J Autom Reasoning manuscript No.
(will be inserted by the editor)

KBO Orientability

Harald Zankl · Nao Hirokawa · Aart Middeldorp

Received: date / Accepted: date

Abstract This article presents three new approaches to prove termination of rewrite systems with the Knuth-Bendix order efficiently. The constraints for the weight function and for the precedence are encoded in (pseudo-)propositional logic or linear arithmetic and the resulting formula is tested for satisfiability using dedicated solvers. Any satisfying assignment represents a weight function and a precedence such that the induced Knuth-Bendix order orients the rules of the encoded rewrite system from left to right. This means that in contrast to the dedicated methods our approach does not directly solve the problem but transforms it to equivalent formulations for which sophisticated back-ends exist. In order to make all approaches complete we present a method to compute upper bounds on the weights. Furthermore, our encodings take dependency pairs into account to increase the applicability of the order.

Keywords Knuth-Bendix order · Term rewriting · Termination

1 Introduction

This article is concerned with proving termination of term rewrite systems (TRSs) with the Knuth-Bendix order (KBO), a method invented by Knuth and Bendix in [25] well before termination research in term rewriting became a very popular and competitive endeavor (as witnessed by the annual termination competition).[1] We know of only two termination tools that contain an implementation of KBO, AProVE [16] and T$_{\text{T}}$T [21], but neither of these tools used KBO in the competition for the TRS category. This is perhaps due to the fact that

The first and third authors are supported by FWF (Austrian Science Fund) project P18763 and the second author is supported by the Grant-in-Aid for Young Scientists 20800022 of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

H. Zankl · A. Middeldorp
Institute of Computer Science, University of Innsbruck, Austria
E-mail: harald.zankl@uibk.ac.at, aart.middeldorp@uibk.ac.at

N. Hirokawa
School of Information Science, Japan Advanced Institute of Science and Technology, Japan
E-mail: hirokawa@jaist.ac.jp

[1] `http://termination-portal.org/wiki/Termination_Competition`

the algorithms known for deciding KBO orientability [7,28] are not easy to implement efficiently, despite the fact that the problem is known to be decidable in polynomial time [28]. The aim of this article is to make KBO a more attractive choice for termination tools by presenting three simple[2] encodings of KBO orientability into propositional satisfiability (SAT), pseudo boolean satisfiability (PB), and satisfiability modulo theories (SMT where the theory of choice is linear arithmetic), such that checking satisfiability of the resulting constraints amounts to proving KBO orientation.

Kurihara and Kondo [29] were the first to encode a termination method for term rewriting into propositional logic. They showed how to encode orientability with respect to the lexicographic path order as a satisfaction problem. In the recent past a vast number of SAT encodings has been proposed for various termination methods. Codish *et al.* [3] presented a more efficient formulation for the properties of a precedence. In [4, 41] encodings of argument filterings are presented which can be combined with propositional encodings of reduction pairs in order to obtain logic-based implementations of the dependency pair method. Encodings of other termination methods are described in [12–15,26,27,36,43,44]. We show that in the case of KBO one can improve upon pure SAT encodings in two ways; on the one hand the implementation effort can be reduced by applying a more expressive constraint language, on the other hand performance can be improved by choosing the right back-end.

In Section 2 the necessary definitions for KBO are presented. Section 3 shows that weights can be bound from above. Then Section 4 introduces a purely propositional encoding of KBO. In Section 5 an alternative encoding is given using pseudo-boolean constraints whereas Section 6 addresses the SMT approach using linear arithmetic which combines the simplicity of the pure SAT encoding with the benefits of PB. Extensions to the dependency pair setting [1, 17, 19, 20] are described in Section 7 before we compare the power and run times of our implementations with the ones of AProVE and T$_\top$T in Section 8 and show the enormous gain in efficiency. We draw some conclusions and summarize the main contributions of this article in Section 9.

Some of the results appeared in earlier conference papers: the SAT and PB encodings [42] and the SAT encoding within the dependency pair setting [41]. Section 3 and the results for SMT are new.

2 Preliminaries

We assume familiarity with the basics of term rewriting (e.g. [2]). Below we recall some important definitions needed in the remainder of the article. A signature $\mathcal{F}$ consists of function symbols equipped with fixed arities. The set of terms constructed from a signature $\mathcal{F}$ and a set of variables $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$. For a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$, $|t|$ denotes its size and $|t|_a$ for $a \in \mathcal{F} \cup \mathcal{V}$ denotes how often the symbol a occurs in t. We have $\mathrm{root}(f(t_1,\ldots,t_n)) = f$. A quasi-order $\succsim$ is a reflexive and transitive relation with strict part $\succ$ ($a \succ b$ if and only if $a \succsim b$ and $b \not\succsim a$) and equivalence part $\sim$ ($a \sim b$ if and only if $a \succsim b$ and $b \succsim a$).

Next we recall the definition of KBO. A *quasi-precedence* $\succsim$ (*strict precedence* $\succ$) is a quasi-order (strict part of a quasi-order) on a signature $\mathcal{F}$. Sometimes we find it convenient to call a quasi-precedence simply precedence. A *weight function* for a signature $\mathcal{F}$ is a pair (w, w_0) consisting of a mapping $w\colon \mathcal{F} \to \mathbb{N}$ and a constant $w_0 > 0$ such that $w(c) \geqslant w_0$ for every constant $c \in \mathcal{F}$. Let $\mathcal{F}$ be a signature and (w, w_0) a weight function for $\mathcal{F}$. The *weight*

[2] Here, simple should be understood in the sense of "easy to implement" and not as "easy to find".

of a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ is defined as follows:

$$w(t) = \begin{cases} w_0 & \text{if } t \text{ is a variable,} \\ w(f) + \sum_{i=1}^{n} w(t_i) & \text{if } t = f(t_1,\ldots,t_n). \end{cases}$$

A weight function (w, w_0) is *admissible* for a quasi-precedence $\succsim$ if $f \succsim g$ for all function symbols g whenever f is a unary function symbol with $w(f) = 0$.

Definition 1 ([25, 7, 37]) Let $\succsim$ be a quasi-precedence and (w, w_0) a weight function. We define the *Knuth-Bendix order* $>_{\mathrm{kbo}}$ on terms inductively as follows: $s >_{\mathrm{kbo}} t$ if $|s|_x \geqslant |t|_x$ for all variables $x \in \mathcal{V}$ and either

(a) $w(s) > w(t)$, or
(b) $w(s) = w(t)$ and one of the following alternatives holds:
  (1) $t \in \mathcal{V}$, $s \in \mathcal{T}(\mathcal{F}^{(1)}, \{t\})$, and $s \neq t$, or
  (2) $s = f(s_1,\ldots,s_n)$, $t = g(t_1,\ldots,t_m)$, $f \sim g$, and there exists an $1 \leqslant i \leqslant \min\{n,m\}$ such that $s_i >_{\mathrm{kbo}} t_i$ and $s_j = t_j$ for all $1 \leqslant j < i$, or
  (3) $s = f(s_1,\ldots,s_n)$, $t = g(t_1,\ldots,t_m)$, and $f \succ g$.

where $\mathcal{F}^{(n)}$ denotes the set of all function symbols $f \in \mathcal{F}$ of arity n. Thus in case (b)(1) the term s consists of a nonempty sequence of unary function symbols applied to the variable t (since $s \neq t$ and $|s|_x \geqslant |t|_x$ for all $x \in \mathcal{V}$).

Specializing the above definition to (the reflexive closure of) a strict precedence, one obtains the definition of KBO in [2], except that we restrict weight functions to have range $\mathbb{N}$ instead of $\mathbb{R}^{\geqslant 0}$. According to results in [28, 31] this does not decrease the power of the order for finite TRSs.

Lemma 1 *A TRS $\mathcal{R}$ is terminating whenever there exist a quasi-precedence $\succsim$ and an admissible weight function (w, w_0) such that $\mathcal{R} \subseteq >_{\mathrm{kbo}}$.* □

Example 1 The TRS SK90/2.42[3] consisting of the rules

$$\begin{aligned}
\mathsf{flatten}(\mathsf{nil}) &\to \mathsf{nil} & \mathsf{rev}(\mathsf{nil}) &\to \mathsf{nil} \\
\mathsf{flatten}(\mathsf{unit}(x)) &\to \mathsf{flatten}(x) & \mathsf{rev}(\mathsf{unit}(x)) &\to \mathsf{unit}(x) \\
\mathsf{flatten}(x +\!\!+ y) &\to \mathsf{flatten}(x) +\!\!+ \mathsf{flatten}(y) & \mathsf{rev}(x +\!\!+ y) &\to \mathsf{rev}(y) +\!\!+ \mathsf{rev}(x) \\
\mathsf{flatten}(\mathsf{unit}(x) +\!\!+ y) &\to \mathsf{flatten}(x) +\!\!+ \mathsf{flatten}(y) & \mathsf{rev}(\mathsf{rev}(x)) &\to x \\
\mathsf{flatten}(\mathsf{flatten}(x)) &\to \mathsf{flatten}(x) & (x +\!\!+ y) +\!\!+ z &\to x +\!\!+ (y +\!\!+ z) \\
x +\!\!+ \mathsf{nil} &\to x & \mathsf{nil} +\!\!+ y &\to y
\end{aligned}$$

can be proved terminating by KBO. The weight function (w, w_0) with $w(\mathsf{flatten}) = w(\mathsf{rev}) = w(+\!\!+) = 0$ and $w(\mathsf{unit}) = w(\mathsf{nil}) = w_0 = 1$ together with the quasi-precedence $\mathsf{flatten} \sim \mathsf{rev} \succ \mathsf{unit} \succ +\!\!+ \succ \mathsf{nil}$ ensures that $l >_{\mathrm{kbo}} r$ for all rules $l \to r$. The use of a quasi-precedence is essential here since the rules $\mathsf{flatten}(x +\!\!+ y) \to \mathsf{flatten}(x) +\!\!+ \mathsf{flatten}(y)$ and $\mathsf{rev}(x +\!\!+ y) \to \mathsf{rev}(y) +\!\!+ \mathsf{rev}(x)$ demand $w(\mathsf{flatten}) = w(\mathsf{rev}) = 0$ but KBO with strict precedence does not allow different unary functions to have weight zero.

[3] Labels in sans-serif font refer to TRSs in the Termination Problems Data Base [32] which is a collection of rewrite systems used for the termination competition.

One can imagine a more general definition of KBO. For instance, in case (b)(2) we could demand that $s_j \sim_{\mathrm{kbo}} t_j$ for all $1 \leqslant j < i$ where $s \sim_{\mathrm{kbo}} t$ if and only if $s \sim t$ and $w(s) = w(t)$. Here $s \sim t$ denotes syntactic equality with respect to equivalent function symbols of the same arity. Another obvious extension would be to compare the arguments according to an arbitrary permutation [35, 36] or as multisets [40, 36]. To keep the discussion and implementation simple, we do not consider such refinements in the sequel.

3 A Bound on Weights

We give a bound on weights to finitely characterize KBO orientability. While there are at most finitely many precedences on a finite signature, the following example demonstrates that there exist TRSs which need arbitrarily large weights.

Example 2 Consider the parametrized TRS consisting of the three rules

$$\mathsf{f}(\mathsf{g}(x,y)) \to \mathsf{g}(\mathsf{f}(x),\mathsf{f}(y)) \qquad \mathsf{h}(x) \to \mathsf{f}(\mathsf{f}(x)) \qquad \mathsf{i}(x) \to \mathsf{h}^k(x)$$

Since the first rule duplicates the function symbol f we must assign weight zero to it. The admissibility condition for the weight function demands that f is a maximal element in the precedence. The second rule ensures that the weight of h is strictly larger than zero. It follows that the minimum weight of $\mathsf{h}^k(x)$ is $k+1$, which at the same time is the minimum weight of $\mathsf{i}(x)$. Thus $w(\mathsf{i})$ is at least k.

Throughout this section we do not distinguish vectors from matrices. We write $\mathbf{e}_i$ for the unit column vector whose i-th position is 1 and all other positions are 0 (the length of the vector is usually clear from the context). Let $A = (a_{ij})_{ij}$ be an $m \times n$ matrix. We define $\|A\| = \max_{i,j} |a_{ij}|$. The i-th row vector of A is denoted by $\mathbf{a}_i$. We say that a vector $\mathbf{x}$ is a solution of A if $A\mathbf{x} \geqslant \mathbf{0}$ and $\mathbf{x} \geqslant \mathbf{0}$. A solution $\mathbf{x}$ that maximizes $\{i \mid \mathbf{a}_i\mathbf{x} > \mathbf{0}\}$ with respect to set inclusion is called *principal*. Unless stated otherwise, matrix entries are integers.

Lemma 2 *Let A be an $m \times n$ matrix. There exists a principal solution $\mathbf{x}$ of A with $\|\mathbf{x}\| \leqslant n^{2^m}(2n\|A\|)^{2^m-1}$.* □

Before proving the lemma we recall the idea from [7] and mention its consequences.

Example 3 Below on the left we give the inequations that the algorithm in [7] starts with (corresponding to a KBO proof attempt with empty precedence) for the TRS from Example 2 where we fix the parameter $k = 2$. The first four equations ensure that every weight is non-negative, the fifth equation captures w_0 and the last three equations express that for every rule $l \to r$ we have $w(l) > w(r)$:

$$\begin{aligned}
w(\mathsf{f}) &\geqslant 0\\
w(\mathsf{g}) &\geqslant 0\\
w(\mathsf{h}) &\geqslant 0\\
w(\mathsf{i}) &\geqslant 0\\
w_0 &> 0\\
w(\mathsf{f}) + w(\mathsf{g}) + 2w_0 &> w(\mathsf{g}) + 2w(\mathsf{f}) + 2w_0\\
w(\mathsf{h}) + w_0 &> 2w(\mathsf{f}) + w_0\\
w(\mathsf{i}) + w_0 &> 2w(\mathsf{h}) + w_0
\end{aligned}
\qquad
\begin{pmatrix}
1 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 1\\
-1 & 0 & 0 & 0 & 0\\
-2 & 0 & 1 & 0 & 0\\
0 & 0 & -2 & 1 & 0
\end{pmatrix}
\begin{pmatrix} w(\mathsf{f})\\ w(\mathsf{g})\\ w(\mathsf{h})\\ w(\mathsf{i})\\ w_0 \end{pmatrix}
\geqslant \mathbf{0}$$

To solve the inequations on the left, the algorithm in [7] starts with the slightly generalized equational system $A\mathbf{x} \geqslant \mathbf{0}$ on the right, which clearly has a (principal) solution. But for a principal solution the algorithm must test if every strict inequation $w(s) > w(t)$ for corresponding terms s and t is indeed satisfied. If this is not the case then it replaces the inequation $w(s) > w(t)$ by $w(s) \geqslant w(t)$, and

(a) fails, if $s \in \mathcal{V}$, $t \in \mathcal{V}$, or $s = t$,
(b) adds the inequation $w(s_i) > w(t_i)$ for the least i with $s_i \neq t_i$ if $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$, or
(c) extends the strict precedence ([7] only supports strict precedences) if possible by
  – $\mathrm{root}(s) \succ f$ for all $f \in \mathcal{F} \setminus \{\mathrm{root}(s)\}$ if $\mathrm{root}(s)$ is unary or
  – $\mathrm{root}(s) \succ \mathrm{root}(t)$ otherwise

and again tries to solve the inequations. In the example no principal solution satisfies the constraint $w(\mathsf{f}(\mathsf{g}(x,y))) > w(\mathsf{g}(\mathsf{f}(x),\mathsf{f}(y)))$ (since it reduces to $0 > w(\mathsf{f})$ contradicting $w(\mathsf{f}) \geqslant 0$) but case (c) applies and the corresponding rule is oriented by extending the precedence with $\mathsf{f} \succ \mathsf{g}, \mathsf{h}, \mathsf{i}$. Since now there exists a principal solution satisfying the current constraints (e.g. $w(\mathsf{f}) = w(\mathsf{g}) = 0$, $w(\mathsf{h}) = w_0 = 1$, and $w(\mathsf{i}) = 3$) the algorithm successfully terminates.

The question remains which A to take for Lemma 2. The example above demonstrates that the matrix A changes during the algorithm. Unfortunately not even the dimension of A stays constant; the columns of A are fixed by the number of unknowns but the rows may increase (cf. case (b)). It is easy to see that the largest dimension of a matrix can be $m \times n$ where $n = |\mathcal{F}| + 1$ and $m = n + \sum_{l \to r \in \mathcal{R}} \min\{\mathrm{depth}(l), \mathrm{depth}(r)\}$ where $\mathrm{depth}(x) = 1$ if $x \in \mathcal{V}$ and $\mathrm{depth}(f(t_1, \ldots, t_q)) = 1 + \max\{\mathrm{depth}(t_i) \mid 1 \leqslant i \leqslant q\}$. Furthermore for every occurring A we have $\|A\| \leqslant \max\{|l|_a, |r|_a \mid l \to r \in \mathcal{R}, a \in \mathcal{F} \cup \mathcal{V}\}$. According to the lemma one can find a principal solution in $[0, n^{2^m}(2n\|A\|)^{2^m-1}]^n$. Hence we get $n^{2^m}(2n\|A\|)^{2^m-1}$ as an upper bound on the weights. Later this number will be referred to as $B_{\mathcal{R}}$. For Example 3 we get $n = 5$, $m = 5 + 7 = 12$, $\|A\| \leqslant 2$, and consequently $B_{\mathcal{R}} = 5^{2^{12}} 20^{2^{12}-1}$. Section 8 shows that in practice much smaller weights suffice. Expressed in terms of the size of the TRS $\mathcal{R}$, the inequality $B_{\mathcal{R}} \leqslant N^{4^{N+1}}$ can be easily shown for $N = 1 + \sum_{l \to r \in \mathcal{R}} (|l| + |r|)$, provided that all symbols from $\mathcal{F}$ appear in $\mathcal{R}$.

In order to prove Lemma 2 we first recall the method of complete description (MCD) introduced by Dick *et al.* [7].

Definition 2 For a row vector $(a_1, \ldots, a_n)$ we define the matrix $(a_1, \ldots, a_n)^\kappa$ as

$$(\mathbf{e}_i \mid a_i \geqslant 0) +\!\!+ (a_j\mathbf{e}_i - a_i\mathbf{e}_j \mid a_i < 0, a_j > 0)$$

with unit vectors $\mathbf{e}_i$, $\mathbf{e}_j$ of length n. The operator $+\!\!+$ merges vectors into a matrix. Let A be an $m \times n$ matrix. For each $0 \leqslant i \leqslant m$ we inductively define S_i^A as follows: S_0^A is the $n \times n$ identity matrix and $S_{i+1}^A = S_i^A(\mathbf{a}_{i+1}S_i^A)^\kappa$. The sum of all column vectors of S_m^A is denoted by $\mathbf{s}^A$.

Proposition 1 ([7]) *Let $A\mathbf{x} \geqslant \mathbf{0}$. Then $\mathbf{s}^A$ is a principal solution of A.* □

The next example demonstrates how to compute $(\cdot)^\kappa$, $+\!\!+$, and $\mathbf{s}^A$.

Example 4 For the matrix $A = \begin{pmatrix} -2 & 0 & 1 & 3 & -1 \end{pmatrix}$ we have

$$S_1^A = S_0^A(\mathbf{a}_1 S_0^A)^\kappa = \mathbf{a}_1^\kappa = \begin{pmatrix} 0&0&0\\ 1&0&0\\ 0&1&0\\ 0&0&1\\ 0&0&0 \end{pmatrix} +\!\!+ \begin{pmatrix} 1&3&0&0\\ 0&0&0&0\\ 2&0&1&0\\ 0&2&0&1\\ 0&0&1&3 \end{pmatrix} = \begin{pmatrix} 0&0&0&1&3&0&0\\ 1&0&0&0&0&0&0\\ 0&1&0&2&0&1&0\\ 0&0&1&0&2&0&1\\ 0&0&0&0&0&1&3 \end{pmatrix} \text{ and } \mathbf{s}^A = \begin{pmatrix} 4\\1\\4\\4\\4 \end{pmatrix}$$

We show that $\mathbf{s}^A$ fulfills the condition of Lemma 2.

Lemma 3 *Let B be a $p \times q$ matrix, C a $q \times r$ matrix and $\mathbf{a}$ a $1 \times q$ row vector. Then the following holds:*

1. $\|BC\| \leqslant q\|B\|\|C\|$,
2. $\|B\mathbf{a}^\kappa\| \leqslant 2\|B\|\|\mathbf{a}^\kappa\|$,
3. $\|\mathbf{a}^\kappa\| = \|\mathbf{a}\|$,
4. *the number of rows of $\mathbf{a}^\kappa$ is q and the number of columns is bounded from above by q^2.*

Proof Claims 1 and 3 are trivial. Claim 2 follows from the fact that all columns in $\mathbf{a}^\kappa$ have at most two non-zero entries. For Claim 4 we reason as follows. The vector $\mathbf{a}$ is partitioned into three sets $\{a_i > 0\}$, $\{a_i = 0\}$, and $\{a_i < 0\}$ with cardinalities c, d, and e. We have $q = c+d+e$ and by construction $\mathbf{a}^\kappa$ has $c+d+ce$ columns. Let $\div$ denote integer division. One easily verifies that $c = (q+1) \div 2$, $d = 0$, and $e = q \div 2$ maximizes this number. Since $((q+1) \div 2)((q+2) \div 2) \leqslant q^2$, this gives the desired result. □

Lemma 4 *If A is an $m \times n$ matrix then S_i^A is an $n \times k$ matrix for some $k \leqslant n^{2^i}$.*

Proof Straightforward induction on i. □

Lemma 5 $\|S_i^A\| \leqslant (2n\|A\|)^{2^i-1}$.

Proof We perform induction on i. The base case is trivial, because S_0^A is the identity matrix and thus $\|S_0^A\| = 1 = (2n\|A\|)^{2^0-1}$. We show the inductive step. Since $S_{i+1}^A = S_i^A(\mathbf{a}_{i+1}S_i^A)^\kappa$, we get $\|S_{i+1}^A\| \leqslant 2\|S_i^A\|\|(\mathbf{a}_{i+1}S_i^A)^\kappa\| = 2\|S_i^A\|\|\mathbf{a}_{i+1}S_i^A\| \leqslant 2n\|\mathbf{a}_{i+1}\|\|S_i^A\|^2 \leqslant 2n\|A\|\|S_i^A\|^2 \leqslant 2n\|A\|((2n\|A\|)^{2^i-1})^2 = (2n\|A\|)^{2^{i+1}-1}$. Here we used the (in)equalities from Lemma 3, the trivial observation $\|\mathbf{a}_{i+1}\| \leqslant \|A\|$ in the fourth step, and the induction hypothesis in the fifth step. □

Now Lemma 2 is an easy consequence of Proposition 1 and Lemmata 4 and 5. Considering that $\mathbf{s}^A$ is an integer vector whenever A is an integer matrix, we obtain a finite characterization of KBO orientability.

Theorem 1 *Termination of $\mathcal{R}$ can be shown by KBO if and only if termination of $\mathcal{R}$ can be shown by KBO whose weights belong to $\{0, 1, \ldots, B_\mathcal{R}\}$.* □

We conclude this section by showing that a principal solution of A can be computed in polynomial time and mention its consequences.

Lemma 6 *Let $\mathbf{s}_i$ ($1 \leqslant i \leqslant m$) be a solution of $A\mathbf{x} \geqslant \mathbf{e}_i$ if such a solution exists and $\mathbf{s}_i = \mathbf{0}$ otherwise. Then $\mathbf{s}_1 + \cdots + \mathbf{s}_m$ is a principal solution of A.*

Proof Straightforward. □

Therefore finding a principal solution of A boils down to solving $A\mathbf{x} \geqslant \mathbf{e}_i$ for $1 \leqslant i \leqslant m$. The latter can be handled in polynomial time due to the following known result.

Proposition 2 ([24, 23]) *$A\mathbf{x} \geqslant \mathbf{b}$ can be solved in polynomial time.* □

Note that a (possibly rational) solution $\mathbf{s}$ satisfying $A\mathbf{s} \geqslant \mathbf{e}_i$ can be transformed into the desired integer solution by multiplication with a sufficiently large scalar since $\mathbf{e}_i \geqslant \mathbf{0}$.

Hence [7] can solve KBO in polynomial time (if MCD is replaced by linear programming) due to a similar argumentation as in [28]: The algorithm performs polynomially many steps, all matrices A that might appear during the algorithm are of polynomial size (cf. the discussion before Definition 2), a principal solution of A can be computed in polynomial time, and testing a finite precedence for well-foundedness (by computing its transitive closure and testing for irreflexivity) is polynomial.

4 A Pure SAT Encoding of KBO

In order to give a propositional encoding of KBO orientability, we must take care of representing a precedence and a weight function. For the former we introduce two sets of propositional variables $X = \{X_{fg} \mid f,g \in \mathcal{F}\}$ and $Y = \{Y_{fg} \mid f,g \in \mathcal{F}\}$ depending on the underlying signature $\mathcal{F}$ [29, 3]. The intended semantics of these variables is that an assignment which satisfies a variable X_{fg} corresponds to a precedence with $f \succ g$ and similarly Y_{fg} suggests $f \sim g$. When dealing with strict precedences it is safe to assign false to all Y_{fg} variables. For the weight function, weights of function symbols are represented by numbers in binary representation and the operations $>$, $=$, $\geqslant$, and $+$ must be redefined accordingly. The propositional encodings of $>$ and $=$ given below are similar to the ones in [3] (apart from some slight optimizations). To save parentheses we employ the binding hierarchy for the connectives where $+$ binds strongest, followed by the relation symbols $>$, $=$, and $\geqslant$. The logical connective $\neg$ is next in the hierarchy, followed by $\vee$ and $\wedge$. The operators $\rightarrow$ and $\leftrightarrow$ bind weakest.

We fix the number k of bits that is available for representing natural numbers in binary. Let $a < 2^k$. We denote by $\mathbf{a}_k = \langle a_k, \ldots, a_1 \rangle$ the binary representation of a where a_k is the most significant bit. Whenever k is not essential we abbreviate $\mathbf{a}_k$ to $\mathbf{a}$.

Definition 3 For natural numbers given in binary representation, the operations $>$, $=$, and $\geqslant$ are defined as follows (for all $1 \leqslant j \leqslant k$):

$$\begin{aligned}
\mathbf{f} >_j \mathbf{g} &= \begin{cases} f_1 \wedge \neg g_1 & \text{if } j = 1 \\ (f_j \wedge \neg g_j) \vee \big((g_j \rightarrow f_j) \wedge \mathbf{f} >_{j-1} \mathbf{g}\big) & \text{if } j > 1 \end{cases} \\
\mathbf{f} > \mathbf{g} &= \mathbf{f} >_k \mathbf{g} \\
\mathbf{f} = \mathbf{g} &= \bigwedge_{i=1}^{k} (f_i \leftrightarrow g_i) \\
\mathbf{f} \geqslant \mathbf{g} &= \mathbf{f} > \mathbf{g} \vee \mathbf{f} = \mathbf{g}
\end{aligned}$$

Next we define a formula which is satisfiable if and only if the encoded weight function is admissible for the encoded precedence.

Definition 4 For a weight function (w, w_0), let $\mathsf{ADM\text{-}SAT}_k(w, w_0)$ be the formula

$$(\mathbf{w_0})_k > \mathbf{0}_k \wedge \bigwedge_{c \in \mathcal{F}^{(0)}} \mathbf{c}_k \geqslant (\mathbf{w_0})_k \wedge \bigwedge_{f \in \mathcal{F}^{(1)}} \big(\mathbf{f}_k = \mathbf{0}_k \rightarrow \bigwedge_{g \in \mathcal{F}} (X_{fg} \vee Y_{fg})\big)$$

For addition one has to take overflows into account. Since two k-bit integers might sum up to a $k+1$-bit number an additional bit is needed for the result. Consequently the case arises when two summands are not of equal bit width. Thus, before adding $\mathbf{a}_k$ and $\mathbf{b}_{k'}$ the shorter one is padded with $|k - k'|$ zeros. To keep the presentation simple we assume that zero-padding is implicitly performed before the operations $+$, $>$, $\geqslant$, and $=$. To carry out addition we employ pairs. The first component represents the bit representation and the second component is a propositional formula which encodes the constraints for each bit.

Definition 5 We define $(\mathbf{f}_k, \varphi) + (\mathbf{g}_k, \psi)$ as $(\mathbf{s}_{k+1}, \varphi \wedge \psi \wedge \gamma \wedge \sigma)$ with

$$\gamma = \neg c_0 \wedge \bigwedge_{i=1}^{k} \big(c_i \leftrightarrow (f_i \wedge g_i) \vee (f_i \wedge c_{i-1}) \vee (g_i \wedge c_{i-1})\big)$$

and

$$\sigma \;=\; (s_{k+1} \leftrightarrow c_k) \wedge \bigwedge_{i=1}^{k} \big(s_i \leftrightarrow (f_i \oplus g_i \oplus c_{i-1})\big)$$

where c_i $(0 \leqslant i \leqslant k)$ and s_i $(1 \leqslant i \leqslant k+1)$ are fresh variables that represent the carry and the sum of the addition and $\oplus$ denotes exclusive or.

Note that although theoretically not necessary, it is a good idea to introduce new variables for the sum. The reason is that in consecutive additions each bit f_i and g_i is duplicated (twice for the carry and once for the sum) and consequently using fresh variables for the sum prevents an exponential blowup of the resulting formula. A further method to keep formulas small is to give an upper bound on the bit width when representing naturals. This can be accomplished after addition by fixing a maximal number m of bits and transforming $(\mathbf{s}_k, \varphi)$ into

$$(\mathbf{s}_m, \varphi \wedge \bigwedge_{i=m+1}^{k} \neg s_i)$$

which just demands that all overflow bits must be zero.

Definition 6 We define $(\mathbf{f}, \varphi) \circ (\mathbf{g}, \psi)$ as $\mathbf{f} \circ \mathbf{g} \wedge \varphi \wedge \psi$ for $\circ \in \{>, \geqslant, =\}$.

In the next definition we show how the weight of terms is computed propositionally.

Definition 7 Let t be a term and (w, w_0) a weight function. The weight of a term is encoded as follows:

$$W_k^t \;=\; \begin{cases} ((\mathbf{w_0})_k, \top) & \text{if } t \in \mathcal{V}, \\ (\mathbf{f}_k, \top) + \sum_{i=1}^{n} W_k^{t_i} & \text{if } t = f(t_1, \ldots, t_n). \end{cases}$$

We are now ready to define a propositional formula that reflects the definition of $>_{\text{kbo}}$.

Definition 8 Let s and t be terms. We define the formula $\mathsf{SAT}_k(s >_{\text{kbo}} t)$ as follows. If $s \in \mathcal{V}$ or $s = t$ or $|s|_x < |t|_x$ for some $x \in \mathcal{V}$ then $\mathsf{SAT}_k(s >_{\text{kbo}} t) = \bot$. Otherwise

$$\mathsf{SAT}_k(s >_{\text{kbo}} t) \;=\; W_k^s > W_k^t \vee \big(W_k^s = W_k^t \wedge \mathsf{SAT}_k(s >'_{\text{kbo}} t)\big)$$

with

$$\mathsf{SAT}_k(s >'_{\text{kbo}} t) = \begin{cases} \top & \text{if } t \in \mathcal{V},\ s \in \mathcal{T}(\mathcal{F}^{(1)}, \{t\}), \text{ and } s \neq t \\ X_{fg} \vee \big(Y_{fg} \wedge \mathsf{SAT}_k(s_i >_{\text{kbo}} t_i)\big) & \text{if } s = f(s_1, \ldots, s_n),\ t = g(t_1, \ldots, t_m) \end{cases}$$

where in the second clause i denotes the least $1 \leqslant j \leqslant \min\{n, m\}$ with $s_j \neq t_j$. Here the first case corresponds to (b)(1) in the definition of KBO, the constraint X_{fg} to (b)(3) and the variable Y_{fg} together with the recursive call to (b)(2).

To ensure the properties of a precedence we follow the approach of Codish *et al.* [3] who propose to interpret function symbols as natural numbers. The "greater than or equal to" relation then ensures that the function symbols are quasi-ordered. Let $|\mathcal{F}| = n$. We are looking for a mapping $m\colon \mathcal{F} \to \{1, \ldots, n\}$ such that for every satisfied propositional variable $X_{fg} \in X$ we have $m(f) > m(g)$ and $Y_{fg} \in Y$ implies $m(f) = m(g)$. To uniquely encode one of the n function symbols, $l := \lceil \log_2(n) \rceil$ fresh propositional variables are needed. The l-bit representation of f is $\mathbf{f}' = \langle f'_l, \ldots, f'_1 \rangle$ with f'_l the most significant bit. Note that the variables f'_i $(1 \leqslant i \leqslant l)$ are different from f_i $(1 \leqslant i \leqslant k)$ which are used to represent weights.

Definition 9 Let $\mathcal{R}$ be a TRS. The formula $\mathsf{KBO\text{-}SAT}_k(\mathcal{R})$ is defined as

$$\mathsf{ADM\text{-}SAT}_k(w,w_0) \wedge \bigwedge_{l\to r\in\mathcal{R}} \mathsf{SAT}_k(l >_{\mathrm{kbo}} r) \wedge \bigwedge_{f,g\in\mathcal{F}} (X_{fg} \to \mathbf{f}' > \mathbf{g}') \wedge (Y_{fg} \to \mathbf{f}' = \mathbf{g}')$$

Theorem 2 *Termination of $\mathcal{R}$ can be shown by KBO whenever the propositional formula $\mathsf{KBO\text{-}SAT}_k(\mathcal{R})$ is satisfiable.* □

According to Theorem 1, the reverse does hold for all $k \geqslant \lceil \log_2(B_{\mathcal{R}}+1)\rceil$. However, in Section 8 we will see that in practice rather small weights suffice.

5 A Pseudo-Boolean Encoding of KBO

A *pseudo-boolean constraint* (PBC) is of the form

$$\big(\sum_{i=1}^{n} a_i * x_i\big) \circ m$$

where $a_1,\ldots,a_n,m$ are fixed integers, $x_1,\ldots,x_n$ boolean variables that range over $\{0,1\}$, and $\circ \in \{\geqslant,=,\leqslant\}$. We separate PBCs that are written on a single line by semicolons. A sequence of PBCs is satisfiable if there exists an assignment which satisfies every PBC in the sequence. This means that PB can easily encode conjunctions of linear arithmetic expressions whereas disjunctions are tricky. In the sequel we show that nevertheless PBCs allow to encode KBO concisely. Since 2005 pseudo-boolean evaluation[4] is a track of the international SAT competition.[5] The next definition captures the admissibility condition of a weight function for a signature $\mathcal{F}$.

Definition 10 For a weight function (w,w_0) let $\mathsf{ADM\text{-}PB}_k(w,w_0)$ be the collection of PBCs

- $\overline{w_0}_k \geqslant 1$
- $\overline{w}_k(c) - \overline{w_0}_k \geqslant 0$ for all $c \in \mathcal{F}^{(0)}$
- $n * \overline{w}_k(f) + \sum_{f,g\in\mathcal{F}}(X_{fg}+Y_{fg}) \geqslant n$ for all $f \in \mathcal{F}^{(1)}$

where $n = |\mathcal{F}|$, $\overline{w}_k(f) = 2^{k-1} * f_k + \cdots + 2^0 * f_1$ denotes the weight of f in $\mathbb{N}$ using k bits, and $\overline{w_0}_k$ denotes the value of $(\mathbf{w_0})_k$.

In the definition above the first two PBCs express that w_0 is strictly larger than zero and that every constant has weight at least w_0. Whenever the considered function symbol f has weight larger than zero the third constraint is trivially satisfied. In the case that the unary function symbol f has weight zero the constraints on the precedence add up to n if and only if f is a maximal element. Note that X_{fg} and Y_{fg} are mutual exclusive (which is ensured when encoding the constraints on a quasi-precedence, cf. Definition 12).

For the encoding of $s >_{\mathrm{kbo}} t$ and $s >'_{\mathrm{kbo}} t$ (case (b) in Definition 1) auxiliary propositional variables $KBO_k(s,t)$ and $KBO'_k(s,t)$ are introduced. The intended meaning is that if $KBO_k(s,t)$ ($KBO'_k(s,t)$) evaluates to true under a satisfying assignment then $s >_{\mathrm{kbo}} t$ ($s >'_{\mathrm{kbo}} t$). The general idea of the encoding is very similar to the pure SAT case. As we do not know anything about weights and the precedence at the time of encoding we have to

[4] `http://www.cril.univ-artois.fr/PB07/`

[5] `http://www.satcompetition.org/`

consider the cases $w(s) > w(t)$ and $w(s) = w(t)$ at the same time. That is why $KBO'_k(s,t)$ and the recursive call to $\mathsf{PB}_k(s >'_{\text{kbo}} t)$ must be considered in any case.

The weight $w(t)$ of a term t is defined similarly as in Section 2 with the only difference that the weight $w(f)$ of the function symbol $f \in \mathcal{F}$ is represented in k bits as described in Definition 10.

Definition 11 Let s, t be terms. The encoding of $\mathsf{PB}_k(s >_{\text{kbo}} t)$ amounts to $KBO_k(s,t) = 0$ if $s \in \mathcal{V}$ or $s = t$ or $|s|_x < |t|_x$ for some $x \in \mathcal{V}$. In all other cases $\mathsf{PB}_k(s >_{\text{kbo}} t)$ is

$$-(m+1) * KBO_k(s,t) + w(s) - w(t) + KBO'_k(s,t) \geqslant -m;\ \mathsf{PB}_k(s >'_{\text{kbo}} t) \tag{1}$$

where $m = 2^k * |t|$. Here $\mathsf{PB}_k(s >'_{\text{kbo}} t)$ is the empty constraint when $t \in \mathcal{V}$, $s \in \mathcal{T}(\mathcal{F}^{(1)}, \{t\})$, and $s \neq t$. In the remaining case $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, and $\mathsf{PB}_k(s >'_{\text{kbo}} t)$ is the combination of $\mathsf{PB}_k(s_i >_{\text{kbo}} t_i)$ and

$$-2 * KBO'_k(s,t) + 2 * X_{fg} + Y_{fg} + KBO_k(s_i,t_i) \geqslant 0$$

where i denotes the least $1 \leqslant j \leqslant \min\{n,m\}$ with $s_i \neq t_i$.

Since the encoding of $\mathsf{PB}_k(s >_{\text{kbo}} t)$ is explained in the example below here we just explain the intended semantics of $\mathsf{PB}_k(s >'_{\text{kbo}} t)$. In the first case where t is a variable there are no constraints on the weights and the precedence which means that the empty constraint is returned. In the other case the constraint expresses that whenever $KBO'_k(s,t)$ is satisfied then either $f \succ g$ or both $f \sim g$ and $KBO_k(s_i,t_i)$ must hold.

To get familiar with the encoding and to see why the definitions are a bit tricky consider the example below. For reasons of readability symbols occurring both in s and in t are removed immediately. This entails that the multiplication factor m should be lowered to

$$m = \sum_{a \in \mathcal{F} \cup \mathcal{V}} \max\{0, 2^k * (|t|_a - |s|_a)\},$$

which again is a lower bound on the left-hand side of constraint (1) if $KBO_k(s,t)$ is false because $w(s) - w(t) \geqslant -m$.

Example 5 Consider the TRS consisting of the rule

$$s = \mathsf{f}(\mathsf{g}(x), \mathsf{g}(\mathsf{g}(x))) \to \mathsf{f}(\mathsf{g}(\mathsf{g}(x)), x) = t$$

The PB encoding $\mathsf{PB}_k(s >_{\text{kbo}} t)$ then looks as follows:

$$-KBO_k(s,t) + w(\mathsf{g}) + KBO'_k(s,t) \geqslant 0 \tag{2}$$

$$-2 * KBO'_k(s,t) + 2 * X_{\mathsf{ff}} + Y_{\mathsf{ff}} + KBO_k(\mathsf{g}(x), \mathsf{g}(\mathsf{g}(x))) \geqslant 0 \tag{3}$$

$$-(2^k+1) * KBO_k(\mathsf{g}(x), \mathsf{g}(\mathsf{g}(x))) - w(\mathsf{g}) + KBO'_k(\mathsf{g}(x), \mathsf{g}(\mathsf{g}(x))) \geqslant -2^k \tag{4}$$

$$-2 * KBO'_k(\mathsf{g}(x), \mathsf{g}(\mathsf{g}(x))) + 2 * X_{\mathsf{gg}} + Y_{\mathsf{gg}} + KBO_k(x, \mathsf{g}(x)) \geqslant 0 \tag{5}$$

$$KBO_k(x, \mathsf{g}(x)) = 0 \tag{6}$$

Constraint (2) states that if $s >_{\text{kbo}} t$ then either $w(\mathsf{g}) > 0$ or $s >'_{\text{kbo}} t$. Note that here the multiplication factor m is 0. Clearly the attentive reader would assign $w(\mathsf{g}) = 1$ and termination of the TRS is shown. The encoding however is not so smart and performs the full recursive translation to PB. In (4) it is not possible to satisfy $s_1 = \mathsf{g}(x) >_{\text{kbo}} \mathsf{g}(\mathsf{g}(x)) = t_1$ since the former is embedded in the latter. Nevertheless the constraint (4) must remain satisfiable because the TRS is KBO orientable. The trick is to introduce a hidden case analysis. The

multiplication factor in front of the $KBO_k(s_1,t_1)$ variable does that job. Whenever $s_1 >_{\text{kbo}} t_1$ is needed then $KBO_k(s_1,t_1)$ must evaluate to true. Then implicitly the constraint demands that $w(s_1) > w(t_1)$ or $w(s_1) = w(t_1)$ and $s_1 >'_{\text{kbo}} t_1$ which reflects the definition of KBO. If $s_1 >_{\text{kbo}} t_1$ need not be satisfied (e.g., because already $s >_{\text{kbo}} t$ in (2)) then the constraint holds in any case since the left-hand side in (4) never becomes smaller than -2^k because $w(\mathrm{g}) < 2^k$.

To encode a precedence in PB we again interpret function symbols in $\mathbb{N}$. For this approach an additional set of propositional variables $Z = \{Z_{fg} \mid f,g \in \mathcal{F}\}$ is used. The intended semantics is that Z_{fg} evaluates to true whenever $g \succ f$ or f and g are incomparable. We remark that the Z_{fg} variables are not necessary as far as termination proving power is concerned (because total precedences suffice as KBO is incremental with respect to the precedence) but they are essential to encode partial precedences which are sometimes handy (as explained in Section 9).

Definition 12 For a signature $\mathcal{F}$ we define $\mathsf{PREC\text{-}PB}(\mathcal{F})$ using the PBCs below. Let $l = \lceil \log_2(|\mathcal{F}|) \rceil$. For all $f,g \in \mathcal{F}$

$$\begin{aligned} 2 * X_{fg} + Y_{fg} + Y_{gf} + 2 * Z_{fg} &= 2 \\ -X_{fg} + 2^l * Y_{fg} + 2^l * Z_{fg} + \overline{p}(f) - \overline{p}(g) &\geqslant 0 \\ 2^l * X_{fg} + Y_{fg} + 2^l * Z_{fg} + \overline{p}(f) - \overline{p}(g) &\geqslant 1 \end{aligned}$$

where $\overline{p}(f) = 2^{l-1} * f'_l + \cdots + 2^0 * f'_1$ denotes the position of f in the precedence by interpreting f in $\mathbb{N}$ using l bits.

The above definition expresses all requirements of a quasi-precedence. The symmetry of $\sim$ and the mutual exclusion of the X, Y, and Z variables is mimicked by the first constraint. The second constraint encodes the conditions that are put on the X variables. Whenever a system needs $f \succ g$ in the precedence to be terminating then X_{fg} must evaluate to true and (because they are mutually exclusive) Y_{fg} and Z_{fg} to false. Hence in order to remain satisfiable $\overline{p}(f) > \overline{p}(g)$ must hold. In a case where $f \succ g$ is not needed (but the TRS is KBO orientable) the constraint must remain satisfiable. Thus Y_{fg} or Z_{fg} evaluate to one and because $\overline{p}(g)$ is bound by $2^l - 1$ the constraint does no harm. Summing up, the second constraint encodes a proper order on the symbols in $\mathcal{F}$. The third constraint forms an equivalence relation on $\mathcal{F}$ using the Y_{fg} variables. Whenever $f \sim g$ is demanded somewhere in the encoding, then X_{fg} and Z_{fg} evaluate to false by the first constraint. Satisfiability of the third constraint implies $\overline{p}(f) \geqslant \overline{p}(g)$ but at the same time symmetry demands that Y_{gf} also evaluates to true which leads to $\overline{p}(g) \geqslant \overline{p}(f)$ and thus to $\overline{p}(f) = \overline{p}(g)$.

The next definition expresses KBO in PB. The constraint $\mathrm{PB}_k(s >_{\text{kbo}} t)$ demands that if $KBO_k(s,t) = 1$ then $s >_{\text{kbo}} t$. To ensure KBO orientation, for every rule $l \to r$ the constraint $KBO_k(l,r) = 1$ is added. Note that without these additional constraints, the encoding would always be satisfiable, so also for TRSs that are not KBO terminating.

Definition 13 Let $\mathcal{R}$ be a TRS. The pseudo-boolean encoding $\mathsf{KBO\text{-}PB}_k(\mathcal{R})$ is defined as the combination of $\mathsf{ADM\text{-}PB}_k(w,w_0)$, $\mathsf{PREC\text{-}PB}(\mathcal{F})$, and

$$\mathrm{PB}_k(l >_{\text{kbo}} r);\ KBO_k(l,r) = 1$$

for all $l \to r \in \mathcal{R}$.

Theorem 3 *Termination of $\mathcal{R}$ can be shown by KBO whenever the PBCs $\mathsf{KBO\text{-}PB}_k(\mathcal{R})$ are satisfiable.* □

Again the reverse holds for all $k \geqslant \lceil \log_2(B_{\mathcal{R}} + 1) \rceil$ (cf. Theorem 1).

6 An SMT Encoding of KBO

Due to the rich input format for SMT solvers one can directly translate orientability of KBO into constraints of linear arithmetic. A further advantage is that in this setting integer (and even real) variables are allowed and consequently encoding integers in binary becomes superfluous. This fact also allows us to get rid of parametrizing the formula by the number of bits used to represent natural numbers.

Thus we employ for every function symbol $f \in \mathcal{F}$ non-negative integer variables w_f and p_f indicating the weight of f and its position in the precedence, respectively. Consequently an assignment satisfying $\mathsf{p}_f > \mathsf{p}_g$ indicates $f \succ g$. Together with an integer variable w_0 the definition of $\mathsf{ADM\text{-}SMT}(w, w_0)$ becomes trivial as can be seen in the next definition.

Definition 14 For a weight function (w, w_0), let $\mathsf{ADM\text{-}SMT}(w, w_0)$ be the constraint

$$\mathsf{w}_0 > 0 \;\wedge \bigwedge_{c \in \mathcal{F}^{(0)}} \mathsf{w}_c \geqslant \mathsf{w}_0 \;\wedge \bigwedge_{f \in \mathcal{F}^{(1)}} \big(\mathsf{w}_f = 0 \;\rightarrow \bigwedge_{g \in \mathcal{F}} (\mathsf{p}_f \geqslant \mathsf{p}_g)\big)$$

Similarly, computing the weight of a term simplifies tremendously.

Definition 15 Let t be a term and (w, w_0) a weight function. The weight of a term is encoded as follows:

$$W_t \;=\; \begin{cases} \mathsf{w}_0 & \text{if } t \in \mathcal{V}, \\ \mathsf{w}_f + \sum_{i=1}^{n} W_{t_i} & \text{if } t = f(t_1, \ldots, t_n). \end{cases}$$

We are now ready to define an SMT formula that reflects the definition of $>_{\text{kbo}}$.

Definition 16 Let s and t be terms. We define the formula $\mathsf{SMT}(s >_{\text{kbo}} t)$ as follows. If $s \in \mathcal{V}$ or $s = t$ or $|s|_x < |t|_x$ for some $x \in \mathcal{V}$ then $\mathsf{SMT}(s >_{\text{kbo}} t) = \bot$. Otherwise

$$\mathsf{SMT}(s >_{\text{kbo}} t) \;=\; W_s > W_t \vee \big(W_s = W_t \wedge \mathsf{SMT}(s >'_{\text{kbo}} t)\big)$$

with $\mathsf{SMT}(s >'_{\text{kbo}} t) = \top$ if $t \in \mathcal{V}$, $s \in \mathcal{T}(\mathcal{F}^{(1)}, \{t\})$, and $s \neq t$, and

$$\mathsf{SMT}(s >'_{\text{kbo}} t) \;=\; \mathsf{p}_f > \mathsf{p}_g \vee \big(\mathsf{p}_f = \mathsf{p}_g \wedge \mathsf{SMT}(s_i >_{\text{kbo}} t_i)\big)$$

if $s = f(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_m)$ where i denotes the least $1 \leqslant j \leqslant \min\{n, m\}$ with $s_j \neq t_j$.

Due to the richer input format the X_{fg} and Y_{fg} variables for abbreviating precedence comparisons are superfluous which allows to present the full SMT encoding for KBO concisely as follows.

Definition 17 Let $\mathcal{R}$ be a TRS. The formula $\mathsf{KBO\text{-}SMT}(\mathcal{R})$ is defined as

$$\mathsf{ADM\text{-}SMT}(w, w_0) \;\wedge \bigwedge_{l \to r \in \mathcal{R}} \mathsf{SMT}(l >_{\text{kbo}} r)$$

Theorem 4 *Termination of $\mathcal{R}$ can be shown by KBO if and only if the SMT constraint* $\mathsf{KBO\text{-}SMT}(\mathcal{R})$ *is satisfiable.* □

Note that the SMT approach is always complete and hence is not parametrized by any constant k indicating the number of bits for weights, in contrast to Theorems 2 and 3.

7 Extensions

One obvious and powerful extension of KBO is to integrate it in the dependency pair method [1, 17, 19, 20]. We shortly recapitulate its key features that are essential for a proper understanding of this section. Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. The defined symbols are the root symbols of the left-hand sides of the rewrite rules in $\mathcal{R}$. The original signature $\mathcal{F}$ is extended to a signature $\mathcal{F}^\sharp$ by adding for every defined symbol f a fresh symbol $f^\sharp$ with the same arity as f. For a term $t = f(t_1,\dots,t_n)$ with defined symbol f we denote $f^\sharp(t_1,\dots,t_n)$ by $t^\sharp$. In examples one often uses capitalization, i.e., one writes F for $f^\sharp$. If $l \to r \in \mathcal{R}$ and t is a subterm of r with defined root symbol, then the rule $l^\sharp \to t^\sharp$ is a *dependency pair* of $\mathcal{R}$. We write $\mathsf{DP}(\mathcal{R})$ for the set of all dependency pairs of $\mathcal{R}$. The nodes of the *dependency graph* $\mathsf{DG}(\mathcal{R})$ are the dependency pairs of $\mathcal{R}$ and there is an edge from node $s \to t$ to node $u \to v$ if there exist substitutions σ and τ such that $t\sigma \to_\mathcal{R}^* u\tau$. An *argument filtering* for a signature $\mathcal{F}$ is a mapping π that assigns to every n-ary function symbol $f \in \mathcal{F}$ an argument position $i \in \{1,\dots,n\}$ or a (possibly empty) list $[i_1,\dots,i_m]$ of argument positions with $1 \leqslant i_1 < \cdots < i_m \leqslant n$. The signature $\mathcal{F}_\pi$ consists of all function symbols f such that $\pi(f)$ is some list $[i_1,\dots,i_m]$, where in $\mathcal{F}_\pi$ the arity of f is m. Every argument filtering π induces a mapping from $\mathcal{T}(\mathcal{F},\mathcal{V})$ to $\mathcal{T}(\mathcal{F}_\pi,\mathcal{V})$, also denoted by π: $\pi(t) = t$ if $t \in \mathcal{V}$, $\pi(t) = \pi(t_i)$ if $t = f(t_1,\dots,t_n)$ with $\pi(f) = i$, and $\pi(t) = f(\pi(t_{i_1}),\dots,\pi(t_{i_m}))$ if $t = f(t_1,\dots,t_n)$ with $\pi(f) = [i_1,\dots,i_m]$. We further abuse notation and define $\pi(l \to r)$ for rewrite rules and $\pi(\mathcal{R})$ for TRSs in an obvious manner. The *usable rules* for a set $\mathcal{C}$ of dependency pairs are denoted by $\mathcal{U}(\mathcal{C})$ where a rule $f(\dots) \to r \in \mathcal{R}$ is usable if f occurs in the right-hand side of a rule in $\mathcal{C}$ or in $\mathcal{U}(\mathcal{C})$. Argument filterings can be used to reduce the number of usable rules (see [19]), i.e., $\mathcal{U}(\mathcal{C},\pi)$ computes the usable rules of $\pi(\mathcal{C})$ on the basis of $\pi(\mathcal{R})$. A *reduction pair* $(\gtrsim,>)$ satisfies that $\gtrsim$ is reflexive, transitive, closed under contexts and substitutions, $>$ is a well-founded order closed under substitutions, and additionally the inclusion $\gtrsim \cdot > \cdot \gtrsim \subseteq >$ holds. If reduction pairs are combined with usable rules, additionally *$\mathcal{C}_\mathcal{E}$-compatibility* [19, 21] must be ensured, i.e., for a fresh function symbol g the constraints $\mathrm{g}(x,y) \gtrsim x$ and $\mathrm{g}(x,y) \gtrsim y$ must hold.

In this article we reformulate the main theorem as a satisfiability problem in propositional logic for specific reduction pairs. In Subsection 7.2 we address the embedding order and in Subsection 7.3 we address KBO, but first (Subsection 7.1) we explain how to represent argument filterings in propositional logic. The encoding for SMT is then a straightforward adaption of the pure propositional logic encoding. We note that for PB this approach does not seem so suitable since the resulting formula contains many disjunctions. The main motivation for using pseudo-boolean in the direct encoding (Section 5) was that it allows a concise implementation because the KBO constraints can easily be expressed in PB. This is no longer true when combining KBO with dependency pairs. One could of course perform a Tseitin-like [38] transformation to PB but that would destroy the elegance of the approach. Why PB can still be advantageous is outlined in Section 9. The rest of this section aims at formulating the theorem below in propositional logic.

Theorem 5 ([19]) *A TRS $\mathcal{R}$ is terminating if and only if for every cycle $\mathcal{C}$ in the dependency graph of $\mathcal{R}$ there exist an argument filtering π and a $\mathcal{C}_\mathcal{E}$-compatible reduction pair $(\gtrsim,>)$ such that $\pi(\mathcal{U}(\mathcal{C},\pi) \cup \mathcal{C}) \subseteq \gtrsim$ and $\pi(\mathcal{C}) \cap > \neq \varnothing$.* □

Before looking closer into the encoding, we demonstrate a termination proof with dependency pairs by means of the following example.

Example 6 The TRS AG01/#3.1

$$\begin{aligned}
\mathsf{minus}(x,\mathsf{0}) &\to x\\
\mathsf{minus}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{minus}(x,y)\\
\mathsf{quot}(\mathsf{0},\mathsf{s}(y)) &\to \mathsf{0}\\
\mathsf{quot}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{s}(\mathsf{quot}(\mathsf{minus}(x,y),\mathsf{s}(y)))
\end{aligned}$$

gives rise to the dependency pairs

$$\mathsf{Minus}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{Minus}(x,y) \tag{7}$$
$$\mathsf{Quot}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{Minus}(x,y)$$
$$\mathsf{Quot}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{Quot}(\mathsf{minus}(x,y),\mathsf{s}(y)) \tag{8}$$

The dependency graph contains the two cycles $\{7\}$ and $\{8\}$. For the first cycle there are no usable rules and rule 7 can easily be handled by KBO. The second cycle is more challenging. To make rule 8 non-duplicating we take an argument filtering satisfying $\pi(\mathsf{Quot}) = \pi(\mathsf{minus}) = 1$ and $\pi(\mathsf{s}) = [1]$, resulting in the rule $\mathsf{s}(x) \to x$ and no usable rules. Now KBO can easily remove this cycle.

7.1 Representing Argument Filterings

Definition 18 Let $\mathcal{F}$ be a signature. The set of propositional variables $\{\pi_f \mid f \in \mathcal{F}\} \cup \{\pi_f^i \mid f \in \mathcal{F} \text{ and } 1 \leqslant i \leqslant \mathrm{arity}(f)\}$ is denoted by $\pi_{\mathcal{F}}$. Let π be an argument filtering for $\mathcal{F}$. The induced assignment α_π is defined as follows:

$$\alpha_\pi(\pi_f) = \begin{cases} 1 & \text{if } \pi(f) = [i_1,\ldots,i_m] \\ 0 & \text{if } \pi(f) = i \end{cases} \quad \text{and} \quad \alpha_\pi(\pi_f^i) = \begin{cases} 1 & \text{if } i \in \pi(f) \\ 0 & \text{if } i \notin \pi(f) \end{cases}$$

for all n-ary function symbols $f \in \mathcal{F}$ and $i \in \{1,\ldots,n\}$. Here $i \in \pi(f)$ if $\pi(f) = i$ or $\pi(f) = [i_1,\ldots,i_m]$ and $i_k = i$ for some $1 \leqslant k \leqslant m$.

Definition 19 An assignment α for $\pi_{\mathcal{F}}$ is said to be *argument filtering consistent* if for every n-ary function symbol $f \in \mathcal{F}$ such that $\alpha \not\vDash \pi_f$ there is a unique $i \in \{1,\ldots,n\}$ such that $\alpha \vDash \pi_f^i$.

It is easy to see that α_π is argument filtering consistent.

Definition 20 The propositional formula $\mathsf{AF}^\pi(\mathcal{F})$ is defined as $\bigwedge_{f\in\mathcal{F}} \mathsf{AF}^\pi(f)$ with

$$\mathsf{AF}^\pi(f) = \pi_f \vee \bigvee_{i=1}^{\mathrm{arity}(f)} \Big(\pi_f^i \wedge \bigwedge_{j\neq i} \neg\pi_f^j\Big).$$

Lemma 7 *An assignment α for $\pi_{\mathcal{F}}$ is argument filtering consistent if and only if $\alpha \vDash \mathsf{AF}^\pi(\mathcal{F})$.* □

Definition 21 Let α be an argument filtering consistent assignment for $\pi_{\mathcal{F}}$. The argument filtering π_α is defined as follows: $\pi_\alpha(f) = [i \mid \alpha \vDash \pi_f^i]$ if $\alpha \vDash \pi_f$ and $\pi_\alpha(f) = i$ if $\alpha \not\vDash \pi_f$ and $\alpha \vDash \pi_f^i$, for all function symbols $f \in \mathcal{F}$.

Example 7 Consider the signature from Example 6. The assignment α only satisfying the variables π_{Minus}, π^2_{Minus}, π_{minus}, π^2_{Quot}, π^1_{quot}, π_{0}, and π_{s} is argument filtering consistent. The induced argument filtering π_α consists of $\pi(\mathsf{Minus}) = [2]$, $\pi(\mathsf{minus}) = \pi(\mathsf{0}) = \pi(\mathsf{s}) = []$, $\pi(\mathsf{Quot}) = 2$, and $\pi(\mathsf{quot}) = 1$.

Corresponding to the definition of $\mathcal{U}(\mathcal{C},\pi)$ we encode $\pi(\mathcal{U}(\mathcal{C},\pi)\cup\mathcal{C}) \subseteq \geqslant$ as the conjunction of

$$\bigwedge_{l\to r\in\mathcal{C}} \left(U_{\mathrm{root}(l)} \wedge l \geqslant^\pi r\right) \wedge \bigwedge_{l\to r\in\mathcal{R}} \left(U_{\mathrm{root}(l)} \to l \geqslant^\pi r\right)$$

and

$$\bigwedge_{l\to r\in\mathcal{R}\cup\mathcal{C}} \left(U_{\mathrm{root}(l)} \to \bigwedge_{\substack{p\in\mathrm{Pos}_{\mathcal{F}}(r)\\ \mathrm{root}(r|_p)\text{ is defined}}} \left(\bigwedge_{q,i\colon iq\leqslant p} \pi^i_{\mathrm{root}(r|_q)} \to U_{\mathrm{root}(r|_p)}\right)\right)$$

Here U_f is a new propositional variable for every defined and every dependency pair symbol f. If U_f evaluates to true, then rules of the form $f(\dots) \to r$ must be oriented. In the formula above the first conjunct expresses that all rules from $\mathcal{C}$ must be oriented by $\geqslant^\pi$. The second conjunct expresses that if a rule is usable, then it must be compatible with $\geqslant^\pi$ whereas the third conjunct computes the usable rules with respect to an argument filtering (if a rule is usable, then defined symbols that survive the argument filtering also give rise to usable rules). The relation $\geqslant^\pi$ can be replaced by an encoding of a reduction pair $\geqslant$ that incorporates argument filterings π. The above formula is abbreviated by $\mathsf{U}(\mathcal{C},\geqslant^\pi)$.

7.2 Embedding

When reformulating Theorem 5 as a satisfaction problem, we have to fix a reduction pair, incorporate argument filterings, and encode the combination in propositional logic. In this section we take the reduction pair $(\trianglerighteq_{\mathsf{emb}}, \triangleright_{\mathsf{emb}})$ corresponding to the embedding order. Because embedding has no parameters it allows for a transparent translation of the constraints $\pi(\mathcal{U}(\mathcal{C},\pi)\cup\mathcal{C}) \subseteq \geqslant$ and $\pi(\mathcal{C}) \cap > \neq \varnothing$ in Theorem 5. In Section 7.3 we consider KBO, which is a bit more challenging.

Definition 22 The *embedding* relation $\trianglelefteq_{\mathsf{emb}}$ is defined on terms as follows: $s \trianglelefteq_{\mathsf{emb}} t$ if $s = t$ or $t = f(t_1,\dots,t_n)$ and either $s \trianglelefteq_{\mathsf{emb}} t_i$ for some i or $s = f(s_1,\dots,s_n)$ and $s_i \trianglelefteq_{\mathsf{emb}} t_i$ for all i. The strict part is denoted by $\triangleleft_{\mathsf{emb}}$. The converse relations are denoted by $\trianglerighteq_{\mathsf{emb}}$ and $\triangleright_{\mathsf{emb}}$.

In the following we define propositional formulas $s \triangleright^\pi_{\mathsf{emb}} t$ and $s \trianglerighteq^\pi_{\mathsf{emb}} t$ which, in conjunction with $\mathsf{AF}^\pi(\mathcal{F})$, represent all argument filterings π that satisfy $\pi_\alpha(s) \triangleright_{\mathsf{emb}} \pi_\alpha(t)$ and $\pi_\alpha(s) \trianglerighteq_{\mathsf{emb}} \pi_\alpha(t)$. We start with defining a formula $s =^\pi t$ that represents all argument filterings which make s and t equal. (In the sequel we assume that $\wedge$ binds stronger than $\vee$.)

Definition 23 Let s and t be terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. We define a propositional formula $s =^\pi t$ by induction on s and t. If $s \in \mathcal{V}$ then

$$s =^\pi t = \begin{cases} \top & \text{if } s = t,\\ \bot & \text{if } t \in \mathcal{V} \text{ and } s \neq t,\\ \neg\pi_g \wedge \bigvee_{j=1}^{m} \left(\pi^j_g \wedge s =^\pi t_j\right) & \text{if } t = g(t_1,\dots,t_m). \end{cases}$$

Let $s = f(s_1,\ldots,s_n)$. If $t \in \mathcal{V}$ then $s =^\pi t = \neg\pi_f \wedge \bigvee_{i=1}^n \big(\pi_f^i \wedge s_i =^\pi t\big)$. If $t = g(t_1,\ldots,t_m)$ with $f \neq g$ then

$$s =^\pi t = \neg\pi_f \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i =^\pi t\big) \vee \neg\pi_g \wedge \bigvee_{j=1}^{m} \big(\pi_g^j \wedge s =^\pi t_j\big).$$

Finally, if $t = f(t_1,\ldots,t_n)$ then

$$s =^\pi t = \neg\pi_f \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i =^\pi t_i\big) \vee \pi_f \wedge \bigwedge_{i=1}^{n} \big(\pi_f^i \to s_i =^\pi t_i\big).$$

To keep readability of the formulas we present a translation related as close to constraints as possible. In an implementation one should minimize the formulas, e.g., the last formula can be expressed more concisely as

$$s =^\pi t = \bigwedge_{i=1}^{n} \big(\pi_f^i \to s_i =^\pi t_i\big)$$

since we know that $\mathsf{AF}^\pi(\mathcal{F})$ must hold anyway.

Definition 24 Let s and t be terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. We define propositional formulas $s \rhd^\pi_{\mathsf{emb}} t$ and $s \trianglerighteq^\pi_{\mathsf{emb}} t = s \rhd^\pi_{\mathsf{emb}} t \vee s =^\pi t$ by induction on s and t. If $s \in \mathcal{V}$ then $s \rhd^\pi_{\mathsf{emb}} t = \bot$. Let $s = f(s_1,\ldots,s_n)$. If $t \in \mathcal{V}$ then

$$s \rhd^\pi_{\mathsf{emb}} t = \pi_f \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i \trianglerighteq^\pi_{\mathsf{emb}} t\big) \vee \neg\pi_f \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i \rhd^\pi_{\mathsf{emb}} t\big).$$

If $t = g(t_1,\ldots,t_m)$ with $f \neq g$ then $s \rhd^\pi_{\mathsf{emb}} t$ is the disjunction of

$$\pi_f \wedge \Big(\pi_g \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i \trianglerighteq^\pi_{\mathsf{emb}} t\big) \vee \neg\pi_g \wedge \bigvee_{j=1}^{m} \big(\pi_g^j \wedge s \rhd^\pi_{\mathsf{emb}} t_j\big)\Big)$$

and $\neg\pi_f \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i \rhd^\pi_{\mathsf{emb}} t\big)$. Finally, if $t = f(t_1,\ldots,t_n)$ then

$$s \rhd^\pi_{\mathsf{emb}} t = \pi_f \wedge \Bigg(\bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i \trianglerighteq^\pi_{\mathsf{emb}} t\big) \vee \bigwedge_{i=1}^{n} \big(\pi_f^i \to s_i \trianglerighteq^\pi_{\mathsf{emb}} t_i\big) \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i \rhd^\pi_{\mathsf{emb}} t_i\big)\Bigg) \vee \neg\pi_f \wedge \bigvee_{i=1}^{n} \big(\pi_f^i \wedge s_i \rhd^\pi_{\mathsf{emb}} t_i\big).$$

The formula $s \rhd^\pi_{\mathsf{emb}} t \wedge \mathsf{AF}^\pi(\mathcal{F})$ is satisfiable if and only if there exists an argument filtering π such that $\pi(s) \rhd_{\mathsf{emb}} \pi(t)$. Even stronger, $s \rhd^\pi_{\mathsf{emb}} t \wedge \mathsf{AF}^\pi(\mathcal{F})$ encodes *all* argument filterings π that satisfy $\pi(s) \rhd_{\mathsf{emb}} \pi(t)$. Analogous statements hold for $s =^\pi t \wedge \mathsf{AF}^\pi(\mathcal{F})$ and $s \trianglerighteq^\pi_{\mathsf{emb}} t \wedge \mathsf{AF}^\pi(\mathcal{F})$.

Lemma 8 *Let $s,t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. If α is an assignment for $\pi_\mathcal{F}$ such that $\alpha \vDash s \rhd^\pi_{\mathsf{emb}} t \wedge \mathsf{AF}^\pi(\mathcal{F})$ then $\pi_\alpha(s) \rhd_{\mathsf{emb}} \pi_\alpha(t)$. If π is an argument filtering such that $\pi(s) \rhd_{\mathsf{emb}} \pi(t)$ then $\alpha_\pi \vDash s \rhd^\pi_{\mathsf{emb}} t \wedge \mathsf{AF}^\pi(\mathcal{F})$.* □

We conclude this section by stating the propositional formulation of the termination criterion of Theorem 5 specialized to embedding.

Theorem 6 *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and let $\mathcal{C}$ be a cycle in the dependency graph of $\mathcal{R}$. The formula*

$$\mathsf{U}(\mathcal{C}, \trianglerighteq^{\pi}_{\mathsf{emb}}) \wedge \bigvee_{l \to r \in \mathcal{C}} l \rhd^{\pi}_{\mathsf{emb}} r \wedge \mathsf{AF}^{\pi}(\mathcal{F})$$

is satisfiable if and only if there exists an argument filtering π such that $\pi(\mathcal{U}(\mathcal{C},\pi) \cup \mathcal{C}) \subseteq \trianglerighteq_{\mathsf{emb}}$ and $\pi(\mathcal{C}) \cap \rhd_{\mathsf{emb}} \neq \varnothing$.[6] □

7.3 Knuth-Bendix Order

Our aim is to define a formula $s >^{\pi}_{\mathrm{kbo}} t \wedge \mathsf{AF}^{\pi}(\mathcal{F}) \wedge \mathsf{PO}(\mathcal{F}) \wedge \mathsf{ADM}^{\pi}(\mathcal{F})$ that is satisfiable if and only if there exist an argument filtering π and a precedence $\succ$ such that $\pi(s) >_{\mathrm{kbo}} \pi(t)$. The conjunct $\mathsf{PO}(\mathcal{F})$ will ensure that the assignment for the variables X_{fg} corresponds to a proper order on the signature. The conjunct $\mathsf{ADM}^{\pi}(\mathcal{F})$ takes care of the admissibility condition.

Below we define the conjunct $s >^{\pi}_{\mathrm{kbo}} t$. The basic idea is to adapt $s \rhd^{\pi}_{\mathsf{emb}} t$ by incorporating the recursive definition of $>_{\mathrm{kbo}}$. First we propose a formula that expresses that after applying the argument filtering no variables are duplicated.

Definition 25 The formula $\mathsf{ND}^{\pi}(s,t)$ is defined as follows:

$$\mathsf{ND}^{\pi}(s,t) = \bigwedge_{x \in \mathcal{V}\mathsf{ar}(t)} |s,\top|_x \geqslant |t,\top|_x$$

with

$$|s,\varphi|_x = \begin{cases} (\langle\varphi\rangle, \top) & \text{if } s = x, \\ (\mathbf{0}, \top) & \text{if } s \in \mathcal{V} \text{ and } s \neq x, \\ \sum_{i=1}^{n} |s_i, \varphi \wedge \pi^i_f|_x & \text{if } s = f(s_1,\ldots,s_n). \end{cases}$$

The idea behind the recursive definition of $|s,\varphi|_x$ is to collect the constraints under which a variable is preserved by the argument filtering. If those constraints are satisfied they correspond to an occurrence of the variable. Adding the constraints yields the number of variables which survive the argument filtering.

Example 8 Consider the rule $l = \mathsf{Minus}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{Minus}(x,y) = r$. Then the formula $\mathsf{ND}^{\pi}(l,r)$ evaluates to $\langle \pi^1_{\mathsf{Minus}} \wedge \pi^1_{\mathsf{s}} \rangle \geqslant \langle \pi^1_{\mathsf{Minus}} \rangle \wedge \langle \pi^2_{\mathsf{Minus}} \wedge \pi^1_{\mathsf{s}} \rangle \geqslant \langle \pi^2_{\mathsf{Minus}} \rangle$ where the first (second) conjunct expresses non-duplication of variable x (y). Informally, the formula expresses that whenever an argument filtering π keeps the first (or second) argument of Minus, then it must also keep the argument of s.

Next we give a formula that computes the weight of a term after an argument filtering has been applied.

Definition 26 We define $w^{\pi}(t)$ as $w'_{\pi}(t,\top)$ with

$$w'_{\pi}(t,\varphi) = \begin{cases} (\varphi \cdot \mathbf{w_0}, \top) & \text{if } t \in \mathcal{V}, \\ ((\pi_f \wedge \varphi) \cdot \mathbf{f}, \top) + \sum_{i=1}^{n} w'_{\pi}(t_i, \pi^i_f \wedge \varphi) & \text{if } t = f(t_1,\ldots,t_n). \end{cases}$$

Here $\psi \cdot \mathbf{g_k}$ abbreviates $\langle \psi \wedge g_k, \ldots, \psi \wedge g_1 \rangle$.

[6] Independently, in [4] a similar encoding is presented for LPO.

Definition 27 Let s and t be terms. We define propositional formulas

$$s >^{\pi}_{\mathrm{kbo}} t = \mathsf{ND}^{\pi}(s,t) \wedge (w^{\pi}(s) > w^{\pi}(t) \vee w^{\pi}(s) = w^{\pi}(t) \wedge s >^{\pi}_{\mathrm{kbo}'} t)$$

and $s \geqslant^{\pi}_{\mathrm{kbo}} t = s >^{\pi}_{\mathrm{kbo}} t \vee s =^{\pi} t$ with $s >^{\pi}_{\mathrm{kbo}'} t$ inductively defined as follows. If $s \in \mathcal{V}$ then $s >^{\pi}_{\mathrm{kbo}'} t = \bot$. Let $s = f(s_1,\ldots,s_n)$. If $t \in \mathcal{V}$ then $s >^{\pi}_{\mathrm{kbo}'} t = s \rhd^{\pi}_{\mathsf{emb}} t$. If $t = g(t_1,\ldots,t_m)$ with $f \neq g$ then

$$s >^{\pi}_{\mathrm{kbo}} t = \pi_f \wedge \pi_g \wedge X_{fg} \vee \neg\pi_g \wedge \bigvee_{j=1}^{m} (\pi^j_g \wedge s >^{\pi}_{\mathrm{kbo}} t_j) \vee \neg\pi_f \wedge \bigvee_{i=1}^{n} (\pi^i_f \wedge s_i >^{\pi}_{\mathrm{kbo}} t).$$

Finally, if $t = f(t_1,\ldots,t_n)$ then

$$s >^{\pi}_{\mathrm{kbo}} t = \pi_f \wedge \langle s_1,\ldots,s_n\rangle >^{\pi,f}_{\mathrm{kbo}} \langle t_1,\ldots,t_n\rangle \vee \neg\pi_f \wedge \bigvee_{i=1}^{n} (\pi^i_f \wedge s_i >^{\pi}_{\mathrm{kbo}} t_i).$$

Here $\langle s_1,\ldots,s_n\rangle >^{\pi,f}_{\mathrm{kbo}} \langle t_1,\ldots,t_n\rangle$ is defined as $\bot$ if $n = 0$ and as

$$\pi^1_f \wedge s_1 >^{\pi}_{\mathrm{kbo}} t_1 \vee (\pi^1_f \to s_1 =^{\pi} t_1) \wedge \langle s_2,\ldots s_n\rangle >^{\pi,f}_{\mathrm{kbo}} \langle t_2,\ldots,t_n\rangle$$

if $n > 0$.

Note that $s >^{\pi}_{\mathrm{kbo}'} t$ corresponds to the definition of $>_{\mathrm{kbo}}$ in the case of equal weights (Definition 1 case (b)). The peculiar looking equation $s >^{\pi}_{\mathrm{kbo}'} t = s \rhd^{\pi}_{\mathsf{emb}} t$ for $t \in \mathcal{V}$ can be explained by the admissibility condition (encoded below) and the fact that $\pi(s)$ and $\pi(t) = t$ are assumed to have equal weight.

Definition 28 The formula $\mathsf{ADM}^{\pi}(\mathcal{F})$ defined below is satisfiable if and only if the weight function is admissible in the presence of an argument filtering.

$$\begin{aligned}\mathbf{w_0} > \mathbf{0} \wedge \bigwedge_{f\in\mathcal{F}} \big(\mathsf{constant}(f) \to \mathbf{f} \geqslant \mathbf{w_0}\big) \wedge \\ \bigwedge_{f\in\mathcal{F}} \big(\mathbf{f} = \mathbf{0} \wedge \mathsf{unary}(f) \to \bigwedge_{g\in\mathcal{F}, f\neq g} (\pi_g \to X_{fg})\big)\end{aligned}$$

with $\mathsf{constant}(f) = \pi_f \wedge \bigwedge_{i=1}^{\mathrm{arity}(f)} \neg\pi^i_f$ and $\mathsf{unary}(f) = \pi_f \wedge \bigvee_{i=1}^{\mathrm{arity}(f)} (\pi^i_f \wedge \bigwedge_{i\neq j} \neg\pi^j_f)$.

Similar as in Section 4 the formula $\mathsf{PO}(\mathcal{F})$ equals $\bigwedge_{f,g\in\mathcal{F}} X_{fg} \to \mathbf{f}' > \mathbf{g}'$. We are now ready to state the propositional encoding of the termination criterion of Theorem 5 specialized to KBO.

Theorem 7 *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and let $\mathcal{C}$ be a cycle in the dependency graph of $\mathcal{R}$. If the formula*

$$\mathsf{U}(\mathcal{C}, \geqslant^{\pi}_{\mathrm{kbo}}) \wedge \bigvee_{l\to r\in\mathcal{C}} l >^{\pi}_{\mathrm{kbo}} r \wedge \mathsf{ADM}^{\pi}(\mathcal{F}) \wedge \mathsf{AF}^{\pi}(\mathcal{F}) \wedge \mathsf{PO}(\mathcal{F})$$

is satisfiable then there are an argument filtering π, a precedence $\succ$, and an admissible weight function (w, w_0) such that $\pi(\mathcal{U}(\mathcal{C},\pi) \cup \mathcal{C}) \subseteq \geqslant_{\mathrm{kbo}}$ and $\pi(\mathcal{C}) \cap >_{\mathrm{kbo}} \neq \varnothing$. □

From a satisfying assignment one can read off the argument filtering, the precedence, and the weight function. We omit the straightforward details. The converse of Theorem 7 holds if we do not put a bound on the number k of bits used for the representation of the weights. This is automatically the case for the SMT back-end. For SAT this is possible due to the results of Section 3.

8 Experimental Results

We implemented our encodings on top of TTT2.[7] MiniSat [10] and MiniSat+ [11] were used to check satisfiability of the SAT and PB based encodings and Yices [8] was the choice for the SMT approach. All three tools are interfaced from TTT2 which is written in OCaml. For the SAT approach propositional formulas are transformed into CNF similar to [34]. For all data given in the following tables addition in SAT (Definition 5) takes overflows into account, i.e., adding two k-bit numbers results in a $(k+1)$-bit number. Below we compare our implementations of KBO, sat, pbc, and smt with the ones of TTT, AProVE [16], and an implementation dkm (also on top of TTT2) as proposed in [7]. TTT and AProVE admit only strict precedences. Both implement the algorithm of Korovin and Voronkov [28] together with techniques of Dick *et al.* [7]. For two of our approaches (sat and pbc) KBO orientability amounts to finding a satisfying assignment for a propositional formula whereas the smt approach is based on linear programming. The other tools find a solution by solving a system of homogeneous linear inequations which also amounts to linear programming. Although this problem is known to be decidable in polynomial time [24,23] in practice algorithms with exponential (worst-case) time complexity such as the simplex method [5] perform much better. Dick *et al.* [7] prefer the method of complete description over simplex due to its support for incrementality. This shows that although computing a KBO for a given TRS can be done in polynomial time none of the existing tools does so.

We used the 1381 TRSs and 724 string rewrite systems (SRSs) of the standard rewriting category in version 4.0 of the Termination Problems Data Base [32]. All tests have been performed on a single core of a server equipped with eight dual-core AMD Opteron® processors 885 running at a clock rate of 2.6GHz and 64GB of main memory with a timeout of 60 seconds.

Concerning optimizations, when computing weights of terms symbols occurring on both sides of rules are ignored. Furthermore the encoding of KBO is only computed if a test for embedding fails. Apart from obvious identities like

$$\varphi \wedge \top \to \varphi \qquad \top \wedge \varphi \to \varphi \qquad \varphi \wedge \bot \to \bot \qquad \bot \wedge \varphi \to \bot \qquad \cdots$$

which help to reduce the size of the encoding no further simplifications are applied. But keeping the encoding in a cache allows to re-use precomputed formulas which drastically reduces encoding time.

8.1 Results for TRSs

As addressed earlier one has to fix the number k of bits which is used to represent natural numbers in binary representation. The actual choice is specified as argument to sat (pbc). Note that a rather small k is sufficient to handle all potential systems from [32] which makes Theorems 2 and 3 powerful in practice. As already indicated in Example 2 there does not exist a uniform upper bound on k but for every given TRS one can compute such a k according to Theorem 1.

The left part of Table 1 summarizes[8] the results for strict precedences. Interestingly, already $k = 4$ suffices to prove the maximum number of systems terminating. The TRS higher-order/AProVE/HO/ReverseLastInit needs weight eight for the constant init and therefore

[7] http://colo6-c703.uibk.ac.at/ttt2/

[8] Full experimental data to be found at http://colo6-c703.uibk.ac.at/ttt2/kbo/.

Table 1 KBO for 1381 TRSs.

	strict precedence			quasi-precedence		
method(#bits)	total time	#successes	#timeouts	total time	#successes	#timeouts
sat/pbc(2)	30.5/90.7	104/104	0/0	34.0/161.3	105/104	0/1
sat/pbc(3)	31.6/93.0	106/106	0/0	34.7/142.6	107/107	0/0
sat/pbc(4)	34.7/99.8	107/107	0/0	37.3/142.2	108/108	0/0
sat/pbc(10)	350.9/187.5	107/107	3/1	377.8/253.2	108/107	3/2
smt_i	26.5	107	0	24.4	108	0
smt_r	26.3	107	0	24.4	108	0
AProVE	1945.2	101	18			
T$_\mathsf{T}$T	329.2	101	1			
T$_\mathsf{T}$T(simplex)	370.0	105	4			
dkm	808.8	99	13			
dkm'	443.9	102	7			

can only be handled by KBO with $k \geqslant 4$. The SMT approach does not need to represent numbers in binary and consequently there are no bit-restrictions on the weights. Even further, this encoding allows to choose the real numbers as domain for the weights. However, Yices can only deal with rationals. The index for smt indicates if integers (smt_i) or rationals (smt_r) are employed.

Since T$_\mathsf{T}$T and AProVE implement the slightly stronger KBO definition of [28] they can prove two TRSs (various/27 and TRCSR/Ex9_Luc06_GM) terminating which cannot be handled by our methods. (We did not investigate if one can specialize the encodings to also capture these systems but are convinced that this is in principle possible.) On the other hand T$_\mathsf{T}$T gives up on HM/t000 (and six more TRSs that derive from context sensitive rewriting) which specifies addition for natural numbers in decimal notation (using 104 rewrite rules). The problem is not the time limit but at some point the algorithm detects that it will require too many resources. To prevent a likely stack overflow from occurring, the computation is terminated and a "don't know" result is reported. AProVE does not cut off execution and consequently for this system (and 17 others) no result is obtained within 60 seconds. Also dkm fails on HM/t000 (timeout) whereas for none of our approaches this system seems to pose a problem at all; sat(4), pbc(4), smt_i, and smt_r succeed within 0.14, 0.12, 0.04, and 0.03 seconds. The algorithm dkm produces large sparse matrices during execution for some systems (e.g. for HM/t000 after 30 seconds 1GB of memory is used). We developed an OCaml module for sparse matrices which could drastically reduce memory usage (e.g. for HM/t000 after 30 seconds only 50MB). Nevertheless this effort just slightly improves the data for dkm. On the whole database the number of successful proofs did not increase and execution time decreases just slightly. Another issue that increased performance of dkm much more was *sorting* the set of equations in order to keep the internal data-structure (the matrices S_i^A) for MCD much smaller. This allowed us to prove various/21 in 0.026 sec whereas it was intractable for this method beforehand (out-of-memory after 8 minutes). The idea of sorting somehow contradicts the claim in [7] that MCD is preferable over the simplex method [5] due to its incremental nature. Our tests showed that restarting MCD (with sorted inequalities) whenever some new inequalities are added performs better than just incrementally adding one inequality after the other. Table 1 shows that by sorting (dkm') the method can prove three additional TRSs while the execution time drops by a factor of two. We also varied (within T$_\mathsf{T}$T) the back-end for solving linear inequations. While the standard implementation of T$_\mathsf{T}$T uses MCD, the special version T$_\mathsf{T}$T(simplex) implements the first phase of [5]. Again the results contradict the claim in [7] that MCD is better suited than the simplex method.

Table 2 KBO for 724 SRSs.

	strict precedence			quasi-precedence		
method(#bits)	total time	#successes	#timeouts	total time	#successes	#timeouts
sat/pbc(3)	12.8/11.6	24/24	0/0	14.3/12.8	24/24	0/0
sat/pbc(4)	13.7/11.7	30/30	0/0	15.3/12.5	30/30	0/0
sat/pbc(6)	16.5/13.0	33/33	0/0	18.4/14.2	33/33	0/0
sat/pbc(8)	18.5/18.3	33/33	0/0	22.0/17.6	33/33	0/0
smt_i	12.6	33	0	12.2	33	0
smt_r	12.5	33	0	12.0	33	0
AProVE	673.7	30	4			
$\mathsf{T_TT}$	45.4	30	0			
$\mathsf{T_TT}$(simplex)	22.3	33	0			
dkm	370.9	29	6			
dkm'	258.0	30	4			

Table 3 KBO with dependency pairs for 1381 TRSs/724 SRSs.

	TRSs			SRSs		
method(#bits)	total time	#successes	#timeouts	total time	#successes	#timeouts
sat(2)	2641.1	488	27	114.4	45	0
sat(3)	3478.3	491	31	336.2	48	1
sat(4)	6468.9	489	56	846.1	54	4
sat(5)	10921.7	488	120	1595.2	55	8
sat(6)	13757.9	488	174	2764.3	57	17
sat(10)	19821.7	478	265	8265.1	54	64
smt_i	1830.9	487	16	64.2	58	0
smt_r	1132.8	489	9	56.8	58	0
AProVE	15562.9	445	232	2902.7	37	36
$\mathsf{T_TT}$	23898.5	323	366	2623.5	27	36

As can be seen from the right part of Table 1, by admitting quasi-precedences one additional TRS (SK90/2.42, Example 1) can be proved terminating.

8.2 Results for SRSs

For SRSs we have similar results, as can be inferred from Table 2. The main difference is the larger number of bits needed for the propositional representation of the weights. The maximum number of SRSs is proved terminating with $k \geqslant 6$. Generally speaking $\mathsf{T_TT}$ performs better on SRSs than on TRSs concerning KBO because it can handle all systems within the time limit. However, again our experiments reveal that the implementation of $\mathsf{T_TT}$ is not complete, i.e., it proves termination of 30 SRSs only whereas our implementations succeed on 33 SRSs. The three SRSs that make up the difference (Trafo/dup11, Zantema/z069, Zantema/z070) derive from algebra (polyhedral groups). Also AProVE and dkm are unable to handle these systems (timeout) while the simplex version of $\mathsf{T_TT}$ performs well. Admitting quasi-precedences does not allow to prove more systems terminating by KBO.

8.3 Results with Dependency Pairs

For the results in this section the dependency pair definition of [6] together with dependency graph refinements presented in [18, 20] and with usable rules like in [18, 19] have been considered. In combination with the dependency pair setting KBO gains much power compared

to the direct approach. One reason is that by allowing argument filterings duplicating systems may become non-duplicating and consequently this (severe) restriction is eased.

We implemented the sat encoding from Section 7 and a similar encoding for smt. As can be seen in Table 3[9] the smt encoding is by far the fastest and for rational weights the solving time is drastically reduced compared to the integer case. Interestingly even if the weights are allowed to take rational values Yices returns integer solutions for almost all systems. (We refer to Section 9 for details.) Although for TRSs smt_r misses proving two systems compared to $\mathsf{sat}(3)$ it remains the optimal choice since no parameters have to be chosen to call the method. Furthermore the two missing systems can be handled by slightly increasing the timeout, i.e., smt_r is successful on TRCSR/PALINDROME_complete-noand_FR within 98.6 seconds and spends 204.4 seconds on TRCSR/PALINDROME_complete-noand_Z. Globally speaking within four minutes smt_r can investigate all systems whereas there remain timeouts for sat. For the SRS category SMT is the clear winner since it is by far the fastest and most powerful as demonstrated in the right part of Table 3.

9 Assessment and Conclusion

In this section we compare the three new approaches presented in this article. Let us start with the most important measurements: power and run time. Here smt is the clear winner. In [42] smt was not considered and pbc performed best. Since here a different database is employed the results look slightly worse for PB. On most examples pbc still outperforms sat but on a few systems (transformations from context-sensitive rewriting) pbc is not efficient at all. But pbc still scales better when using more bits (cf. Table 1). Furthermore, the pseudo-boolean approach is less implementation work since additions are performed by the SAT solver and also the transformation to CNF is not necessary. Of course smt combines the benefits of both approaches. The encoding is straight forward, short in implementation, and efficient.

But an advantage of the pseudo-boolean approach is the option of a *goal function* which should be minimized while preserving satisfiability of the constraints. Although the usage of such a goal function is not of computational interest it is useful for generating easily human readable proofs. We experimented with functions minimizing the weights for function symbols and reducing the comparisons in the precedence. The former has the advantage that one obtains a KBO proof with minimal weights which is nicely illustrated on the SRS Zantema/z113 consisting of the rules

$$\begin{array}{lll} 11 \to 43 & 33 \to 56 & 55 \to 62 \\ 12 \to 21 & 22 \to 111 & 34 \to 11 \\ 44 \to 3 & 56 \to 12 & 66 \to 21 \end{array}$$

TTT and AProVE produce the proof

$$\begin{array}{lll} w(1) = 32471712256 & w(2) = 48725750528 & w(3) = 43247130624 \\ w(4) = 21696293888 & w(5) = 44731872512 & w(6) = 40598731520 \\ & 3 \succ 1 \succ 2 & 1 \succ 4 \end{array}$$

[9] We stress that TTT has a weaker implementation of the dependency pair framework and consequently the results cannot directly be compared.

whereas pbc(6) produces

$$w(1) = 31 \qquad w(2) = 47 \qquad w(3) = 41$$
$$w(4) = 21 \qquad w(5) = 43 \qquad w(6) = 39$$
$$3 \succ 1 \succ 2 \qquad 3 \succ 5 \succ 6 \succ 2 \qquad 1 \succ 4$$

So for the first time it became clear that these large numbers are not needed to prove KBO orientability of the system Zantema/z113. Regarding the goal function dealing with the minimization of comparisons in the precedence we detected that using two (three, four, ten) bits to encode weights of function symbols 49 (56, 60, 60) TRSs can be proved terminating with empty precedence. To some extent it is clear that AProVE and T$_\text{T}$T produce such large weights since these implementations are based on the work in [7] which always ensures minimal precedences. Surprisingly our dkm' implementation produced a proof for various/21 with minimal (since empty) precedence and weight function $w(+) = 12$, $w(\mathsf{p1}) = 24$, $w(\mathsf{p2}) = 42$, $w(\mathsf{p5}) = 90$, and $w(\mathsf{p10}) = 141$, contradicting the discussion at the end of [7, Example 2] claiming that a precedence $\mathsf{p10} \succ \mathsf{p5} \succ \mathsf{p2} \succ \mathsf{p1}$ is needed.

Without dependency pairs there is no real gain in speed when allowing rationals for SMT. This might be due to the fact that only two proofs (HM/t000 and SK90/2.46) make use of rational valued weights in the TRS category and the same is true for SRSs (Trafo/dup11 and Trafo/dup16). But the difference becomes larger within the dependency pair setting. Suddenly 57 proofs for TRSs and 41 for SRSs contain rational valued weights. As already mentioned earlier [28] proves that restricting to integer weights does not change the power of the order.

All encodings presented in this article are polynomial in size. Performing binary addition using circuits as in Definition 5 stays polynomial because of introducing fresh variables. Clearly these fresh variables might double the search space for the SAT solver. The same holds for the transformation to CNF.

While running the experiments, sat and pbc produced different answers for the SRS Zantema/z13; pbc claimed KBO termination whereas sat answered "don't know". Chasing that discrepancy revealed a bug [9] in MiniSat+ (which has been corrected in the meantime).

Our experiments reveal that SMT suits an efficient and simple implementation best. However, for KBO linear arithmetic is sufficient which is not the case for other popular termination techniques like polynomial interpretations [30]. Currently SMT solvers do not support non-linear arithmetic at all or completely inappropriate. Thus it seems inevitable to use SAT as a back-end [13] for efficient implementations dealing with polynomials.

Comparing KBO with other direct approaches for proving termination such as LPO or polynomial interpretations, the question arises how powerful KBO is. Despite the severe restriction of non-duplication, there are KBO terminating TRSs that cannot be oriented by LPO or polynomial interpretations. Taking derivational complexity as measure for the power of an order, KBO surpasses the other approaches. The *derivation length*, denoted $\mathrm{dl}_{\mathcal{R}}(n)$, computes the length of a longest possible derivation starting at a term of size n. Hofbauer and Lautemann [22] showed that KBO can prove TRSs $\mathcal{R}$ terminating for which $\mathrm{dl}_{\mathcal{R}}$ cannot be bounded by a primitive recursive function whereas polynomials are bounded from above by double exponential functions. Lepper [31] proved that the Ackermann function gives an upper bound on $\mathrm{dl}_{\mathcal{R}}$ if termination of $\mathcal{R}$ can be proven by KBO. Moser [33] extended this result to infinite signatures. Concerning LPO, Weiermann [39] showed that multiple recursive functions suffice for bounding derivational complexity.

To stress the significance of our work we state some details about the use of KBO in the termination prover T$_\text{T}$T$_2$. It used the SMT encoding (with rationals) in the November

2008 termination competition for both categories in which it participated. Here KBO had to compete with a number of other termination techniques in T$_{\text{T}}$T$_2$. For the category *TRS Standard* (*SRS Standard*) KBO was used in about 22% (18%) of the successful termination proofs which shows the applicability of the method.

In this article we presented a method to compute upper bounds for weights. Furthermore we presented three logic-based encodings of KBO—pure SAT, PB, and SMT—which can be implemented more efficiently and with considerably less effort than the dedicated methods described in [7, 28]. Our experiments reveal enormous gains in efficiency. Especially the SMT encoding gives rise to a very fast and user-friendly implementation since the method is parameter-free (no restriction of bits for weights).

Acknowledgements We thank the AProVE-team for providing a special version of their tool, Stefan Jörer for developing a sparse matrix module for OCaml, Sarah Winkler for interfacing MiniSat and MiniSat+ into OCaml, and the anonymous referees for providing numerous suggestions which helped to improve the presentation.

References

1. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theor. Comput. Sci. **236**(1-2), 133–178 (2000)
2. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
3. Codish, M., Lagoon, V., Stuckey, P.: Solving partial order constraints for LPO termination. In: Proc. 17th International Conference on Rewriting Techniques and Applications, *LNCS*, vol. 4098, pp. 4–18 (2006)
4. Codish, M., Schneider-Kamp, P., Lagoon, V., Thiemann, R., Giesl, J.: SAT solving for argument filterings. In: Proc. 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, *LNAI*, vol. 4246, pp. 30–44 (2006)
5. Danzig, G.: Linear Programming and Extensions. Princeton University Press (1963)
6. Dershowitz, N.: Termination by abstraction. In: Proc. 20th International Conference on Logic Programming, *LNCS*, vol. 3132, pp. 1–18 (2004)
7. Dick, J., Kalmus, J., Martin, U.: Automating the Knuth-Bendix ordering. Acta Inform. **28**, 95–119 (1990)
8. Dutertre, B., de Moura, L.: A fast linear-arithmetic solver for DPLL(T). In: Proc. 18th International Conference on Computer Aided Verification, *LNCS*, vol. 4144, pp. 81–94 (2006)
9. Eén, N.: Personal conversation (2007). Google Group on MiniSat
10. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Proc. 6th International Conference on Theory and Applications of Satisfiability Testing, *LNCS*, vol. 2919, pp. 502–518 (2004)
11. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. J. on Satisf., Bool. Model. and Comput. **2**, 1–26 (2006)
12. Endrullis, J., Waldmann, J., Zantema, H.: Matrix interpretations for proving termination of term rewriting. J. Autom. Reason. **40**(2-3), 195–220 (2008)
13. Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: SAT solving for termination analysis with polynomial interpretations. In: Proc. 10th International Conference on Theory and Applications of Satisfiability Testing, *LNCS*, vol. 4501, pp. 340–354 (2007)
14. Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: Maximal termination. In: Proc. 19th International Conference on Rewriting Techniques and Applications, *LNCS*, vol. 5117, pp. 110–125 (2008)
15. Fuhs, C., Navarro-Marset, R., Otto, C., Giesl, J., Lucas, S., Schneider-Kamp, P.: Search techniques for rational polynomial orders. In: Proc. 9th International Conference on Artificial Intelligence and Symbolic Computation, *LNAI*, vol. 5144, pp. 109–124 (2008)
16. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic termination proofs in the dependency pair framework. In: Proc. 3rd International Joint Conference on Automated Reasoning, *LNAI*, vol. 4130, pp. 281–286 (2006)
17. Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: Combining techniques for automated termination proofs. In: Proc. 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, *LNAI*, vol. 3452, pp. 301–331 (2005)

18. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Proc. 5th International Workshop on Frontiers of Combining Systems, *LNAI*, vol. 3717, pp. 216–231 (2005)
19. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. J. Autom. Reason. **37**(3), 155–203 (2006)
20. Hirokawa, N., Middeldorp, A.: Automating the dependency pair method. Inf. and Comput. **199**(1-2), 172–199 (2005)
21. Hirokawa, N., Middeldorp, A.: Tyrolean termination tool: Techniques and features. Inf. and Comput. **205**(4), 474–511 (2007)
22. Hofbauer, D., Lautemann, C.: Termination proofs and the length of derivations (preliminary version). In: Proc. 3rd International Conference on Rewriting Techniques and Applications, *LNCS*, vol. 355, pp. 167–177 (1989)
23. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Comb. **4**, 373–395 (1984)
24. Khachiyan, L.: A polynomial algorithm in linear programming. Doklady Akademia Nauk SSSR **244**, 1093–1096 (1979)
25. Knuth, D., Bendix, P.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press (1970)
26. Koprowski, A., Middeldorp, A.: Predictive labeling with dependency pairs using SAT. In: Proc. 21st International Conference on Automated Deduction, *LNAI*, vol. 4603, pp. 410–425 (2004)
27. Koprowski, A., Waldmann, J.: Arctic termination ... below zero. In: Proc. 19th International Conference on Rewriting Techniques and Applications, *LNCS*, vol. 5117, pp. 202–216 (2008)
28. Korovin, K., Voronkov., A.: Orienting rewrite rules with the Knuth-Bendix order. Inf. and Comput. **183**, 165–186 (2003)
29. Kurihara, M., Kondo, H.: Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In: Proc. 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, *LNAI*, vol. 3029, pp. 827–837 (2004)
30. Lankford, D.: On proving term rewrite systems are noetherian. Tech. Rep. MTP-3, Louisiana Technical University, Ruston, LA, USA (1979)
31. Lepper, I.: Derivation lengths and order types of Knuth-Bendix orders. Theor. Comput. Sci. **269**(1-2), 433–450 (2001)
32. Marché, C.: Termination problems data base (TPDB), version 4.0 (2007). `http://www.lri.fr/~marche/tpdb`
33. Moser, G.: Derivational complexity of Knuth-Bendix orders revisited. In: Proc. 13th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, *LNAI*, vol. 4246, pp. 75–89 (2006)
34. Plaisted, D., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comp. **2**(3), 293–304 (1986)
35. Sato, H., Kurihara, M.: Implementation and performance evaluation of multi-completion procedures for term rewriting systems with recursive path orderings with status. IEICE Trans. on Inf. and Syst. **J89-D**(4), 624–631 (2006). In Japanese.
36. Schneider-Kamp, P., Thiemann, R., Annov, E., Codish, M., Giesl, J.: Proving termination using recursive path orders and SAT solving. In: Proc. 6th International Symposium on Frontiers of Combining Systems, *LNAI*, vol. 4720, pp. 267–282 (2007)
37. Steinbach, J.: Extensions and comparison of simplification orders. In: Proc. 3rd International Conference on Rewriting Techniques and Applications, *LNCS*, vol. 355, pp. 434–448 (1989)
38. Tseitin, G.: On the complexity of derivation in propositional calculus. In: Studies in Constructive Mathematics and Mathematical Logic, Part 2, pp. 115–125 (1968)
39. Weiermann, A.: Termination proofs for term rewriting systems by lexicographic path orderings imply multiply recursive derivation lengths. Theor. Comput. Sci. **139**(1-2), 355–362 (1995)
40. Zankl, H.: BDD and SAT techniques for precedence based orders. Master's thesis, University of Innsbruck (2006)
41. Zankl, H., Hirokawa, N., Middeldorp, A.: Constraints for argument filterings. In: Proc. 33rd International Conference on Current Trends in Theory and Practice of Computer Science, *LNCS*, vol. 4362, pp. 579–590 (2007)
42. Zankl, H., Middeldorp, A.: Satisfying KBO constraints. In: Proc. 18th International Conference on Rewriting Techniques and Applications, *LNCS*, vol. 4533, pp. 389–403 (2007)
43. Zankl, H., Middeldorp, A.: Increasing interpretations. In: Proc. 9th International Conference on Artificial Intelligence and Symbolic Computation, *LNAI*, vol. 5144, pp. 191–205 (2008)
44. Zantema, H., Waldmann, J.: Termination by quasi-periodic interpretations. In: Proc. 18th International Conference on Rewriting Techniques and Applications, *LNCS*, vol. 4533, pp. 404–418 (2007)