How to choose the weights in the Knuth Bendix ordering

Ursula Martin*
Department of Computer Science
The University
Manchester M13 9PL

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 twofold. First we give some examples to indicate the flexibility of the method. Then we give a simple and practical algorithm, based on the simplex algorithm, for determining whether or not a set of rules can be ordered by a Knuth Bendix ordering.

1 Introduction

Well-founded orderings on terms arise in several areas of computer science; proving termination [De 85], 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 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 85] for a survey. A term rewriting system such as REVE [Le 83] or Eril [Di 85] 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 together cope with many of the kinds of rule that occur in practise. 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, section 11], "Generating a

*Supported in part by the Science and Engineering Research Council under grant GR/D 81718

canonical term rewriting system is similar to compilation; you run the 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 then if the weights are the same, comparing the subterms lexicographically. 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 sections 3 and 4. It is easy to implement, and fast to execute. It is very flexible; the choice of weights means that there are infinitely many possible 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 section 5 a simple and practical decision procedure, based on the simplex algorithm, 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 a certain set of linear equalities has a particular kind of solution. We do not, in this paper, explore variations of the Knuth Bendix ordering, although our method would also apply.

The idea of ordering terms by associating to each a polynomial rather than a weight appears in [MN 70] and [L 75]. 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 86].

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

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

example REVE 2.4 contains three different ones. Howver in contrast to the Knuth Bendix orderings there are only $n!$ recursive path orderings on terms in n function symbols. We shall see in section 4 that in the case when we have two unary function symbols the recursive path ordering is closely related to a special case of the Knuth Bendix ordering.

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 can be adapted to treat the Knuth Bendix ordering in a similar way. At each stage (roughly speaking) we have a set of rules, and a set of inequalities which have a solution, proving termination of the rules. When we generate a new equation the two possible orderings correspond to two possible new inequalities to add to the previous set, and testing when the new sets have a solution tells us which way to order the equation. Of course both ways may be possible, which leads to the generation of a search tree.

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. 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 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.

Now we choose a partial ordering $\gg$ on Σ, and we assign a non-negative real number, or weight, $w(f)$, to each operator f, subject to

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

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, the Knuth Bendix ordering, which we denote by $> (\gg, w)$, or just $>$. To do this we assign a weight to each term as follows. We let w be the minimum of the weights of the operators of zero arity, so that

$$w = min\{w(f) | \alpha(f) = 0\} > 0.$$

Then 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

1 $w(s) > w(t)$ and $n(x,s) \geq n(x,t)$ for all variables x

or

2 $w(s) = w(t)$ and $n(x,s) = n(x,t)$ for all variables x

and either

(a) $s = f^n(x)$ and $t = x$ for some $n \geq 1$,

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 $1 \leq k \leq \alpha(f)$.

This is slightly different from the definition given in Knuth and Bendix's paper–they require the ordering $\gg$ to be total. A different formulation of the Knuth Bendix ordering is given in [De 82]. It is always a simplification ordering, so that in particular it is well founded, and if $s > t$ then $s\sigma > t\sigma$ for any substitution σ. This means that any rewriting system $R = \{L_i \rightarrow R_i | 1 \leq i \leq n\}$ over T with $L_i > R_i$ for each i is terminating.

3 Examples

Here are some examples.

Example 1

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

```
1      x * e -> x
2      e* x -> x
3      i(x) * x -> e
4      x * i(x) -> e
5      (x * y) * z -> x * (y * z)
6      i(e) -> e
7      i(x) * (x * y)  -> y
8      x * (i(x) * y)  -> y
9      i(i(x)) -> x
10     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 $i \gg * \gg e$, 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 [De 85].

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

We have $\Sigma = \{-, +, *, e\}$, where $\alpha(+) = \alpha(*) = 2, \alpha(-) = 1$, and $\alpha(e) = 0$. Let $w(+) = w(*) = w(e) = 1$, and let $w(-) = 0$. Let the ordering on the operators be $- \gg + \gg *$. 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 84] and Cherifa and Lescanne [CL 86]. It arose in the study of transformations of FP 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 with $w(*) > 0$.

4 An extended example

In this section we describe how the Knuth Bendix ordering orders terms in two unary function symbols s and t and one nullary operator e , for different choices of weight. For comparison we also describe the recursive path ordering, which turns out to be closely related to one of the Knuth Bendix orderings.

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

Case 1 – s and t have positive weight

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

$$w(s)n(s,r) + w(t)n(t,r) = n.$$

Terms of the same weight are ordered lexicographically from the left. The ordering depends only on the ratio of $w(s)$ to $w(t)$, and the operator precedence $s >_1 t$ or $t >_2 s$, and not on the values of $w(s)$ and $w(t)$. Thus for each positive real r there are two Knuth Bendix orderings, namely $> (>_1, w)$ and $> (>_2, w)$ where $w(s) = r, w(t) = 1$, 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 s's and t's.

Example 1: $\mathbf{s \gg t, w(s) = w(t)}$

Terms are ordered by weight, and then lexicographically.

$$t < s < t^2 < ts < st < s^2 < t^3 < t^2s < tst < ts^2 < st^2 < sts < s^2t < s^3 < \dots$$

Example 2: $\mathbf{t \gg s, w(s) = w(t)}$

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.

$$s < t < s^2 < st < ts < t^2 < s^3 < s^2t < sts < st^2 < ts^2 < tst < t^2s < t^3 < \dots$$

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

$$t < t^2 < s < t^3 < ts < st < t^4 < t^2s < tst < st^2 < s^2$$

$$< t^5 < t^3s < t^2st < tst^2 < ts^2 < st^3 < sts < sst < t^6 < \dots$$

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

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

$$t < s < t^2 < st < ts < t^3 < s^2 < st^2 < tst < t^2s < t^4 < \dots$$

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

$$t < s < t^2 < ts < st < t^3 < s^2 < t^2s < tst < st^2 < t^4 < ..$$

$$ts^2 < sts < s^2t < \dots$$

Case 2 – s has zero weight

Now suppose that $w(s) = 0$, in which case $w(t) > 0$ and $s \gg t$. There are now infinitely many terms of given weight. It is easy to see that $u > v$ if and only $n(t, u) > n(t, v)$ or $n(t, u) = n(t, v)$ and u is greater than v in a lexicographic ordering from the left, that is if $u = f_1 f_2 \dots f_n$ and $v = g_1 g_2 \dots g_m$, either there is an i with $1 \leq i \leq min(m, n)$ and $f_1 = g_1, \dots, f_{i-1} = g_{i-1}, f_i = t, g_i = s$ or $n > m$ and $f_1 = g_1, \dots, f_m = g_m$.

Thus we have

$$s < s^2 < s^3 \ldots < s^i < \ldots < t < ts < ts^2 < \ldots < ts^i < \ldots$$

$$< st < sts < sts^2 < \ldots < sts^i < \ldots < t^2 < t^2s < \ldots < t^2s^i$$

$$< \ldots < tst < tsts < \ldots < tsts^i < \ldots.$$

The recursive path ordering

The recursive path ordering $>_*$ [De 82] on terms in two unary operators s and t and a nullary operator e, with $t > s > e$, is defined recursively as follows.

$$u = f(a) >_* e$$

and

$$u = f(a) >_* g(b) = v$$

if and only if

$$f = g \textit{ and } a >_* b$$

or

$$f = t \textit{ and } g = s \textit{ and } u >_* b$$

or

$$f = s \textit{ and } g = t \textit{ and } a >_* v \textit{ or } a = v.$$

It is not hard to see that $u >_* v$ if and only $n(t,u) > n(t,v)$ or $n(t,u) = n(t,v)$ and u is greater than v in a lexicographic ordering from the right.

5 The decision procedure

In this section we describe a procedure to determine whether or not there is a Knuth Bendix ordering which will prove the termination of a finite set of rules. Similar techniques may be used in an implementation of the Knuth Bendix algorithm to determine which way to order new critical pairs, and hence to build up an ordering iteratively, but we do not consider this here.

Our procedure works by maintaining a set of inequalities which the weights must satisfy, and an operator precedence. Properties of the solution to the inequalities are determined using the simplex method. If at any stage the inequalities have no appropriate solution, or the operator precedence cannot be extended, we halt with failure. Otherwise there is a Knuth Bendix ordering which orders the terms, and explicit values for the weights can be found if required.

Thus we begin by describing linear inequalities.

Linear inequalities

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

$$a_{i1}x_1 + \cdots + a_{in}x_n \geq 0, \quad 1 \leq i \leq m.$$

If $\mathbf{x}$ is a vector we write $\mathbf{x} \geq 0$ 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 to T if and only if

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

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

Let C be the set of all solutions to T. Then C is a finite cone [Gale 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 $r \geq 0$ then $r\mathbf{x}$ is in C.
3. There are vectors $\mathbf{u_1}, \dots, \mathbf{u_k}$ such that

$$C = \{r_1\mathbf{u_1} + \cdots + r_k\mathbf{u_k} | r_i \in \mathbf{R}, r_i \geq 0, 1 \leq i \leq k\}.$$

The solutions of T can be found by using the simplex method to determine the vectors $\mathbf{u_1}, \dots, \mathbf{u_k}$. A thorough account is given in, for example, [Chvatal 1983, Chapters 16 and 18], and we do not describe the method here. We shall need to know some properties of the solution, and these are given in the next lemma.

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

1. If

$$b_1x_1 + \cdots b_nx_n \geq 0$$

is in I_T then

$$b_1y_1 + \cdots b_ny_n = 0$$

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

2. If $C = \{(0, \dots, 0)\}$ then $I_T = T$.

3. If $I_T \neq T$ then there is an element $(z_1, \dots, z_n)$ of C such that

$$d_1z_1 + \cdots d_nz_n > 0$$

whenever

$$d_1x_1 + \cdots d_nx_n \geq 0$$

is in $T \setminus I_T$.

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

Proof Let $H_i = \{\mathbf{x} \in \mathbf{R}^n | x_i = 0\}$, the ith coordinate hyperplane. The set of solutions to T, that is to $A\mathbf{y} \geq 0$, 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. If $A\mathbf{x}$ and $i \in I$ then

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

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 $1 \leq j \leq m$ and

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

Then the cone AC lies in $\cup_{j \in J} H_j$. 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

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

whenever

$$d_1 x_1 + \cdots d_n x_n \geq 0$$

is in $T \setminus I_T$.□

Lemma 2 *Let $H_i = \{\mathbf{x} \in \mathbf{R}^n | x_i = 0\}$, and let C be a cone in $\mathbf{R}^n$. If I is a non-empty subset of $\{1, \ldots, n\}$ and $C \subseteq \cup_{i \in I} H_i$ 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 $C \subseteq \cup_{i \in I} H_i$ there is a k in I with $c_k = 0$. Now suppose that $C \not\subseteq H_k$. Pick $d \in C \setminus H_k$. 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. □

Partial orderings

If $>$ is a partial ordering on a finite set S and H is a set of pairs of elements of S then either there is a unique minimal partial ordering $>_*$ on S with $> \subseteq >_*$, and $s >_* t$ for all pairs (s, t) in H, or there is no partial ordering $>_*$ with these properties. If such a partial ordering exists we denote it by $extend(>, H)$; if not we set $extend(>, H) = \perp$. If $H = \{(s, t)\}$ and $s \not> t$ then $extend(>, H)$ exists if and only if $s \not< t$. We can determine if $extend(>, H)$ exists by extending $>$ by one element of H at a time.

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 terms with $hd(s) = hd(t)$ we want to replace them by two smaller terms, the reduced pair $(red(s), red(t))$. Thus if $s = f(s_1, \ldots, s_{\alpha(f)})$, $t = f(t_1, \ldots, t_{\alpha(f)})$, $s \neq t$, and $\alpha(f) \geq 1$ we let $(red(s), red(t)) = (s_i, t_i)$ where $1 \leq i \leq \alpha(f), s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$ and $s_i \neq t_i$. Otherwise we let $(red(s), red(t)) = (s, t)$.

The algorithm

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

At any stage in the algorithm we have

Q a set of pairs of terms consisting of $Q_0 = \{(L_i, R_i) | L_i \to R_i\}$ together with $(red(s), red(t))$ for certain (s,t) in Q

$>$ a partial ordering on the function symbols

Associated to these we have

E(Q) a set of inequalities in the variables $w(f)$ for each function symbol $f \in \Sigma$, and the variable w, given as follows

$\mathbf{e_{st}}$ each pair (s,t) in Q contributes

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

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

$$w(f) \geq 0$$

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

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

$\mathbf{e_w}$ the variable w is positive

$$w \geq 0$$

I(Q) the subset $I(Q) = I_{E(Q)}$ of $E(Q)$ defined in Lemma 1, obtained using the simplex algorithm or otherwise.

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

```
$Q := Q_0, > := \{\}$
repeat
    $Q := Q \cup \{(red(s), red(t)) | e_{st} \in I(Q)\}$
    if $(e_{st} \in E(Q)$ and $\exists x \in X : n(x,s) < n(x,t))$ then *fail* ;
    if $(e_{st} \in I(Q)$ and $\exists x \in X : n(x,s) \neq n(x,t))$ then *fail* ;
    if $e_w \in I(Q)$ then *fail* ;
    if $(e_f, e_g \in I(Q)$ for unary $f$ and $g$, $f \neq g)$ then *fail* ;
    if $(e_f \in I(Q)$ for $f$ unary$)$ then
        if $(extend(>, \{(f,h) | f \neq h, h \in \Sigma\}) = \perp)$ then $fail$
        else $(>:= extend(>, \{(f,h) | f \neq h, h \in \Sigma\}))$;
    if $(\exists (s,t) \in I(Q) : (s \in X))$ then $fail$;
    if $(\exists (s,t) \in I(Q) : (t \notin X$ and $hd(s) \neq hd(t)))$ then
        if $(extend(>, \{(hd(s), hd(t))\}) = \perp)$ then $fail$
        else $(>:= extend(>, \{(hd(s), hd(t))\}))$;
until $(Q = Q \cup \{(red(s), red(t)) | e_{st} \in I(Q)\})$
```

We have the following theorem,

Theorem 1 *This algorithm always terminates. R can be ordered by a Knuth Bendix ordering if and only if it does not fail.*

Acknowledgements

I should like to thank Jeremy Dick, John Kalmus and the referee for their helpful comments on an earlier version of this paper.

References

[Be 84] F. Bellegarde, Rewriting Systems on FP expressions that reduce the number of sequences they yield, Symposium on LISP and functional programming, ACM, Austin, USA, 1984

[BM 79] R. S. Boyer and J. S. Moore, A Computational Logic, Academic Press, New York 1979

[CL 86] A. B. Cherifa and P. Lescanne, An actual implementation of a procedure that mechanically proves termination of rewriting systems based on inequalities between polynomial interpretations, *in* Eighth International Conference on Automated Deduction, Lecture Notes in Computer Science 230, Springer 1986

[Chvatal 83] V. Chvatal, Linear Programming, Freeman, New York, 1983

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

[De85] N. Dershowitz, Termination of Rewriting , *in* Proceedings of the First International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science 202, Springer 1985

[DF 85] R. Forgaard and D. Detlefs, A procedure for automatically proving the termination of a set of rewrite rules, *in* Proceedings of the First International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science 202, Springer 1985

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

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

[Gale 60] D. Gale, The theory of linear economic models, McGraw Hill, New York 1960

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

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

[HO 80] G. Huet and D. C. Oppen, Equations and rewrite rules: a survey, *in* Formal Language Theory, Perspectives and Open Problems, R. Book (ed), Academic Press 1980, 349-405

[KB 70] D. E. Knuth and P. B. Bendix, Simple word problems in universal algebras, *in* Computational problems in abstract algebra, ed J. Leech, Pergamon 1970, 263-297

[L 75] D. S. Lankford, Canonical algebraic simplification in computational logic, Mem ATP-25, Automatic Theorem Proving Project, University of Texas, Austin TX, May 1975

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

[Le 86] P. Lescanne, Divergence of the Knuth Bendix completion procedure and termination orderings, Bulletin of the European Association for Theoretical Computer Science 30 (1986), 80-83

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

[O'D 77] M. J. O'Donnell, Computing in systems described by equations, Lecture Notes in Computer Science 58, Springer 1977

[Ru 85] M.Rusinowitch, Plaisted ordering and recursive decomposition ordering revisited, *in* Proceedings of the First International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science 202, Springer 1985