Orienting Equalities with the Knuth-Bendix Order

Konstantin Korovin
MPI Informatik
D-66123 Saarbrücken, Germany
korovin@mpi-sb.mpg.de

Andrei Voronkov
Computer Science Department
University of Manchester, M139PL, UK
voronkov@cs.man.ac.uk

Abstract

Orientability of systems of equalities is the following problem: given a system of equalities $s_1 \simeq t_1, \ldots, s_n \simeq t_n$, does there exist a simplification ordering $\succ$ which orients the system, that is for every $i \in \{1, ..., n\}$, either $s_i \succ t_i$ or $t_i \succ s_i$. This problem can be used in rewriting for finding a canonical rewrite system for a system of equalities and in theorem proving for adjusting simplification orderings during completion. We prove that (rather surprisingly) the problem can be solved in polynomial time when we restrict ourselves to the Knuth-Bendix orderings.

1. Introduction

In this section we give an informal overview of the results proved in this paper. The formal definitions will be given in the next section.

Let $\succ$ be any order on ground terms and $l \rightarrow r$ be a rewrite rule. We say that $\succ$ *orients* $l \rightarrow r$ if for every ground instance $l' \rightarrow r'$ of $l \rightarrow r$ we have $l' \succ r'$. We write $l \succeq r$ if for every ground instance $l' \rightarrow r'$ of $l \rightarrow r$ we have $l' \succ r'$ or $l' = r'$. We say that $\succ$ *orients an equality* $s \simeq t$, if it orients either the rewrite rule $s \rightarrow t$ or the rewrite rule $t \rightarrow s$. The *orientability problem* is a problem of determining whether there exists a simplification ordering which orders a given system of equalities and rewrite rules.

There are situations where we want to check if there *exists* a simplification order on ground terms that orients a given system of (possibly non-ground) rewrite rules and equalities. One example is when we would like to obtain a canonical rewrite system equivalent to a given set of equalities. This can be achieved in the following way. We can apply a paramodulation-based completion procedure using different ways of orienting critical pairs. Of course, this could be done by trying, one by one, all possible orientations of the current set of equalities, but this may result in a combinatorial explosion of the search space.

A similar situation arises in paramodulation-based reasoning, where we may want to adjust a reduction ordering during the saturation of a clause set. Again, we may want to check whether a given set of equalities is orientable at all.

It is hard to understand how to solve the orientability problem in general, since there is no good description of the class of *all* simplification orderings. Instead of solving the general problem, one may want to solve the orientability problem for known large classes of orderings, such as lexicographic path orderings or the Knuth-Bendix orderings. In this paper we investigate the orientability problem for the class of the Knuth-Bendix orderings and prove a surprising result that for this class of orderings there exists a polynomial time algorithm for checking orientability.

Algorithms for, and complexity of, orientability problem of systems of rewrite rules for the Knuth-Bendix orders and various versions of the recursive path orders are considered in [Dick, Kalmus and Martin 1990, Korovin and Voronkov 2003, Lescanne 1984, Detlefs and Forgaard 1985, Krishnamoorthy and Narendran 1985].

Related problems of solving ordering constraints for lexicographic, recursive path orders and for KBO are NP-complete, see [Comon 1990, Jouannaud and Okada 1991, Nieuwenhuis 1993, Narendran, Rusinowitch and Verma 1999, Korovin and Voronkov 2001*a*]. However, to check if $\succ$ orients $l \rightarrow r$, it is sufficient to check solvability of a *single* ordering constraint $r \succeq l$. This problem is NP-complete for LPO as shown in [Comon and Treinen 1994], and therefore the problem of checking if an LPO orients a single rewrite rule is coNP-complete. As far as we know, the orientability problem for sets of equalities has not been previously studied.

We had to omit some proofs due to lack of space. For example, we removed all of the proofs related to rewrite systems as they can be found in [Korovin and Voronkov 2003]. But we give a reasonably detailed proof related to the main idea of our algorithm, which allows one to avoid exponential search in the space of all possible orientations.

2. Preliminaries

A *signature* is a finite set of function symbols with associated arities. In this paper Σ denotes an arbitrary signature. *Constants* are function symbols of the arity 0. We assume that Σ contains at least one constant. We denote variables by x, y, z, constants by a, b, c, d, e, function symbols by f, g, h, and terms by l, r, s, t. A *rewrite rule* is a pair of terms (l, r), usually denoted by $l \to r$. An *equality* is a multiset of two terms s, t, usually denoted by $s \simeq t$. Note that $s \simeq t$ and $t \simeq s$ are regarded as the same equality. A *system of equalities and rewrite rules* is a finite set of equalities and rewrite rules. An expression E (e.g. a term, equality, or a rewrite rule) is called *ground* if no variable occurs in E. Denote the set of natural numbers by $\mathbb{N}$.

The Knuth-Bendix order is a family of orders parametrized by two parameters: a weight function and a precedence relation.

DEFINITION 2.1 (weight function) We call a *weight function* on Σ any function $w : \Sigma \to \mathbb{N}$ such that (i) $w(a) > 0$ for every constant $a \in \Sigma$, (ii) there exist at most one unary function symbol $f \in \Sigma$ such that $w(f) = 0$. Given a weight function w, we call $w(g)$ the *weight* of g. The *weight* of any ground term t, denoted $|t|$, is defined as follows: for every constant c we have $|c| = w(c)$ and for every function symbol g of a positive arity $|g(t_1, \ldots, t_n)| = w(g) + |t_1| + \ldots + |t_n|$.

DEFINITION 2.2 A *precedence relation* on Σ is any total order $\gg$ on Σ. A precedence relation $\gg$ is said to be *compatible* with a weight function w if for every unary function symbol f, if $w(f) = 0$, then f is the greatest element w.r.t. $\gg$.

DEFINITION 2.3 (Knuth-Bendix order) Let w be a weight function on Σ and $\gg$ a precedence relation on Σ compatible with w. The instance of the *Knuth-Bendix order induced by* $(w, \gg)$ is the binary relation $\succ$ on the set of ground terms of Σ defined as follows. For all ground terms $t = g(t_1, \ldots, t_n)$ and $s = h(s_1, \ldots, s_k)$ we have $t \succ s$ if one of the following conditions holds:

1. $|t| > |s|$;
2. $|t| = |s|$ and $g \gg h$;
3. $|t| = |s|$, $g = h$ and for some $1 \leq i \leq n$ we have $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}$ and $t_i \succ s_i$.

The compatibility condition ensures that every instance of the Knuth-Bendix order is a simplification order total on ground terms.

In the sequel we will often refer to the least and the greatest terms among the terms of the minimal weight for a given instance of KBO. It is easy to see that every term of the minimal weight is either a constant of the minimal weight, or a term $f^n(c)$, where c is a constant of the minimal weight, and $w(f) = 0$. Therefore, the least term of the minimal weight is always the constant of the minimal weight which is the least among all such constants w.r.t. $\gg$. This constant is also the least term w.r.t. $\succ$.

The greatest term of the minimal weight exists if and only if there is no unary function symbol of the weight 0. In this case, this term is the constant of the minimal weight which is the greatest among such constants w.r.t. $\gg$.

DEFINITION 2.4 (substitution) A *substitution* is a mapping from a set of variables to the set of terms. A substitution θ is *grounding* for an expression E (i.e., term, rewrite rule etc.) if for every variable x occurring in E the term $\theta(x)$ is ground. We denote by $E\theta$ the expression obtained from E by replacing in it every variable x by $\theta(x)$. A *ground instance* of an expression E is any expression $E\theta$ which is ground.

The following definition is central to this paper.

DEFINITION 2.5 (orientability) An instance $\succ$ of KBO *orients* a rewrite rule $l \to r$ if for every ground instance $l' \to r'$ of $l \to r$ we have $l' \succ r'$. An instance $\succ$ of KBO *orients* an equality $s \simeq t$ if it orients either $s \to t$ or $t \to s$. An instance of KBO *orients a system* R of equalities and rewrite rules if it orients every equality and rewrite rule in R.

In [Korovin and Voronkov 2001*b*, Korovin and Voronkov 2003] we proved that orientability can be solved in polynomial time for systems consisting of rewrite rules only.

Let us show that the problem of orientability of systems of equalities is at least as hard as the problem of orientability of systems of rewrite rules.

PROPOSITION 2.6 *There exists a logarithmic-space algorithm which for a given system of rewrite rules R produces a system of equalities E such that R is orientable by an instance of KBO if and only if so is E.*

PROOF. Consider a rewrite system R. Let g be a new binary symbol and c be a new constant which do not occur in R. Consider a rewrite system R' which is obtained from R by replacing each rewrite rule $l \to r$ with a rewrite rule $g(l, x) \to g(r, c)$ where x is a variable which does not occur in $l \to r$. Let us check that R is orientable by the Knuth–Bendix order if and only if R' is. Indeed, let $\succ$ be an instance of the Knuth–Bendix order which orients R then we extend parameters of this order to the new symbols in such a way that c becomes a minimal term in this order. Now it is straightforward to check that the obtained order $\succ'$ orients R'. For the converse direction let us note that if an instance of the Knuth–Bendix order orients R' then the same instance also orients R.

To conclude the proof we consider the system of equalities E induced by R'. Since in each rewrite rule from R' there exists a variable occurring in the left hand-side and not occurring in the right hand-side it is easy to see that E is orientable if and only if R' is orientable. $\square$

Note that this reduction also works for the lexicographic path orderings.

3. Systems of homogeneous linear inequalities

In our proofs and in the algorithm we will use several properties of homogeneous linear inequalities. The definitions related to systems of linear inequalities can be found in standard textbooks, see e.g. [Schrijver 1998]. We will denote column vectors of variables by X, integer or real vectors by V, W, integer or real matrices by A, B. Column vectors consisting of 0's will be denoted by $\mathbf{0}$. The set of real numbers is denoted by $\mathbb{R}$, and the set of non-negative real numbers by $\mathbb{R}^+$.

DEFINITION 3.1 (homogeneous linear inequalities) A *homogeneous linear inequality* has the form either $VX \geq 0$ or $VX > 0$. A *system of homogeneous linear inequalities* is a finite set of homogeneous linear inequalities.

Solutions (real or integer) to systems of homogeneous linear inequalities are defined as usual. When we write a system of homogeneous linear inequalities as $AX \geq \mathbf{0}$, we assume that every inequality in the system is of the form $VX \geq 0$ (but not of the form $VX > 0$).

In [Korovin and Voronkov 2003] one can find a proof of a key property of integer systems of homogeneous linear inequalities: the existence of a real solution implies the existence of an integer solution.

LEMMA 3.2 *Let $\mathbb{W}$ be a system of homogeneous linear inequalities with an integer matrix. Let V be a real solution to this system and for some subsystem of $\mathbb{W}$ with the matrix C we have $CV > \mathbf{0}$. Then there exists an integer solution V' to $\mathbb{W}$ for which we also have $CV' > \mathbf{0}$.* $\square$

The following lemma was proved in [Martin 1987] for the systems of linear homogeneous inequalities over the real numbers. A simpler proof can also be found in [Korovin and Voronkov 2003]. We formulate this lemma for integer solutions, which makes no difference by Lemma 3.2.

LEMMA 3.3 *Let $AX \geq \mathbf{0}$ be a system of homogeneous linear inequalities where A is an integer matrix and let* Sol *be the set of all integer solutions to the system. Then the system can be split into two disjoint subsystems $BX \geq \mathbf{0}$ and $CX \geq \mathbf{0}$ such that*

1. *$BV = \mathbf{0}$ for every $V \in$ Sol.*
2. *If C is non-empty then there exists a solution $V \in$ Sol such that $CV > \mathbf{0}$.* $\square$

Let $\mathbb{W}$ be a system of homogeneous linear inequalities. We will call the subsystem $BX \geq \mathbf{0}$ of $\mathbb{W}$ the *degenerate subsystem* if the following holds. Denote by C the matrix of the complement to $BX \geq \mathbf{0}$ in $\mathbb{W}$ and by Sol the set of all integer solutions to $\mathbb{W}$. Then

1. $BV = \mathbf{0}$ for every $V \in$ Sol.
2. If C is non-empty then there exists a solution $V \in$ Sol such that $CV > \mathbf{0}$.

For every system $\mathbb{W}$ of homogeneous linear inequalities the degenerate subsystem of $\mathbb{W}$ will be denoted by $\mathbb{W}^=$. Note that the degenerate subsystem is defined for arbitrary systems, not only those of the form $AX \geq 0$.

The following lemma follows from Lemmas 3.3 and 3.2.

LEMMA 3.4 *Let $\mathbb{W}$ be a system of homogeneous linear inequalities with an integer matrix and its degenerate subsystem is different from $\mathbb{W}$. Let C be the matrix of the complement of the degenerate subsystem. Then there exists an integer solution V to $\mathbb{W}$ such that $CV > \mathbf{0}$.* $\square$

We will call any such solution to $\mathbb{W}$ *best-positive*.

We will use the following fundamental property of system of homogeneous linear inequalities to prove the lemma below.

THEOREM 3.5 *Let $AX \geq \mathbf{0}$ be a system of homogeneous linear inequalities, where A is an integer matrix. Then there exists a finite number of integer vectors $V_1, \ldots, V_n$ such that the set of solutions to $AX \geq \mathbf{0}$ is*

$$\{r_1V_1 + \ldots + r_nV_n \mid r_1, \ldots, r_n \in \mathbb{R}^+\}. \qquad (1)$$

The proof can be found in, e.g. [Schrijver 1998].

LEMMA 3.6 *Consider a system of homogeneous linear inequalities $\mathbb{W}$ and an integer homogeneous linear inequality $UX > 0$. If there exists a solution S to the system $\mathbb{W} \cup \{UX > 0\}$ then the degenerate subsystem of $\mathbb{W}$ coincides with the degenerate subsystem of $\mathbb{W} \cup \{UX > 0\}$.*

PROOF. We can assume that $\mathbb{W}$ is of the form $AX \geq \mathbf{0}$. By Theorem 3.5 we can find integer vectors $V_1, \ldots, V_n$ such that the set of solutions to $AX \geq \mathbf{0}$ is (1). Since we have that $US > 0$ for a solution to $AX \geq \mathbf{0}$ then for some $1 \leq i \leq n$ we have $UV_i > 0$. Also from Lemma 3.3 we have that there exists a solution S to $AX \geq \mathbf{0}$ such that for each inequality $WX \geq 0$ from the nondegenerate subsystem of $AX \geq \mathbf{0}$ we have $WS > 0$. Now we consider a positive number r such that $rUV_i + US > 0$, such a number always exists since we have $UV_i > 0$. It is straightforward to check that $rV_i + S$ satisfies the required properties. □

COROLLARY 3.7 *Consider a system of homogeneous linear inequalities $\mathbb{W}$, then $\mathbb{W}^=$ coincides with $(\mathbb{W}^=)^=$.*

PROOF. From the previous lemma it follows that if we add to the system $\mathbb{W}^=$ an inequality from the non-degenerate subsystem of $\mathbb{W}$ then we obtain a new system with the degenerate part equal to $(\mathbb{W}^=)^=$. If we continue this process until we have added all inequalities from the non-degenerate subsystem of $\mathbb{W}$ we obtain that $\mathbb{W}^=$ coincides with $(\mathbb{W}^=)^=$. □

Let us consider a system of homogeneous linear inequalities $\mathbb{W}$. We say that an equality $VX = 0$ *follows* from $\mathbb{W}$ if for every solution S to $\mathbb{W}$ we have $VS = 0$. Now our goal is to show that for every equality $VX = 0$ if it follows from $\mathbb{W}$ then it already follows from the degenerate subsystem of $\mathbb{W}$. For this we use the following theorem.

THEOREM 3.8 *(Fundamental theorem of linear inequalities.) Let $A_1, \ldots, A_m, U$ be vectors in n–dimensional space. Then, either*

1. *U is a non-negative linear combination of linearly independent vectors from $A_1, \ldots, A_m$, or*
2. *there exists a vector W such that $UW < 0$ and $A_iW \geq 0$ for $1 \leq i \leq m$.*

PROOF. The proof can be found in, e.g. [Schrijver 1998]. □

LEMMA 3.9 *Consider a system of homogeneous linear inequalities W with an integer matrix and an integer homogeneous linear equality $UX = 0$. If $UX = 0$ follows from W then it follows from the degenerate subsystem of W.*

PROOF. We can assume that $\mathbb{W}$ is of the form $AX \geq \mathbf{0}$. First we prove that if $UX = 0$ follows from $AX \geq \mathbf{0}$ then the vector U is a non-negative linear combination of the row vectors of the degenerate subsystem of $AX \geq \mathbf{0}$. For this we apply Theorem 3.8 to the row vectors of the matrix A and the vector U. There are two possible cases.

- U is a non-negative linear combination of the row vectors from the matrix A. $1 \leq i \leq k$ Let us show that in this combination all coefficients of the vectors from the non-degenerate subsystem of $AX \geq \mathbf{0}$ are equal to zero. Otherwise, we consider such a vector C. Since C is a row vector from the non-degenerate subsystem, there exists a solution S to $AX \geq \mathbf{0}$ such that $CS > 0$ and therefore $US > 0$, which contradicts to the assumption that $UX = 0$ follows from $AX \geq \mathbf{0}$.
- there exists a vector W such that for each row vector Q of A we have $QW \geq 0$ and also $UW < 0$. But this contradicts to the assumption that $UX = 0$ follows from $AX \geq \mathbf{0}$.

We have shown that U is a non-negative linear combination of the row vectors from the degenerate subsystem of $AX \geq \mathbf{0}$.

Now using Corollary 3.7 it is easy to see that $UX = 0$ follows from the degenerate subsystem of $AX \geq \mathbf{0}$. □

The following result due to [Khachiyan 1979] is well-known.

LEMMA 3.10 *The existence of a real solution to a system of linear inequalities can be decided in polynomial time.* □

This lemma and Lemma 3.2 imply the following key result, see [Korovin and Voronkov 2003].

LEMMA 3.11 *(i) The existence of an integer solution to an integer system of homogeneous linear inequalities can be decided in polynomial time. (ii) If an integer system $\mathbb{W}$ of homogeneous linear inequalities has a solution, then its degenerate subsystem $\mathbb{W}^=$ can be found in polynomial time.* □

4. Constraints

In Section 6 we will present an algorithm for orientability by the Knuth-Bendix order. The algorithm works not only with equalities and rewrite rules. It also uses linear inequalities on the weights of the signature symbols, constraints on the precedence relation, and some additional information. All this information will be formalized using the notion of *constraint*.

Let $>$ be any binary relation on ground terms. We extend it lexicographically to a relation on tuples of ground terms as follows: we have $\langle l_1, \ldots, l_n \rangle > \langle r_1, \ldots, r_n \rangle$ if for some $i \in \{1, \ldots, n\}$ we have $l_1 = r_1, \ldots, l_{i-1} = r_{i-1}$ and $l_i > r_i$.

In the sequel we assume that Σ is a fixed signature. We also assume that different equalities and rewrite rules have disjoint sets of variables. This can be achieved by renaming variables.

Our algorithm will work on *constraints*. Orientability of a rewrite rule or an equality are special kinds of constraints. In addition, there are constraints on the precedence relation and on the weights of the symbols in Σ. The algorithm will transform constraints step by step. We will show that every step preserves satisfiability of constraints. Before defining constraints, we introduce special kind of variables, called *marked variables*. Intuitively, marked variables range only over terms of the minimal weight.

DEFINITION 4.1 (Constraint) An *atomic constraint* is an expression having one of the following forms:

1. $\langle l_1, \ldots, l_n \rangle \; ?{\succ} \; \langle r_1, \ldots, r_n \rangle$, where $l_1, \ldots, l_n, r_1, \ldots, r_n$ are terms. Such constraints are called *rewriting constraints*.
2. $\langle l_1, \ldots, l_n \rangle \prec ?{\succ} \langle r_1, \ldots, r_n \rangle$, where $l_1, \ldots, l_n, r_1, \ldots, r_n$ are terms. Such constraints are called *orientability constraints*.
3. A (strict or non-strict) homogeneous linear inequality over the variables $\{w_g \mid g \in \Sigma\}$. Such constraints are called *weight constraints*.
4. $g \; ?{\gg} \; h$, where $g, h \in \Sigma$. Such constraints are called *precedence constraints*.
5. $gtmw(c)$, where c is a constant.

A *constraint* C is a conjunction $C_1 \wedge \ldots \wedge C_n$ of (zero or more) atomic constraints. Alternatively, we will sometimes regard a constraint as the *set* $\{C_1, \ldots, C_n\}$ of all atomic constraints in it. In this case we say that C *contains* the atomic constraints $C_1, \ldots, C_n$. Conjunctions (or sets) of atomic rewriting constraints are called rewriting constraints, and similar for the orientability, weight, and precedence constraints.

We consider constraints as conditions on the Knuth-Bendix order. Every instance of the Knuth-Bendix order which satisfies all atomic constraints in C is called a *solution* to this constraint. In order to define solutions, let us give a technical definition. A substitution σ is called an *admissible substitution* for a weight function w if for every marked variable x the term $\sigma(x)$ is a ground term of the minimal weight, that is $w(\sigma(x))$ is equal to the smallest weight of a constant in Σ.

DEFINITION 4.2 (Solution) Let $\succ$ be the instance of the Knuth-Bendix order induced by $(w, \gg)$. This instance is called a *solution* to an atomic constraint C if one of the following conditions holds.

1. C is a rewriting constraint $\langle l_1, \ldots, l_n \rangle \; ?{\succ} \; \langle r_1, \ldots, r_n \rangle$, and for every admissible substitution σ we have $\langle l_1\sigma, \ldots, l_n\sigma \rangle \succ \langle r_1\sigma, \ldots, r_n\sigma \rangle$.
2. C is an orientability constraint $\langle l_1, \ldots, l_n \rangle \prec ?{\succ} \langle r_1, \ldots, r_n \rangle$ and $\succ$ is a solution to either $\langle l_1, \ldots, l_n \rangle \; ?{\succ} \; \langle r_1, \ldots, r_n \rangle$ or $\langle r_1, \ldots, r_n \rangle \; ?{\succ} \; \langle l_1, \ldots, l_n \rangle$.
3. C is a weight constraint and w solves C in the following sense: replacement of each w_g by $w(g)$ gives a tautology.
4. C is a precedence constraint $g \; ?{\gg} \; h$, and $g \gg h$.
5. C is a constraint $gtmw(c)$, and c is the greatest term of the minimal weight.

A solution to an arbitrary constraint C is a solution to every atomic constraint in C. A constraint C is *satisfiable* if it has a solution. A constraint C_1 *implies* a constraint C_2, denoted by $C_1 \supset C_2$, if every solution to C_1 is also a solution to C_2. Two constraints are *equivalent* if they have the same solutions.

We will often write atomic constraints in $\mathbb{W}$ in an equivalent form, for example write $w_c > w_e$ instead of $w_c - w_e > 0$.

We will now show how to reduce the orientability problem for the systems of equalities and rewrite rules to the satisfiability problem for constraints.

Let R be a system of equalities and rewrite rules such that every two different rules in R have disjoint variables. Denote by C_R the conjunction of the following constraints:

1. rewriting constraints $\langle l \rangle \; ?{\succ} \; \langle r \rangle$ such that $l \to r$ belongs to R.
2. orientability constraints $\langle l \rangle \prec ?{\succ} \langle r \rangle$ such that $l \simeq r$ belongs to R.

The following lemma is straightforward.

LEMMA 4.3 *An instance $\succ$ of KBO orients R if and only if $\succ$ is a solution to C_R.* □

5. Rich constraints and trivial signatures

For technical reasons, it will be convenient for us to work with constraints which contain enough information to decide some properties of its solutions, for example, which of the constants of Σ is the smallest. Such constraints are introduced here and called *rich constraints*.

DEFINITION 5.1 (Rich Constraint) A constraint C is called *rich* if

1. C contains all the constraints $w_c > 0$, where $c \in \Sigma$ is a constant, and all the constraints $w_g \geq 0$, where $g \in \Sigma$ is a non-constant function symbol.

2. There is a constant $e \in \Sigma$ such that for all constants $c \in \Sigma$ distinct from e, C contains the atomic constraint $c \mathrel{?\!\succ} e$.

3. Exactly one of the following conditions holds. (i) There is a unary function symbol $f \in \Sigma$ such that C contains the atomic constraint $w_f \leq 0$, all of the atomic constraints $f \mathrel{?\!\gg} g$ for $g \in \Sigma$ distinct from f, and all of the atomic constraints $w_g > 0$ for unary function symbols g distinct from f. (ii) For some constant $d \in \Sigma$, C contains the constraint $gtmw(d)$. For every unary function symbol $g \in \Sigma$, C contains the atomic constraint $w_g > 0$.

LEMMA 5.2 *Let C be a rich constraint and the KBO $\succ$ induced by $(w, \gg)$ satisfies C.*

1. *e is the least term with respect to $\succ$.*

2. *There exists a unary function symbol $f \in \Sigma$ such that $w(f) = 0$ if and only if (i) holds. In addition, if such a function f does not exist, then the constraint contains $gtmw(d)$, and hence d is the greatest term of the minimal weight.*

3. *There exists more than one term of the minimal weight if and only if either there exists a unary function symbol $f \in \Sigma$ such that $w(f) = 0$ or there exists a constant $d \in \Sigma$ distinct from e such that C contains the atomic constraint $gtmw(d)$.*

LEMMA 5.3 *The orientability problem can be solved in polynomial time if the orientability problem for rich constraints can be solved in polynomial time.*

The idea of the proof of the lemma is as follows: one can "guess" the following properties of solutions: (a) which of the constants is smallest one, (b) does there exist a unary function symbol of the weight 0, (c) if such a function does not exist, then which of the constants is the greatest term of the minimal weight. Note that we make only a constant number of guesses.

For technical reasons, we will distinguish two kinds of signatures. Essentially, our algorithm depends on whether the weights of terms are restricted or not. For the so-called *non-trivial* signatures, the weights are not restricted. When we present the orientability algorithm for the non-trivial signatures, we will use the fact that terms of sufficiently large weights always exist. The (straightforward) proof for trivial signatures is omitted due to lack of space.

A signature is called *trivial* if it contains no function symbols of arity ≥ 2, and at most one unary function symbol. Note that a signature is non-trivial if and only if it contains either a function symbol of arity ≥ 2 or at least two function symbols of arity 1. The following lemma is straightforward, the proof can be found in [Korovin and Voronkov 2003].

LEMMA 5.4 *Let Σ be a non-trivial signature and w be a weight function for Σ. Then for every integer m there exists a ground term of the signature Σ such that $|t| > m$.*

6. The orientability algorithm

In this section we only consider non-trivial signatures. Our algorithm works as follows.

Given a system R of equalities or rewrite rules, we build the initial constraint $C = C_R$. Using Lemma 5.3 we can assume that C is rich. We will always denote by e the constant such that C contains all atomic constraints $c \mathrel{?\!\succ} e$, where c is a constant distinct from e (such a constant e exists, since C is rich). Then we repeatedly transform C as described below. We call the *essential size* of a constraint the total number of occurrences of function symbols and variables in its rewriting and orientability part. Every transformation step will either terminate with success or failure, or replace an equality by a rewrite rule, or decrease the essential size of C.

At each step the constraint C can be represented as a conjunction $\mathbb{R} \wedge \mathbb{W} \wedge \mathbb{O} \wedge \mathbb{P} \wedge \mathbb{G}$, where $\mathbb{R}$ is a rewrite constraint, $\mathbb{W}$ a weight constraint, $\mathbb{O}$ an orientability constraints, $\mathbb{P}$ a precedence constraint, and $\mathbb{G}$ either empty or has the form $gtmw(c)$.

For every variable x and term t, denote by $n(x, t)$ the number of occurrences of x in t. For example, $n(x, g(x, h(y, x))) = 2$. Likewise, for every function symbol $g \in \Sigma$ and term t, denote by $n(g, t)$ the number of occurrences of g in t. For example, $n(h, g(x, h(y, x))) = 1$.

For every term t, denote by $W(t)$ the linear expression obtained as follows. Let v be the number of occurrences of variables in t. Then

$$W(t) = \sum_{g \in \Sigma} n(g, t) w_g + v w_e. \qquad (2)$$

For example, if $t = h(x, x, c, e, f(y))$, then

$$W(l) = w_h + w_c + w_f + 4w_e.$$

6.1. The algorithm

The algorithm works as follows. Every step consists of a number of state transformations, beginning with REWRITE RULE defined below. During the algorithm, we will perform two kinds of *satisfiability checks*:

- The *satisfiability check on* $\mathbb{W}$ is the check whether $\mathbb{W}$ has a solution. If it does not, we terminate with failure.

- The *satisfiability check on* $\mathbb{P}$ is the check whether $\mathbb{P}$ is satisfiable, that is the transitive closure of the set $\{(g,h) \mid g \mathbin{?\!\gg} h$ is an atomic constraint in $\mathbb{P}\}$ is irreflexive. i.e., contains no pair (g,g). If $\mathbb{P}$ is inconsistent, then we terminate with failure.

It is not hard to argue that both kinds of satisfiability checks can be performed in polynomial time. The satisfiability check on $\mathbb{W}$ is polynomial by Lemma 3.11. The satisfiability check on $\mathbb{P}$ is polynomial since the transitive closure of a binary relation can be computed in polynomial time, see, e.g. [Cormen, Leiserson and Rivest 1991].

When any of the sets $\mathbb{W}$ or $\mathbb{P}$ changes, we assume that we perform the corresponding satisfiability check and terminate with failure if it fails.

We will label parts of the algorithm, these labels will be used in the proof of its soundness.

REWRITE RULE.

(R0) Do the following transformations while possible. If $\mathbb{R}$ contains a tuple inequality $\langle l_1,\ldots,l_n\rangle \mathbin{?\!\succ} \langle l_1,\ldots,l_n\rangle$, terminate with failure. Otherwise, if $\mathbb{R}$ contains a tuple inequality $\langle l,l_1,\ldots,l_n\rangle \mathbin{?\!\succ} \langle l,r_1,\ldots,r_n\rangle$, replace it by $\langle l_1,\ldots,l_n\rangle \mathbin{?\!\succ} \langle r_1,\ldots,r_n\rangle$.

Now $\mathbb{R}$ has the form

$$\begin{array}{c}\langle l_1,L_1\rangle \mathbin{?\!\succ} \langle r_1,R_1\rangle,\\ \cdots\\ \langle l_k,L_k\rangle \mathbin{?\!\succ} \langle r_k,R_k\rangle,\end{array}\qquad(3)$$

such that each l_i is a term different from the corresponding term r_i.

(R1) For all x and i such that $n(x,l_i) > n(x,r_i)$, mark the variable x.

(R2) If for some i there exists an unmarked variable x such that $n(x,l_i) < n(x,r_i)$, then terminate with failure.

(R3) Add to $\mathbb{W}$ all the linear inequalities $W(l_i) \geq W(r_i)$ for all i and perform the satisfiability check on $\mathbb{W}$.

Now compute $\mathbb{W}^=$. If $\mathbb{W}^=$ contains none of the inequalities $W(l_i) \geq W(r_i)$ proceed to EQUALITY. Otherwise, for all i such that $(W(l_i) \geq W(r_i)) \in \mathbb{W}^=$ apply the applicable case below, depending on the form of l_i and r_i.

(R4) If $l_i = g(s_1,\ldots,s_n)$ and $r_i = h(t_1,\ldots,t_p)$, where g is different from h, then replace the constraint $\langle l_i,L_i\rangle \mathbin{?\!\succ} \langle r_i,R_i\rangle$ by $g \mathbin{?\!\gg} h$. Perform the satisfiability check on $\mathbb{P}$.

(R5) If $l_i = g(s_1,\ldots,s_n)$ and $r_i = g(t_1,\ldots,t_n)$, then replace $\langle l_i,L_i\rangle \mathbin{?\!\succ} \langle r_i,R_i\rangle$ by $\langle s_1,\ldots,s_n,L_i\rangle \mathbin{?\!\succ} \langle t_1,\ldots,t_n,R_i\rangle$.

(R6) If (l_i,r_i) has the form (x,y), where x and y are different variables, do the following. (Note that at this point both x and y are marked.) If L_i is empty, then terminate with failure. If the constraint guarantees the existence of more than one term of the minimal weight (see Lemma 5.2), then also terminate with failure. Otherwise, replace $\langle l_i,L_i\rangle \mathbin{?\!\succ} \langle r_i,R_i\rangle$ by $\langle L_i\rangle \mathbin{?\!\succ} \langle R_i\rangle$.

(R7) If (l_i,r_i) has the form (x,t), where t is not a variable, do the following. If t is different from e, or L_i is empty, then terminate with failure. Otherwise replace in L_i and R_i the variable x by e, obtaining L_i' and R_i' respectively, and then replace $\langle l_i,L_i\rangle \mathbin{?\!\succ} \langle r_i,R_i\rangle$ by $\langle L_i'\rangle \mathbin{?\!\succ} \langle R_i'\rangle$.

(R8) If (l_i,r_i) has the form (t,x), where t is not a variable, do the following. If t contains x, remove $\langle l_i,L_i\rangle \mathbin{?\!\succ} \langle r_i,R_i\rangle$ from C. Otherwise, if t is a nonconstant or L_i is empty, terminate with failure. (Note that at this point x is marked and $(W(t) \geq W(x)) \in \mathbb{W}^=$.) Let now t be a constant c. If C does not contain the atomic constraint $gtmw(c)$, then terminate with failure. Otherwise replace in L_i and R_i the variable x by c, obtaining L_i' and R_i' respectively, and then replace $\langle l_i,L_i\rangle \mathbin{?\!\succ} \langle r_i,R_i\rangle$ by $\langle L_i'\rangle \mathbin{?\!\succ} \langle R_i'\rangle$.

After this step repeat REWRITE RULE.

EQUALITY.

(E0) Do the following transformations while possible. If $\mathbb{O}$ contains an atomic constraint $\langle s_1,\ldots,s_n\rangle \mathbin{\prec\!?\!\succ} \langle s_1,\ldots,s_n\rangle$, terminate with failure. Otherwise, if $\mathbb{O}$ contains $\langle s,s_1,\ldots,s_n\rangle \mathbin{\prec\!?\!\succ} \langle s,t_1,\ldots,t_n\rangle$, replace it by $\langle s_1,\ldots,s_n\rangle \mathbin{\prec\!?\!\succ} \langle t_1,\ldots,t_n\rangle$.

If $\mathbb{O}$ is empty, proceed to TERMINATE. Otherwise, $\mathbb{O}$ now has the form

$$\begin{array}{c}\langle s_1,S_1\rangle \mathbin{\prec\!?\!\succ} \langle t_1,T_1\rangle,\\ \cdots\\ \langle s_k,S_k\rangle \mathbin{\prec\!?\!\succ} \langle t_k,T_k\rangle,\end{array}\qquad(4)$$

such that each s_i is a term different from the corresponding term t_i.

(E1) If, for some i and variable x we have $n(x,s_i) > n(x,t_i)$, replace $\langle s_i,S_i\rangle \mathbin{\prec\!?\!\succ} \langle t_i,T_i\rangle$ by $\langle s_i,S_i\rangle \mathbin{?\!\succ} \langle t_i,T_i\rangle$ and proceed to REWRITE RULE. Likewise, if for some i and variable x we have $n(x,t_i) > n(x,s_i)$, replace $\langle s_i,S_i\rangle \mathbin{\prec\!?\!\succ} \langle t_i,T_i\rangle$ by $\langle t_i,T_i\rangle \mathbin{?\!\succ} \langle s_i,S_i\rangle$ and proceed to REWRITE RULE.

Note that after this step for every i and variable x, the number of occurrences of x in s_i coincides with its number of occurrences in t_i.

Now for each $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ in $\mathbb{O}$ such that $\mathbb{W} \supset W(s_i) = W(t_i)$ apply (E2) below, if there is no such tuples in $\mathbb{O}$ then proceed to TERMINATE.

(E2) If the top symbols of s_i and t_i coincide, i.e., $s_i = g(u_1, \ldots, u_m)$ and $t_i = g(v_1, \ldots, v_m)$, then replace $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ by $\langle u_1, \ldots, u_m, S_i\rangle \prec ?\succ \langle v_1, \ldots, v_m, T_i\rangle$ and proceed to REWRITE RULE. Otherwise, remove $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ from the constraint, and proceed to EQUALITY.

TERMINATE. If the constraint contains $gtmw(d)$, then for all constants c different from d such that $w_c \geq w_e$ belongs to $\mathbb{W}^=$ add $d\ ?\gg c$ to the constraint. Perform the satisfiability check on $\mathbb{P}$. Terminate with success.

Note that after TERMINATE, for each $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ in $\mathbb{O}$ either $\mathbb{W} \wedge W(s_i) > W(t_i)$ or $\mathbb{W} \wedge W(t_i) > W(s_i)$ is satisfiable.

6.2. Correctness

In this section we prove correctness of the algorithm and show how to find a solution when the algorithm terminates with success. The correctness will follow from a series of lemmas asserting that all of the transformation steps performed by the algorithm preserve the set of solutions. Although the algorithm can terminate with success without eliminating all orientability constraints, we will be able to show that in this case the resulting constraint is always satisfiable. To prove this we employ lemmas on homogeneous linear inequalities from Section 3.

We will use the following notation and terminology in the proof. We say that a step of the algorithm is *equivalence-preserving* if the set of solutions to the constraint before this step coincides with the set of solutions after the step. When we use substitutions in the proof, we always assume that the substitutions are grounding for the relevant terms.

The following lemma can be proved by a straightforward application of the definition of solution to a state.

LEMMA 6.1 (satisfiability check) *If satisfiability check on $\mathbb{W}$ or on $\mathbb{P}$ terminates with failure, then $\mathbb{S}$ has no solution.* □

In [Korovin and Voronkov 2003] we presented an algorithm for checking orientability of systems of rewrite rules by the KBO. Since REWRITE RULE uses the same steps as the algorithm of [Korovin and Voronkov 2003] (though written in a slightly different notation), we can deduce the following lemma about REWRITE RULE.

LEMMA 6.2 *Steps (R0)–(R8) are equivalence-preserving.* □

LEMMA 6.3 *Step (E1) is equivalence-preserving .*

PROOF. Consider $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ in $\mathbb{O}$ such that for some variable x, $n(x, s_i) > n(x, t_i)$. To prove the lemma it suffices to show that if we replace $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ by $\langle t_i, S_i\rangle\ ?\succ \langle s_i, T_i\rangle$ in our constraint, then we obtain an unsatisfiable constraint C'. Assume that C' has a solution $\succ$. Let σ be any substitution grounding for this tuple inequality. Take any term u and modify σ by mapping x into u, obtaining σ^u_x. We have

$$\begin{aligned}&|s_i\sigma^u_x| - |t_i\sigma^u_x| = \\ &|s_i\sigma| - |t_i\sigma| + (n(x, s_i) - n(x, t_i)) \cdot (|u| - |x\sigma|).\end{aligned}$$

Since there exist terms of an arbitrarily large weight, for a term u of a large enough weight we have $|s_i\sigma^u_x| > |t_i\sigma^u_x|$, which contradicts to the assumption $\langle t_i, S_i\rangle\sigma^u_x \succ \langle s_i, T_i\rangle\sigma^u_x$. □

LEMMA 6.4 *Step (E2) is equivalence-preserving.*

PROOF. At this step we have that for each variable x the number of occurrences of x in s_i is the same as the the number of occurrences of x in t_i and therefore neither s_i nor t_i is a variable. Also, for every solution to the constraint and every grounding substitution σ we have $|s_i\sigma| = |t_i\sigma|$.

Consider the case when top symbols of s_i and t_i coincide, i.e., $s_i = g(u_1, \ldots, u_m)$ and $t_i = g(v_1, \ldots, v_m)$. Then it easy to see that if we have a solution to our constraint such that $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ the same solution will satisfy $\langle u_1, \ldots, u_m, S_i\rangle \prec ?\succ \langle v_1, \ldots, v_m, T_i\rangle$ and vice versa.

Now we consider the case when top symbols of s_i and t_i are different, i.e. $s_i = g(\bar{u})$ and $t_i = h(\bar{v})$. It suffices to show that if we have a solution $\succ$ to the constraint after removing $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$, denoted as C', then $\succ$ is also a solution to $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$. Consider a solution $\succ$ to C' induced by $(w, \gg)$. Assume that $g \gg h$, then for every substitution σ we have $s_i\sigma \succ t_i\sigma$ since $|s_i\sigma| = |t_i\sigma|$. Similar, if $h \gg g$ then for every substitution σ we have $t_i\sigma \succ s_i\sigma$. □

Let us show that TERMINATE preserves satisfiability.

LEMMA 6.5 TERMINATE *is equivalence-preserving.*

PROOF. Let us show that the addition of all atomic constraints $d\ ?\gg c$ at this step preserves equivalence. If C has no solution, then this is obvious. Otherwise, take any solution $\succ$ to C and let this solution be induced by $(w, \gg)$.

We know C contains $gtmw(d)$, hence d must be the greatest term of the minimal weight. It is not hard to argue that at the TERMINATE step, $\mathbb{W}$ contains all constraints $w_c \geq w_e$, where c is a constant different from d. If such a constraint belongs to $\mathbb{W}^=$, then we have $w(c) = w(e)$, hence c is a term of the minimal weight. But then we must have $d \succ c$. By the construction, C also contains $w_e \geq w_d$, so $C \supset w_e = w_d$. Therefore, $d \succ c$ also implies $d \gg c$, and the addition of $d \mathrel{?{\gg}} c$ does not change the set of solutions. $\square$

We have shown that all steps of our algorithm preserve satisfiability of constraints. Now we show that if the algorithm terminates with success then the constraint is satisfiable, moreover we will be able to find a solution to the constraint in polynomial time.

We call a constraint C *saturated* if application of our orientability algorithm to C does not change C and terminates with success.

LEMMA 6.6 *If a constraint $C = \mathbb{R} \wedge \mathbb{W} \wedge \mathbb{P} \wedge \mathbb{G} \wedge \mathbb{O}$ is saturated then the constraint $C' = \mathbb{R} \wedge \mathbb{W} \wedge \mathbb{P} \wedge \mathbb{G}$ is satisfiable.*

PROOF. We have that $\mathbb{W}$ is satisfiable, and for each rewriting constraint $\langle l_i, L_i\rangle \mathrel{?{\succ}} \langle r_i, R_i\rangle$ the weight constraint $W(l_i) \geq W(r_i)$ does not belong to $\mathbb{W}^=$. By Lemma 3.4 there exists a solution w to $\mathbb{W}$ such that for each rewriting constraint $\langle l_i, L_i\rangle \mathrel{?{\succ}} \langle r_i, R_i\rangle$ we have $W(l_i)w > W(r_i)w$. Let $\succ$ be an ordering induced by $(w, \gg)$, where $\gg$ is an arbitrary extension of $\mathbb{P}$ to a linear order. We need to show that $\succ$ satisfies the rewriting constraint $\mathbb{R}$ (constraints $\mathbb{W}$,$\mathbb{P}$,$\mathbb{G}$, are obviously satisfied). For this let us consider a tuple $\langle l_i, L_i\rangle \mathrel{?{\succ}} \langle r_i, R_i\rangle$ in $\mathbb{R}$ and an admissible substitution σ and show that $\langle l_i, L_i\rangle\sigma \succ \langle r_i, R_i\rangle\sigma$. From algorithm (rules (R1), (R2)) we have that for each unmarked variable x, $n(x, l_i) = n(x, r_i)$, also for each marked variable y we have $|y\sigma| = w(e)$. Therefore

$$|l_i\sigma| - |r_i\sigma| = W(l_i)w - W(r_i)w > 0,$$

this shows that $\langle l_i, L_i\rangle\sigma \succ \langle r_i, R_i\rangle\sigma$. $\square$

LEMMA 6.7 *Every saturated constraint is satisfiable.*

PROOF. Consider a saturated constraint

$$C = \mathbb{R} \wedge \mathbb{W} \wedge \mathbb{P} \wedge \mathbb{G} \wedge \mathbb{O}.$$

We show that C is satisfiable by induction on the number of atomic constraints in $\mathbb{O}$. If $\mathbb{O}$ is empty then the claim follows from Lemma 6.6. Now assume that $\mathbb{O}$ is not empty. Since C is saturated we have that for each atomic constraint $\langle s_i, S_i\rangle \mathrel{\prec?\succ} \langle t_i, T_i\rangle$ in $\mathbb{O}$ either $\mathbb{W} \wedge W(s_i) > W(t_i)$ or $\mathbb{W} \wedge W(t_i) > W(s_i)$ is satisfiable. Assume that $\mathbb{W} \wedge W(s_i) > W(t_i)$ is satisfiable, then add $W(s_i) > W(t_i)$ to $\mathbb{W}$ and remove $\langle s_i, S_i\rangle \mathrel{\prec?\succ} \langle t_i, T_i\rangle$ from $\mathbb{O}$, obtaining $\mathbb{W}'$ and $\mathbb{O}'$ respectively. Let us show that the obtained constraint

$$C' = \mathbb{R} \wedge \mathbb{W}' \wedge \mathbb{P} \wedge \mathbb{G} \wedge \mathbb{O}'$$

is saturated. From Lemma 3.6 it follows that the degenerate subsystem of $\mathbb{W}'$ coincides with the degenerate subsystem of $\mathbb{W}$ and since C is saturated we have that none of the rules (R0)–(R8), (E0), (E1) can change the constraint C'. Also from Lemma 3.9 it follows that for each $\langle s'_i, S'_i\rangle \mathrel{\prec?\succ} \langle t'_i, T'_i\rangle$ in $\mathbb{O}'$ either $\mathbb{W}' \wedge W(s'_i) > W(t'_i)$ or $\mathbb{W}' \wedge W(t'_i) > W(s'_i)$ is satisfiable. Hence, rule (E2) also can not change the constraint C' and we conclude that C' is saturated. Since $\mathbb{O}'$ contains less atomic constraints than $\mathbb{O}'$ and C' is saturated, we can apply the induction hypothesis. $\square$

6.3. Time complexity

Provided that we use a polynomial-time algorithm for solving systems of homogeneous linear inequalities, and a polynomial-time algorithm for transitive closure, a careful analysis of our algorithm shows the following.

LEMMA 6.8 *The algorithm runs in time polynomial of the size of the system of rewrite rules.* $\square$

7. Main results

Lemmas 6.1–6.7 guarantee that the orientability algorithm is correct. Lemma 6.8 implies that it runs in polynomial time. Hence we obtain the following theorem.

THEOREM 7.1 *The problem of the existence of an instance of KBO which orients a given system of equalities and rewrite rules can be solved in the time polynomial in the size of the system. Moreover, if the system of equalities and rewrite rules is orientable by an instance of KBO we can find a such instance in polynomial time.* $\square$

In [Korovin and Voronkov 2003] we proved that the problem of orientability by the KBO is P-complete for systems of rewrite rules, moreover it is P-hard even for ground rewrite rule systems. Therefore, the following result follows from [Korovin and Voronkov 2003] and Proposition 2.6.

THEOREM 7.2 *The problem of orientability of systems of equalities and rewrite rules by the KBO is P-complete. Moreover, it is P-hard even for systems consisting only of equalities.* $\square$

References

COMON H. [1990], 'Solving symbolic ordering constraints', *International Journal of Foundations of Computer Science* **1**(4), 387–411.

COMON H. AND TREINEN R. [1994], Ordering constraints on trees, *in* S. Tison, ed., 'Trees in Algebra and Programming: CAAP'94', Vol. 787 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 1–14.

CORMEN T., LEISERSON C. AND RIVEST R. [1991], *Introduction to Algorithms*, The MIT Press.

DETLEFS D. AND FORGAARD R. [1985], A procedure for automatically proving the termination of a set of rewrite rules, *in* J.-P. Jouannaud, ed., 'Rewriting Techniques and Applications, First International Conference, RTA-85', Vol. 202 of *Lecture Notes in Computer Science*, Springer Verlag, Dijon, France, pp. 255–270.

DICK J., KALMUS J. AND MARTIN U. [1990], 'Automating the Knuth-Bendix ordering', *Acta Informatica* **28**(2), 95–119.

JOUANNAUD J.-P. AND OKADA M. [1991], Satisfiability of systems of ordinal notations with the subterm property is decidable, *in* J. Albert, B. Monien and M. Rodríguez-Artalejo, eds, 'Automata, Languages and Programming, 18th International Colloquium, ICALP'91', Vol. 510 of *Lecture Notes in Computer Science*, Springer Verlag, Madrid, Spain, pp. 455–468.

KHACHIYAN L. [1979], 'A polynomial algorithm in linear programming', *Soviet Mathematical Doklady* **20**, 191–194.

KOROVIN K. AND VORONKOV A. [2001*a*], Knuth-Bendix ordering constraint solving is NP-complete, *in* F. Orejas, P. Spirakis and J. van Leeuwen, eds, 'Automata, Languages and Programming, 28th International Colloquium, ICALP 2001', Vol. 2076 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 979–992.

KOROVIN K. AND VORONKOV A. [2001*b*], Verifying orientability of rewrite rules using the Knuth-Bendix order, *in* A. Middeldorp, ed., 'Rewriting Techniques and Applications, 12th International Conference, RTA 2001', Vol. 2051 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 137–153.

KOROVIN K. AND VORONKOV A. [2003], 'Verifying orientability of rewrite rules using the Knuth-Bendix order', *Information and Computation* pp. 1–29. To appear.

KRISHNAMOORTHY M. AND NARENDRAN P. [1985], 'On recursive path ordering', *Theoretical Computer Science* **40**, 323–328.

LESCANNE P. [1984], Term rewriting systems and algebra, *in* R. Shostak, ed., '7th International Conference on Automated Deduction, CADE-7', Vol. 170 of *Lecture Notes in Computer Science*, pp. 166–174.

MARTIN U. [1987], How to choose weights in the Knuth-Bendix ordering, *in* 'Rewriting Techniques and Applications', Vol. 256 of *Lecture Notes in Computer Science*, pp. 42–53.

NARENDRAN P., RUSINOWITCH M. AND VERMA R. [1999], RPO constraint solving is in NP, *in* G. Gottlob, E. Grandjean and K. Seyr, eds, 'Computer Science Logic, 12th International Workshop, CSL'98', Vol. 1584 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 385–398.

NIEUWENHUIS R. [1993], 'Simple LPO constraint solving methods', *Information Processing Letters* **47**, 65–69.

SCHRIJVER A. [1998], *Theory of Linear and Integer Programming*, John Wiley and Sons.