]) exploits this fundamental incrementality in Knuth Bendix completion, as it determines which way to order new critical pairs, thus allowing an ordering to be build up iteratively.

First we explain how a matrix may be used to represent the extreme directions of the finite cone described by a single linear inequality.

The finite cone for a single inequality

Suppose we are given a homogeneous inequality

$$\mathbf{a}^{\mathrm{T}}\mathbf{x} \geqq 0,$$

where

$$\mathbf{a}^{\mathrm{T}} = (a_1, a_2, \ldots, a_n) \text{ and } \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \geqq 0.$$

Then we define the matrix $\kappa(\mathbf{a}^{\mathrm{T}})$ in the following way.

We classify the indices $1, 2, \ldots, n$ according to the signs of the vector components $a_1, a_2, \ldots, a_n$, thus:

$$\begin{aligned} N_- &= \{i \mid a_i < 0\} = \{i_1, i_2, \ldots, i_r\}, \\ N_0 &= \{j \mid a_j = 0\} = \{j_1, j_2, \ldots, j_s\}, \\ N_+ &= \{k \mid a_k > 0\} = \{k_1, k_2, \ldots, k_t\}. \end{aligned}$$

Then we define column vectors $\mathbf{e_j}, \mathbf{e_k}, \mathbf{e_{ik}}$ by

$$\mathbf{e_j} = \begin{pmatrix} 0 \\ \cdot \\ 1 \\ \cdot \\ 0 \end{pmatrix} (j), \quad \text{for } j \in N_0,$$

$$\mathbf{e_k} = \begin{pmatrix} 0 \\ \cdot \\ 1 \\ \cdot \\ 0 \end{pmatrix} (k), \quad \text{for } k \in N_+$$

(i.e. unit vectors in the appropriate directions), and

$$\mathbf{e_{ik}} = \begin{pmatrix} 0 \\ \cdot \\ a_k \\ \cdot \\ -a_i \\ \cdot \\ 0 \end{pmatrix} = a_k\,\mathbf{e_i} - a_i\,\mathbf{e_k}, \qquad \text{for } i \in N_-, k \in N_+.$$

Then the $(n, s+t+rt)$-matrix $\kappa(\mathbf{a}^\mathrm{T})$ is defined by

$$\kappa(\mathbf{a}^\mathrm{T}) = (\mathbf{e}_{\mathbf{j}_1}, \ldots, \mathbf{e}_{\mathbf{j}_s}, \mathbf{e}_{\mathbf{k}_1}, \ldots, \mathbf{e}_{\mathbf{k}_t}, \mathbf{e}_{\mathbf{i}_1}\,\mathbf{k}_1, \ldots, \mathbf{e}_{\mathbf{i}_r\,\mathbf{k}_t}).$$

The column vectors of the matrix $\kappa(\mathbf{a}^\mathrm{T})$ are the extreme directions of the finite cone of solutions of the inequality

$$\mathbf{a}^\mathrm{T}\mathbf{x} \geqq 0, \quad \text{where } x \geqq 0.$$

Outline of the Method

We now give a brief outline of the Method of Complete Description (MCD), which we use to find the solutions of T, our system of m homogeneous linear inequalities in n unknowns. It determines the extreme vectors $\mathbf{u}_\mathbf{i}(i=1, \ldots, k)$ of the finite cone C of solutions – the general solution C being simply the set of all positive linear combinations of these vectors $\mathbf{u}_\mathbf{i}$. (See Part 3 of Gale's Theorem in Sect. 5, above.) Thus MCD achieves, for a system of linear inequalities, what calculating $\kappa(\mathbf{a}^\mathrm{T})$ did for the single inequality $\mathbf{a}^\mathrm{T}\mathbf{x} \geqq 0$ in the previous section.

Suppose we are given the system T of m homogeneous linear inequalities in n variables $x_1, \ldots, x_n$ over $\mathbf{R}$, the real numbers, of the form

$$\mathbf{a}_\mathbf{i}^\mathrm{T}\mathbf{x} \geqq 0, \quad 1 \leqq i \leqq m,$$

where

$$\mathbf{a}_\mathbf{i} = \begin{pmatrix} a_{1i} \\ a_{2i} \\ \cdot \\ \cdot \\ a_{ni} \end{pmatrix} \quad \text{and} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \geqq 0.$$

Define the subsystems $T_1, T_2, \ldots, T_m$ of T as follows.

For $1 \leqq l \leqq m$, let T_l be the subsystem of inequalities

$$\mathbf{a}_\mathbf{j}^\mathrm{T}\mathbf{x} \geqq 0, \quad 1 \leqq j \leqq l.$$

Notice that T_1 is the system involving just the first inequality $\mathbf{a}_\mathbf{1}^\mathrm{T}\mathbf{x} \geqq 0$, and $T_m \equiv T$.

Then MCD constructs the solution matrices $S_1, S_2, \ldots, S_m$ to the monotonic sequence of subsystems $T_1, T_2, \ldots, T_m$, inductively, as follows.

First, we initialise $S_0 \equiv P_0 = I_n$ (the $n \times n$ identity matrix). (This is merely for convenience – we place no interpretation on either P_0 or S_0.)

Then, inductively, for $l = 0, \ldots, m-1$, we define

$$S_{l+1} = (P_0 P_1 \ldots P_l) P_{l+1} = S_l P_{l+1}$$

and

$$\mathbf{a}'^{\mathrm{T}}_{l+1} \equiv \mathbf{a}^{\mathrm{T}}_{l+1} (P_0 P_1 \ldots P_l),$$

where

$$P_{l+1} = \kappa(\mathbf{a}'^{\mathrm{T}}_{l+1}) = \kappa(\mathbf{a}^{\mathrm{T}}_{l+1}(P_0 P_1 \ldots P_l)) = \kappa(\mathbf{a}^{\mathrm{T}}_{l+1} S_l).$$

Now (for $l > 0$) the column vectors of the matrix S_l are the extreme vectors of the finite cone of solutions of the subsystem T_l containing the first l inequalities of T.

Thus the method works incrementally. At the $(l+1)^{\text{th}}$ stage it accepts as input the vector $\mathbf{a}^{\mathrm{T}}_{l+1}$, representing the inequality $\mathbf{a}^{\mathrm{T}}_{l+1}\mathbf{x} \geqq 0$, together with the solution S_l of the previous subsystem T_l. By performing a matrix multiplication of these two inputs, it obtains the vector $\mathbf{a}'^{\mathrm{T}}_{l+1}$, representing a new, "prepared" inequality, $\mathbf{a}'^{\mathrm{T}}_{l+1}\mathbf{x} \geqq 0$. It then calculates $\kappa(\mathbf{a}'^{\mathrm{T}}_{l+1})$ for this prepared inequality, and pre-multiplies the resulting matrix by (the input solution) S_l. This yields as output the solution S_{l+1} of the new subsystem T_{l+1}.

This underlying incrementality of MCD makes it particularly useful in our algorithm. New rules may result from orienting some of the axioms in our set, or through the generation of non-deleted critical pairs. In either case, our system of linear inequalities will be augmented. There is no need, however, to recompute the entire solution space of this larger system from scratch. Rather, starting with the (known) solution to the previous system, we need only perform a few iterations of MCD, as described above, to yield the general solution of the extended system.

Example

Consider the system of two homogeneous inequalities

$$\left\{\begin{array}{l} -x_1 + x_2 \geqq 0 \\ 2x_1 - x_2 \geqq 0 \end{array}\right\}, \quad x_1, x_2 \geqq 0,$$

with

$$\begin{pmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ 2 & -1 \end{pmatrix}.$$

From the first inequality we obtain the matrix

$$P_1 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

and

$$S_1 = S_0 P_1 = I_2 P_1 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

Then from

$$\mathbf{a}'^{\mathrm{T}}_2 = \mathbf{a}^{\mathrm{T}}_2 S_1 = (2, -1) \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = (-1, 1)$$

we obtain

$$P_2 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix},$$

and finally

$$S_2 = S_1 P_2 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}.$$

The column vectors of the matrix S_2 are the extreme vectors of the finite cone of solutions of the given system of two inequalities.

8. The implementation in ERIL

Theorem 2 says that the success of our algorithm is a necessary and sufficient condition for a finite set of rewrite rules to be Knuth Bendix orderable. This has major implications for the version of KBO which is currently implemented in ERIL (see [DK 88]).

For a given set of rewrite rules (pairs of terms) Q_0, we start by solving the system of linear inequalities $E(Q_0)$ using the Method of Complete Description (see Sect. 7 above). Thus a characteristic of the implementation is that we consider all possible choices of weights which could give us a Knuth Bendix ordering for Q_0, without at first resorting to the use of function precedences. However, if the degenerate subsystem $I(Q)$ happens to contain either the inequality $\mathbf{e}_f\, w(f) \geqq 0$, where f is a unary function symbol, or else the inequality $\mathbf{e}_{st}\, w(s) - w(t) \geqq 0$, where s and t are non-variable terms with $hd(s) \neq hd(t)$, it will be necessary to extend the function precedences appropriately and perform another pass of the algorithm, using MCD again to solve the new (possibly augmented) system of inequalities $E(Q)$. We continue in this manner until the algorithm terminates. Thus the procedure, if successful, yields a KBO with a minimal ordering on operators, by considering all possible choices of weights at each stage before attempting to extend operator precedences further.

This contrasts with implementations of KBO involving the use of fixed weights and function precedences. On one hand, they may fail to prove termination simply as a result of unlucky choices of such parameters, without first ensuring that all other possible choices have been exhausted. On the other hand, termination may be successfully demonstrated, but using an ordering which is stronger than necessary, in the sense that some of the operator precedences assumed may not be required.

When performing critical pair completion, however, we are often faced with a rather different problem. Instead of being given a ready made set of rules, we are presented with an initial set of axioms (unoriented equations), from which we build our rewrite system incrementally.

The implementation in ERIL maintains a system of inequalities which is initialised to the one-by-one identity matrix I_1, corresponding to the single inequality $\mathbf{e}_w\, w \geqq 0$, requiring that the weight assigned to a variable be non-negative (see Sect. 5). The complete initial solution is also I_1.

The number of unknowns in the system is augmented each time a new function symbol f appears in the equations to be oriented. This is done by:

1. adding an extra row of zeros to the matrix representing the solution space; and
2. introducing a new inequality, $\mathbf{e}_f\, w(f) \geqq 0$, by an iteration of the MCD.

This process preserves the solution space.

After augmenting the solution space in this way, the orientation of new equation is considered by constructing two new inequalities, one for each possible orientation of the equation. Alternative iterations of the MCD are computed, and orientations for which there exists a KBO are determined. If just one orientation is possible, the corresponding computtion is selected. If both orientations are possible, the user is asked to determine the orientation (and the corresponding computation selected). If neither orientation is possible, then no KBO exists which can orient the equation.

Example 1

Pierre Lescanne showed us the following example [Le 87], which cannot be completed using the recursive path ordering. It is an axiomatisation for groups in terms of the binary operator / representing right division.

The initial equations are

```
A1   x / x = e
A2   x / e = x
A3   i(x / y) = y / x
A4   (x / y) / z = x / (z / i(y)).
```

The implementation of ERIL which uses our algorithm generates 77 critical pairs during the completion process, of which 70 are deleted. Two rules are produced (R4 and R5 below) which can be oriented in either direction, and in each case the user is prompted to make a choice. Eventually we obtain the following canonical rewrite system comprising 11 rules

```
R1   x / x ⇒ e
R2   x / e ⇒ x
R3   i(x / y) ⇒ y / x
R4   (x / y) / z ⇒ x / (z / i(y))
R5   e / x ⇒ i(x)
R6   i(i(x)) ⇒ x
R7   i(e) ⇒ e
R8   x / (y / i(x)) ⇒ i(y)
R9   x / (y / (i(x) / z)) ⇒ i(z) / y
R10  i(x) / (y / x) ⇒ i(y)
R11  i(x) / (y / (x / z)) ⇒ i(z) / y.
```

When the algorithm terminates, $E(Q)$ contains 16 inequalities in 4 unknowns (one for each function symbol and one representing the weight of variables). From the solution calculated by MCD we discover that a possible assignment of weights proving termination of the canonical system above is

Symbol	Weight
i	0
e	1
$/$	0
variable	1

and the corresponding (minimal) function precedences are

$$i > /$$
$$i > e,$$

which are forced by the choice of zero weight for the unary operator i.

Example 2

As part of the description of a vending machine, suppose we are given constants $1p$, $2p$, $5p$, $10p$ and a binary operation $+$, which satisfy the axioms

```
A1   1p + 1p = 2p
A2   1p + (2p + 2p) = 5p
A3   5p + 5p = 10p
A4   (x + y) + z = x + (y + z).
```

Very large matrices are produced during the completion process, since many rules are generated which can be ordered in either direction, and which the user is asked to orient. Eventually, we obtain the following canonical rewrite system, comprising 21 rules

```
R1   1p + 1p ⇒ 2p
R2   1p + (2p + 2p) ⇒ 5p
R3   5p + 5p ⇒ 10p
R4   (x + y) + z ⇒ x + (y + z)
R5   1p + (1p + x) ⇒ 2p + x
R6   1p + (2p + (2p + x)) ⇒ 5p + x
R7   2p + 1p ⇒ 1p + 2p
R8   2p + (1p + x) ⇒ 1p + (2p + x)
R9   2p + (2p + 2p) ⇒ 1p + 5p
R10  2p + (2p + (2p + x)) ⇒ 1p + (5p + x)
R11  5p + 1p ⇒ 1p + 5p
R12  5p + (1p + x) ⇒ 1p + (5p + x)
R13  5p + 2p ⇒ 2p + 5p
R14  5p + (2p + x) ⇒ 2p + (5p + x)
R15  5p + (5p + x) ⇒ 10p + x
R16  10p + 1p ⇒ 1p + 10p
```

```
R17  10p + (1p + x) ⇒ 1p + (10p + x)
R18  10p + 2p ⇒ 2p + 10p
R19  10p + (2p + x) ⇒ 2p + (10p + x)
R20  10p + 5p ⇒ 5p + 10p
R21  10p + (5p + x) ⇒ 5p + (10p + x).
```

A possible assignment of weights proving termination of the canonical system above is

Symbol	Weight
$1p$	1
$2p$	1
$5p$	1
$10p$	1
$+$	0
variable	1

The corresponding (minimal) function precedences are

$$10p > 5p > 2p > 1p,$$

which are necessary to orient rules 7, 11, 13, 16, 18 and 20 in the direction shown. Clearly, any such rule whose left- and right-hand sides contain identical symbols, cannot by ordered by weights alone (since both sides balance), but will require on ordering an operators.

A. Proof of Theorem 1

Theorem 1. *Let $>$ be defined as above. Then $>$ is a simplification ordering.*

We begin with a lemma.

Lemma 2. 1. *If $w(s) = w(t)$ and t is a proper subterm of s then $s = f^k(t)(k \geqq 1)$ for some unary operator f of weight zero.*

2. *If x is a variable and s is a term containing x then $s > x$, and $x \not> s$.*

Proof.

1. We have

$$0 = w(s) - w(t) = w \sum_{x \in X} (n(x, s) - n(x, t)) + \sum_{g \in \Sigma} (n(g, s) - n(g, t))\, w(g).$$

Since t is a subterm of s, we have $n(u, s) \geqq n(u, t)$ for each symbol u appearing in s. As $w(s) = w(t)$ we must have $w(u) = 0$ for each symbol with $n(u, s) > n(u, t)$, and $n(u, s) = n(u, t)$ for each symbol u with $w(u) > 0$. All variables and function symbols of arity zero have positive weight, and thus s contains no other occurrences of these, apart from those already in t. Since t is a proper subterm, this means that s must be of the form $f_1(f_2(\ldots(f_k(t))))$, for some $(k \geqq 1)$, where the f_i are all unary of weight 0. Then all these f_i must be equal as by condition **B** at most one unary operator has weight 0. So $s = f^k(t)$.

2. Since x occurs in s we have $n(x,s) \geqq n(x,x)$. Clearly $w(s) \geqq w(x) = w$. If $w(s) > w$ then $s > x$. If $w(s) = w$ then part 1 holds, and $s = f^k(x) > x$ by Case 2a. Now suppose (for contradiction) that $x > s$. This would necessitate $w(s) = w \geqq w(x)$, but since $w(x) \geqq w$, we must have equality. So again $s = f^k(t)$ by Part 1, and an inspection of the definition of the Knuth Bendix ordering ($>$) yields a contradiction.

Proof (of Theorem 1)

We need to show four things:

1. $>$ is a partial ordering on terms.
2. If s is a term and t is a proper subterm (i.e. $s \neq t$) then $s > t$.
3. If $s > t$, f is a function symbol of arity $n \geqq 1$ and $s_1, \ldots, s_n$ are terms then

$$f(s_1, \ldots, s_i, \ldots, s_n)[s_i \leftarrow s] > f(s_1, \ldots, s_i, \ldots, s_n)[s_i \leftarrow t],$$

for $1 \leqq i \leqq n$.
4. If $s > t$ then $s\sigma > t\sigma$ for all substitutions σ.

1. We need to check that for any terms s, t and u
(a) $s \not> s$
(*b*) If $s > t$ and $t > u$ then $s > u$.
For (a) we observe that $s > s$ cannot arise from any part of the definition of $>$.

For (b) we observe that if $s > t$ and $t > u$ then for all $x \in X$ we have $n(x,s) \geqq n(x,t) \geqq n(x,u)$, so $n(x,s) \geqq n(x,u)$. If $w(s) > w(t)$ or $w(t) > w(u)$ the result is clear, so we consider the case $w(s) = w(t) = w(u)$. Neither of s or t can be a variable. If u is a variable then $n(u,s) \geqq n(u,t) \geqq 1$ so $s > u$ by Lemma 2, Part 2. Otherwise, $s > t$ and $t > u$ must each arise because of 2b or 2c. If either arises from 2b the result is clear. If both arise from 2c the result is also clear as $hd(s) = hd(t) = hd(u)$, an operator of arity $n \geqq 1$, and we are comparing n-tuples lexicographically.
2. The proof is by induction on the number of symbols in s. Suppose that s, t form a minimal counterexample, so that if t' is a proper subterm of s' and s' has fewer symbols than s then $s' > t'$.
It is clear that we have $n(x,s) \geqq n(x,t)$ for each variable x. We have

$$w(s) - w(t) = w \sum_{x \in X} (n(x,s) - n(x,t)) + \sum_{g \in \Sigma} (n(g,s) - n(g,t))\, w(g).$$

Since t is a subterm of s, we have $n(u,s) \geqq n(u,t)$ for each symbol u appearing in s, and since w and ach $w(g)$ is non-negative the right hand side of the equation is nonnegative. Thus $w(s) \geqq w(t)$.

If $w(s) > w(t)$ then $s > t$ as required. If $w(s) = w(t)$ then by the lemma $s = f^k(t)$. If t is a variable then $s > t$ by condition 2a. If t is not a variable then $t = g(t_1, \ldots, t_n)$ for some function symbol g of arity n. If $f \neq g$ then, by condition **B**, $f \gg g$ and so $s > t$ by condition 2b. If $f = g$ then $t = f(t_1)$, and $s = f(f^k(t_1))$. By induction $f^k(t_1) > t_1$, and so by condition 2c we have $s > t$.
3. Let $u = f(s_1, \ldots, s_i, \ldots, s_n)[s_i \leftarrow s]$, and $v = f(s_1, \ldots, s_i, \ldots, s_n)[s_i \leftarrow t]$. It is clear that we have $n(x,u) \geqq n(x,v)$ for each variable x, since the same is true for s and t. Since $s > t$ then $w(s) \geqq w(t)$. If $w(s) > w(t)$ then $w(u) > w(v)$, so $u > v$.

If $w(s)=w(t)$ then $w(u)=w(v)$, and we see that part 2c of the definition applies, and $u>v$.

4. The proof is by induction on the number of symbols in s. Suppose that s, t form a minimal counterexample, so that if $s'>t'$ and s' has fewer symbols than s then $s'\sigma>t'\sigma$ for all substitutions σ.

   It is clear that we have $n(x, s\sigma)\geqq n(x, t\sigma)$ for each variable x, as

$$n(x, s\sigma)-n(x, t\sigma)=\sum_{y\in X} n(x, y\sigma)(n(y, s)-n(y, t))\geqq 0.$$

   We have

$$w(s\sigma)-w(t\sigma)=w(s)-w(t)+\sum_{x\in X}(n(x, s)-n(x, t))(w(x\sigma)-w).$$

   It follows from the definitions that $w(u)\geqq w$ for all terms u, since u must contain a variable or a function of arity zero. Thus the expression after the sum sign is non-negative, and so $w(s\sigma)\geqq w(t\sigma)$.

   If $w(s\sigma)>w(t\sigma)$ then $s\sigma>t\sigma$, as required. If $w(s\sigma)=w(t\sigma)$ then $w(s)=w(t)$. As $s>t$, it follows from the definition of $>$ that s is not a variable. If t is a variable, then as $s>t$ the variable t must be a subterm of s, so $t\sigma$ is a subterm of $s\sigma$, and it follows from 2 that $s\sigma>t\sigma$. Thus we may assume that $s=f(s_1, \ldots, s_n)$ and $t=g(t_1, \ldots, t_m)$, for some function symbols f and g of arities m and n, and hence $s\sigma=f(s_1\sigma, \ldots, s_n\sigma)$ and $t\sigma=g(t_1\sigma, \ldots, t_m\sigma)$. We have $s>t$, and so if $f\neq g$ then we must have $f\gg g$. Thus $s\sigma>t\sigma$. If $f=g$ then we must have $s_k>t_k$ for some $k\leqq n$, and $s_i=t_i$ for $i<k$. Thus $s_k\sigma>t_k\sigma$, and $s_i\sigma=t_i\sigma$ for $i<k$, so that $s\sigma>t\sigma$.

B. Proof of Lemma 1

Lemma 1. *Let T be a system of linear inequalities as above. Let C be the set of all feasible solutions of T. Then there is a subsystem I_T of T such that*

1. *If*
$$b_1x_1+\ldots+b_nx_n\geqq 0$$
is in I_T then
$$b_1y_1+\ldots+b_ny_n=0$$
for all $(y_1, \ldots, y_n)\in C$.
2. *If $C=\{(0, \ldots, 0)\}$ then $I_T=T$.*
3. *If $I_T\neq T$ then there is an element $(z_1, \ldots, z_n)$ of C such that*
$$d_1z_1+\ldots+d_nz_n>0$$
whenever
$$d_1x_1+\ldots+d_nx_n\geqq 0$$
is in $T\backslash I_T$.

If $S\subset T$ then $I_S\subseteq I_T$.

Proof. Let $H_i=\{x\in\mathbf{R}^n|x_i=0\}$, the ith coordinate hyperplane. The set of solutions to T, that is to $Ay\geqq 0$, forms a cone C and so does the set of vectors

$AC=\{A\mathbf{c}|\mathbf{c}\in C\}$. Let $I=\{i|AC\subseteq H_i\}$, and let I_T be the corresponding subset of T. Let $J=\{1, \ldots, n\}-I$. If $A\mathbf{x}$ and $i\in I$ then

$$a_{i1}\,x_1+\ldots a_{in}\,x_n=0.$$

It is clear that if $C=\{(0, \ldots, 0)\}$ then $I_T=T$. Now suppose that $I_T\neq T$, so that $C\neq\{(0, \ldots, 0)\}$. Suppose that for each element $(z_1, \ldots, z_n)$ of C there is a $j\notin I$, with $1\leqq j\leqq m$ and

$$a_{j1}\,z_1+\ldots a_{jn}\,z_n=0.$$

Then the cone AC lies in $\bigcup_{j\in J}H_j$. Thus by the next lemma $AC\subseteq H_j$ for some $j\in J$, which contradicts our choice of I. Thus there is an element $(z_1, \ldots, z_n)$ of C such that

$$d_1\,z_1+\ldots d_n\,z_n>0$$

whenever

$$d_1\,x_1+\ldots d_n\,x_n\geqq 0$$

as in $T\backslash I_t$. □

Lemma 3. *Let $H_i=\{x\in\mathbf{R}^n|x_i=0\}$, and let C be a cone in $\mathbf{R}^n$. If I is a non-empty subset of $\{1, \ldots, n\}$ and $C\subseteq\bigcup_{i\in I}H_i$ then C is contained in one of the H_i.*

Proof. Choose $c=(c_1, \ldots, c_n)$ in C so that the smallest possible number of c_i with $i\in I$ are zero. Since $C\subseteq\bigcup_{i\in I}H_i$ there is a k in I with $c_k=0$. Now suppose that $C\nsubseteq H_k$. Pick $d\in C\backslash H_k$. Since $\mathbf{R}$ is infinite there is a positive scalar r with $rc_j\neq -d_j$ for any j in I with $c_j\neq 0$. Then $rc+d\in C$. If $rc_i+d_i=0$ then, by our choice of r, $c_i=0$. Also $rc_k+d_k=d_k\neq 0$. So $rc+d$ is an element of C with fewer non-zero entries than c, which contradicts our choice of c. Thus C lies in H_k. □

C. Proof of Theorem 2

Theorem 2. *The algorithm always terminates. R can be ordered by a Knuth Bendix ordering if and only if it does not fail.*

We introduce some further notation. As above,

$$Q_0=\{(L_i, R_i)|L_i\rightarrow R_i\}.$$

Let

$$Q_{i+1}=Q_i\cup\{(red(s), red(t))|e_{st}\in I(Q_i)\}.$$

Let

$$O_i=H(Q_i)\cup\{(f, g)|f, g\in\Sigma, e_f\in I(Q_i), \alpha(f)=1\}.$$

Lemma 4. 1. *If the algorithm reaches the end of the i-th pass without failing then $Q=Q_i$, $>=extend(\{\ \}, O_i)$ is a partial ordering on Σ and $b>c$ for all $(b, c)\in O_i$.*

2. *There is an i such that $Q_i=Q_j$ for all $j>i$.*
3. *The algorithm always terminates.*

Proof

1. Clear from examining the algorithm. If $> = \perp$ then the algorithm fails. So $>$ is of the form *extend*$(\{\ \}, O_1)$ if $i=1$, or *extend*(*extend*$(\{\ \}, O_{i-1}), O_i)$ if $i>1$. But $O_{i-1} \subseteq O_i$, so the result follows.
2. We have $Q_i \subseteq Q_{i+1}$ for each i. Since each Q_i is a subset of the finite set of all pairs of subterms of the L_i and R_i we must have $Q_i = Q_{i+1}$ for some i, and so $Q_i = Q_j$ for all $j>i$.
3. Since $Q_i = Q_j$ for all $j>i$, the algorithm will terminate at the end of the ith pass if it has not already halted with failure.

We define $\bar{Q}$ to be the limiting value of the Q_i, so that $\bar{Q}$ is the first Q_i such that $Q_i = Q_{i+1}$.

Now we can prove one half of Theorem 2.

Lemma 5. *If the algorithm does not fail there is a Knuth Bendix ordering $\succ$ with $s \succ t$ for each rule $(s,t) \in Q_0$.*

Proof. Suppose that the algorithm halts without failure at in the ith pass, so that by the previous lemma $Q = Q_i = \bar{Q}$, $>$ is a partial ordering on Σ and $b>c$ for all $(b,c) \in O_i$. By Lemma 1 there is a solution $w(f) = \bar{w}(f)$ to $E(Q)$ such that an inequality in $E(Q)$ is an equality when evaluated at $\bar{w}$ if and only if it lies in $I(Q)$. I claim that $>$ and $\bar{w}$ define the Knuth Bendix ordering $\succ$, with $s \succ t$ for all s, t in Q.

First of all, since $\bar{w}$ is a solution to $E(Q)$, $\bar{w}(f) \geqq 0$ for each f, and if $\alpha(f)=0$ then $\bar{w}(f) \geqq \bar{w} > 0$, since the algorithm did not fail at (2). If $\alpha(f)=1$ and $\bar{w}(f)=0$ then $f>g$ for all $g \neq f$ as $(f,g) \in O_i$. Thus conditions **A** and **B** in the definition of the ordering are satisfied.

Now suppose $(s,t) \in Q$. The proof is by induction on the number of symbols in s. Since $e_{st} \in E(Q)$, we have $\bar{w}(s) \geqq \bar{w}(t)$. If $e_{st} \notin I(Q)$ then $\bar{w}(s) > \bar{w}(t)$, and since the algorithm did not fail at (1) $n(x,s) \geqq n(x,t)$ for all $x \in X$, and so $s \succ t$. If $e_{st} \in I(Q)$ then $\bar{w}(s) = \bar{w}(t)$, and since the algorithm did not fail at (2) $n(x,s) = n(x,t)$ for all $x \in X$.

Now s is not a variable, because of (5). If t is a variable, $t=x$ say, then as $n(x,s) = n(x,t) = 1$ and $\bar{w}(s) = \bar{w}$, s consists of a single variable together with other function symbols of weight zero. No function symbol of arity zero has weight zero and so these function symbols must be unary, as otherwise s would contain a function symbol of arity zero. Because of (3) there is only one unary function symbol of weight zero, f say, so $s = f^n(x)$ and $s \succ t$.

Thus s and t are not variables, so we consider the function symbols $hd(s)$ and $hd(t)$. If $hd(s) \neq hd(t)$ then $(hd(s), hd(t)) \in O_i$, so $hd(s) > hd(t)$ and $s \succ t$. If $hd(s) = hd(t)$ then $(red(s), red(t)) \in Q$ and $red(s)$ contains fewer symbols than s, so that by induction $red(s) \succ red(t)$, and so $s \succ t$.

To prove the converse of Lemma 4 we need some additional properties of the Q_i.

Lemma 6. *Let $\succ$ be a Knuth Bendix ordering which is given by a weight function $\bar{w}$ and a partial ordering $\gg$ on Σ.*

1. *If $s \succ t$ for all (s,t) in $Q_i, i \geqq 1$ then the $\bar{w}$ satisfy every equation in $E(Q_i)$, and if some equation in $E(Q_i)$ gives rise to a strict inequality when evaluated at $\bar{w}$ then that equation is not in $I(Q_i)$.*
2. *$s \succ t$ for all (s,t) in Q_0 if and only if $s \succ t$ for all (s,t) in $\bar{Q}$*
3. *If $s \succ t$ for all (s,t) in $Q_i, i \geqq 1$ then $f \gg g$ for all $(f,g) \in O_i$.*

Proof

1. If $s \succ t$ then $\bar{w}(s) \geqq \bar{w}(t)$, $\bar{w}(f) \geqq 0$ for each $f \in \Sigma$, $\bar{w}(f) \geqq \bar{w}$ for each f with $\alpha(f)=1$ and $\bar{w} \geqq 0$. Thus each equation in $E(Q_i)$ is satisfied. We know from Lemma 1 that if some equation in $E(Q_i)$ gives rise to a strict inequality when evaluated at $\bar{w}$ then that equation is not in $I(Q_i)$.
2. If $s \succ t$ for all (s, t) in $\bar{Q}$ then, as $Q_0 \subset \bar{Q}$, $s \succ t$ for all (s, t) in Q_0.
   Conversely, suppose that $s \succ t$ for all (s, t) in Q_0. We show by induction on j that $s \succ t$ for all (s, t) in Q_j. So suppose that $s \succ t$ for all (s, t) in Q_k, and that $(s, t) \in Q_{k+1} \setminus Q_k$, so that $(s, t) = (red(u), red(v))$ for some (u, v) in Q_k with $e_{uv} \in I(Q_k)$ and $hd(u) = hd(v)$. Since $e_{uv} \in I(Q_k)$, we have $\bar{w}(u) = \bar{w}(v)$. Now $u \succ v$ by our induction hypothesis, and as $\bar{w}(u) = \bar{w}(v)$ and $hd(u) = hd(v)$ we must have $s = red(u) \succ red(v) = t$.
3. This follows from the definition of the Knuth Bendix ordering. If $s \succ t$, $e_{s,t} \in I(Q_i)$, $s, t \notin X$, and $hd(s) \neq hd(t)$ then $\bar{w}(s) = \bar{w}(t)$ and so $hd(s) \gg hd(t)$. Since $\succ$ is a Knuth Bendix ordering we have $f \gg g$ for any $g \in \Sigma$ if $e_f \in I(Q_i)$ and $\alpha(f) = 1$.

Now we can prove the other half of the theorem.

Lemma 7. *If there is a Knuth Bendix ordering $\succ$ with $s \succ t$ for each rule $(s, t) \in Q_0$ then the algorithm does not fail.*

Proof. Let $\succ$ be given by the weight function $\bar{w}$ and a partial ordering $\gg$ on Σ, and suppose that $s \succ t$ for all $(s, t) \in Q_0$, so that by the previous lemma $s \succ t$ for all (s, t) in $\bar{Q}$. Suppose for a contradiction that the algorithm fails during the ith pass. Then $Q = Q_i$ when it fails.

We go through the possibilities for failure one by one. If $e_{st} \in E(Q_i)$ then $(s, t) \in Q_i$ so $s \succ t$ and for each variable x we have $n(x, s) \geqq n(x, t)$. Thus the algorithm cannot fail at (1). Since $\succ$ is a Knuth Bendix ordering we have $\bar{w} > 0$, so e_w is not in $I(Q_i)$ and the algorithm cannot fail at (2). Since there is at most one unary f with $\bar{w}(f) = 0$ there is at most one unary f with $e_f \in I(Q_i)$, and so the algorithm cannot fail at (3). If x is a variable we cannot have $x \succ t$ for any term t, so the algorithm cannot fail at (5).

We are left with the possibility that the algorithm fails at (4) or (6). Now by Lemma 6 part 3 we have $f \gg g$ for all $(f, g) \in O_i$. Thus the orderings of the form $extend(>, P)$ required in (4) and (6) must exist as in each case P and $>$ are suborderings of $\gg$.

Finally we have the proof of Theorem 2 – it follows from Lemma 4 part 3, Lemma 5 and Lemma 7.

References

[Be] Bellegarde, F.: Rewriting systems on FP expressions to reduce the number sequences yielded. Sci. Comput. Program. **6**, 11–34 (1986)

[BM 79] Boyer, R.S., Moore, J.S.: A computational logic. New York: Academic Press 1979

[BL] Ben Cherifa, A., Lescanne, P.: Termination of rewriting systems by polynomial interpretations and its implementation. Sci. Comput. Program. **9**, 137–160 (1987)

[Ch 83] Chvatal, V.: Linear programming. New York: Freeman 1983

[De 82] Dershowitz, N.: Orderings for term rewriting systems. Theor. Comp. Sci. **17**, 279–301 (1982)

[De] Dershowitz, N.: Termination of rewriting. J. Symbol. Comput. **3**, 69–116 (1987)

[DF 85] Forgaard, R., Detlefs, D.: A procedure for automatically proving the termination of a set of rewrite rules. Proceedings of the First International Conference on Rewriting Techniques and Applications. (Lect. Notes Comput. Sci., vol. 202) Berlin Heidelberg New York: Springer 1985

[Di 85] Dick, A.J.J.: ERIL – Equational reasoning: an interactive laboratory. Rutherford Appleton Laboratory Report RAL-86-010, March 1985

[DK 88] Dick, A.J.J., Kalmus, J.R.: ERIL (Equational reasoning: an interactive laboratory). User's manual. Version R1.6, Rutherford Appleton Laboratory Report RAL-88-055, September 1988

[Fe 68] Feferman, S.: Systems of predicative analysis II: Representation of ordinals. J. Symbol. Logic **33**, 193–220 (1968)

[Ga 68] Gale, D.: The theory of linear economic models. New York: McGraw Hill 1960

[H] Huet, G.: Confluent reductions: abstract properties and applications to term rewriting systems. J. Ass. Comp. Mach. **27**(4) 797–821 (1980), preliminary version in 18th Symposium on Foundations of Computer Science, IEEE, 1977

[HL 78] Huet, G., Lankford, D.S.: On the uniform halting problem for term rewriting systems. Rapport Laboria 283, INRIA, March 1978

[Hi 52] Higman, G.: Ordering by divisibility in abstract algebras. Proc LMS(3) **2**, 326–336 (1952)

[HO 80] Huet, G., Oppen, D.C.: Equations and rewrite rules: a survey. In: Book, R. (ed.) Formal language theory, perspectives and open problems, pp. 349–405. New York: Academic Press 1980

[KB 70] Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational problems in abstract algebra, pp. 263–297. New York: Pergamon Press 1970

[Kr 68] Kreko, B.: Linear programming. London: Pitman 1968

[L] Lankford, D.S.: Canonical algebraic simplification in computational logic. Mem ATP-25. Automatic Theorem Proving Project. University of Texas, Austin TX, May 1975

[Le 83] Lescanne, P.: Computer experiments with the REVE term rewriting system generator, Proc 10th ACM POPL symposium, Austin, TX 1983, pp. 99–108

[Le 86] Lescanne, P.: Divergence of the Knuth Bendix completion procedure and termination orderings. Bull. Eur. Assoc. Theoret. Comput. Sci. **30**, 80–83 (1986)

[Le 87] Lescanne, P.: Private communication

[MN 70] Manna, Z., Ness, S.: On the termination of Markov algorithms. Proceedings of the Third Hawaii International Conference on System Science, Honolulu, HI, January 1970, pp. 789–792

[Ma 87] Martin, U.: How to choose the weights in the Knuth Bendix ordering. Proceedings of the Second International Conference on Rewriting Techniques and Applications (RTA 87), Bordeaux, France, May 1987. (Lect. Notes Comput. Sci., vol. 256, pp. 42–53) Berlin Heidelberg New York: Springer 1987

[O'D 77] O'Donnell, M.J.: Computing in systems described by equations (Lect Notes Comput. Sci., vol. 58) Berlin Heidelberg New York: Springer 1977

[Ru 85] Rusinowitch, M.: Plaisted ordering and recursive decomposition ordering revisited. Proceedings of the First International Conference on Rewriting Techniques and Applications. (Lect. Notes Comput. Sci., vol. 202) Berlin Heidelberg New York: Springer 1985

[Uz 58] Uzawa, H.: An elementary method for linear programming. In: Arrow, K.J., Hurwicz, L., Uzawa, H. (eds.) Studies in linear and non-linear programming, pp. 179–188. Stanford: Stanford University Press 1958