1 KBO as a Satisfaction Problem[⋆]

Harald Zankl and Aart Middeldorp

Institute of Computer Science, University of Innsbruck, Austria
{Harald.Zankl,Aart.Middeldorp}@uibk.ac.at

Abstract This note presents an approach to prove termination of term rewrite systems (TRSs) with the Knuth-Bendix order efficiently. The constraints for the weight function as well as for the precedence are encoded in propositional logic and the resulting formula is tested for satisfiability. Any satisfying assignment represents a weight function and a precedence such the induced Knuth-Bendix order orients the rules of the encoded TRS from left to right.

1.1 Introduction

This note is concerned with proving termination of term rewrite systems (TRSs) with the Knuth-Bendix order (KBO), a method invented by Knuth and Bendix 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 [4] and T$_\mathsf{T}$T [5], but neither of these tools incorporate KBO in their fully automatic mode. This is perhaps due to the fact that the algorithms known for deciding KBO orientability ([3,6]) are not easy to implement efficiently, despite the fact that the problem is known to be decidable in polynomial time [6]. The aim of this note is to make KBO a more attractive choice for termination tools by presenting a simple encoding of KBO orientability into propositional logic such that checking satisfiability of the resulting formula amounts to proving KBO termination.

Kurihara and Kondo [7] 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. Codish *et al.* [2] presented a more efficient formulation for the properties of a precedence. In [8] we showed how argument filterings in the dependency pair method can be encoded and combined it with the embedding order. The latter approach easily extends to other base orders which admit a propositional encoding. In this note we show that not only purely syntactical reduction orders like the lexicographic path order can be encoded into propositional logic, but also orders which additionally have a semantic component like KBO.

Section 1.2 presents a propositional encoding for KBO. In Section 1.3 we review the approach of [2] to model a strict precedence. After addressing some simple optimizations in Section 1.4 we compare our implementation with the one of T$_\mathsf{T}$T in Section 1.5 and show the enormous gain in efficiency.

1.2 KBO Encoding

We adopt the definition of KBO in [1] with the difference that we restrict the range of the weight function to $\mathbb{N}$. According to [6] this does not decrease the power of the order. In order to give a propositional encoding of KBO termination, we must take care of representing a

[⋆] This research is supported by FWF (Austrian Science Fund) project P18763.
[1] `www.lri.fr/~marche/termination-competition`

strict precedence and a weight function. For the former we introduce a set of new variables $X = \{X_{fg} \mid f, g \in \mathcal{F} \text{ with } f \neq g\}$ depending on the underlying signature $\mathcal{F}$ ([7]). The intended semantics of these variables is that an assignment which satisfies a variable X_{fg} corresponds to a precedence with $f \succ g$. For the weight function, symbols are considered in binary representation and the operations $>$, $=$, $\geq$, and $+$ must be redefined accordingly. The propositional encodings of $>$ and $=$ are similar to the ones in [2].

We fix the number k of bits that is available for representing natural numbers in binary. Let $a < 2^k$. We denote by $\mathbf{a} = \langle a_k, \dots, a_1 \rangle$ the binary representation of a where a_k is the most significant bit.

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

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

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

Definition 2. For a weight function (w, w_0), let $\mathrm{adm}(w, w_0)$ be the formula

$$\ulcorner \mathbf{w_0} > \mathbf{0} \urcorner \;\wedge \bigwedge_{c \in \mathcal{F}^{(0)}} \ulcorner \mathbf{c} \geq \mathbf{w_0} \urcorner \;\wedge \bigwedge_{f \in \mathcal{F}^{(1)}} \Big(\ulcorner \mathbf{f} = \mathbf{0} \urcorner \;\rightarrow \bigwedge_{g \in \mathcal{F},\, f \neq g} X_{fg}\Big).$$

For addition we use pairs. The first component represents the bit representation and the second component is a propositional formula which encodes the constraints for each digit.

Definition 3. We define $\ulcorner (\mathbf{f}, \varphi) + (\mathbf{g}, \psi) \urcorner$ as $(\mathbf{s}, \varphi \wedge \psi \wedge \gamma \wedge \sigma)$ with

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

and

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

where c_i $(0 \leqslant i \leqslant k)$ and s_i $(1 \leqslant i \leqslant k)$ are fresh variables that represent the carry and the sum of the addition. The condition $\neg c_k$ prevents a possible overflow.

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.

Definition 4. We define $\ulcorner (\mathbf{f}, \varphi) > (\mathbf{g}, \psi) \urcorner$ as $\ulcorner \mathbf{f} > \mathbf{g} \urcorner \wedge \varphi \wedge \psi$.

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

Definition 5. Let t be a term and (w, w_0) a weight function. The weight of a term is encoded as follows:

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

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

Definition 6. Let s and t be terms. We define the formula $\ulcorner s \succ_{\text{kbo}} t\urcorner$ as follows. If $s \in \mathcal{V}$ or $s = t$ or both $t \in \mathcal{V}$ and $t \notin \mathcal{V}\mathsf{ar}(s)$ or $|s|_x < |t|_x$ for some $x \in \mathcal{V}$ then $\ulcorner s \succ_{\text{kbo}} t\urcorner = \bot$. Otherwise

$$\ulcorner s \succ_{\text{kbo}} t\urcorner = \ulcorner W_s > W_t\urcorner \vee \left(\ulcorner W_s = W_t\urcorner \wedge \ulcorner s \succ'_{\text{kbo}} t\urcorner\right)$$

with

$$\ulcorner s \succ'_{\text{kbo}} t\urcorner = \begin{cases} \top & \text{if } s = f^n(t) \text{ with } t \in \mathcal{V} \text{ and } n > 0 \\ \ulcorner s_i \succ_{\text{kbo}} t_i\urcorner & \text{if } s = f(s_1, \ldots, s_n) \text{ and } t = f(t_1, \ldots, t_n) \\ X_{fg} & \text{if } s = f(s_1, \ldots, s_n),\ t = g(t_1, \ldots, t_m), \text{ and } f \neq g \end{cases}$$

where in the second clause i denotes the least $1 \leqslant j \leqslant n$ such that $s_j \neq t_j$.

1.3 Encoding the Precedence

To ensure the properties of a strict precedence we follow the approach of Codish *et al.* [2] who propose to interpret function symbols as natural numbers. The greater than relation $(>)$ then ensures that the function symbols are properly ordered. Let $|\mathcal{F}| = n$. Then we are looking for a mapping $m\colon \mathcal{F} \to \{1, \ldots, n\}$ such that for every propositional variable $X_{fg} \in X$ we have $m(f) > m(g)$. 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 $\langle f_l, \ldots, f_1\rangle$ with f_l the most significant bit.

Definition 7. For all $1 \leqslant j \leqslant l$:

$$||X_{fg}||_j = \begin{cases} (f_1 \wedge \neg g_1) & \text{if } j = 1, \\ (f_j \wedge \neg g_j) \vee \left((f_j \leftrightarrow g_j) \wedge ||X_{fg}||_{j-1}\right) & \text{if } j > 1. \end{cases}$$

After this step it is rather easy to define a propositional formula which is satisfiable whenever the TRS is KBO terminating and sufficiently many bits are allowed for the semantic part.

Definition 8. Let $\mathcal{R}$ be a TRS. The formula $\text{KBO}(\mathcal{R})$ is defined as

$$\text{adm}(w, w_0) \wedge \bigwedge_{l \to r \,\in\, \mathcal{R}} \ulcorner l \succ_{\text{kbo}} r\urcorner \;\wedge\; \bigwedge_{x \,\in\, X} (x \leftrightarrow ||x||_l).$$

Theorem 1. *A TRS $\mathcal{R}$ is KBO terminating whenever the propositional formula* $\text{KBO}(\mathcal{R})$ *is satisfiable.* □

method(#bits)	total time	# successes	# timeouts
kbo-sat(2)	12.72	72	0
kbo-sat(3)	15.21	77	0
kbo-sat(4)	18.77	78	0
kbo-sat(10)	126.03	78	1
T_TT	255.12	76	3

Table 1.1. KBO for 823 TRSs.

Since we do not know in advance the number k of bits needed to represent the weights, satisfiability of KBO($\mathcal{R}$) is not a necessary condition for KBO termination. As already mentioned in the introduction, the problem of finding correct weights can be solved in polynomial time. Nevertheless our exponential algorithm is more powerful in practice. In Section 1.5 we shall see that for all systems in the Termination Problem Data Base a rather small number of bits suffices.

1.4 Optimizations

When computing the constraints for the weights for terms s and t, removing function symbols and variables that occur both in s and in t is highly recommended. Concerning the admissibility condition, testing whether a function symbol f has weight zero can be expressed more concisely as $\neg(f_1 \vee \cdots \vee f_k)$. This similarly works for the constraint $\ulcorner \mathbf{w_0} > \mathbf{0} \urcorner$. Another optimization is the simplification of the constraint formula KBO($\mathcal{R}$) using equalities like $\neg\neg x \to x$, $x \wedge \top \to x$, and $x \wedge (x \vee y) \to x$.

1.5 Experimental Results

We implemented our encoding on top of T_TT. For testing satisfiability MiniSat was employed because it produced considerably faster times than an approach based on binary decision diagrams. Table 1.1 compares our implementation (kbo-sat) of KBO with the one in T_TT. AProVE, the only other known termination tool that contains an implementation of KBO, is not considered in the table because it produced seriously slower results than T_TT. We used the 823 TRSs in version 3.1 of the Termination Problem Data Base[2] that do not specify any strategy or theory. All tests were performed on a server equipped with an Intel® Xeon™ processor running at a CPU rate of 2.40GHz and 512MB of system memory.

As addressed in Section 1.2 one has to fix the number of bits which is used to represent natural numbers in binary representation. The actual choice is specified as the argument to kbo-sat. The column labeled total time mentions the execution time in seconds, success states how many of the 823 TRSs could be proved terminating whereas the last column indicates the number of timeouts. Interestingly, with $k = 4$ equally many TRSs could be proved terminating as with $k = 10$. The TRS higher-order_AProVE_HO_ReverseLastInit needs weight eight for the constant init and therefore can only be proved KBO terminating with $k \geqslant 4$.

Since T_TT employs the slightly stronger KBO definition of [6] it can prove one TRS (various_27) terminating which cannot be handled by kbo-sat. On the other hand, T_TT cannot prove the TRS various_21 terminating within 600 seconds whereas kbo_sat(4) only

[2] `http://www.lri.fr/~marche/tpdb/tpdb-3.1`

TRS	time	time to compute subformulas	# subformulas
HM_t000	1.69	0.87	7405
HM_t009	0.56	0.20	4200
currying_D33_01	0.37	0.08	3260

Table 1.2. The three most expensive TRSs for kbo-sat(4).

needs 0.17 seconds. Interestingly it also cannot handle HM_t000 which specifies addition for natural numbers in decimal notation (using 104 rewrite rules). The problem is not the timeout 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 behaves in a similar fashion on this TRS.) Our approach can show the KBO termination (already with $k = 3$) but it is by far the most expensive TRS. The bottleneck is the computation of all subformulas which are needed for the (linear) transformation to CNF, which is the required input format for MiniSat. Table 1.2 lists the three most time consuming TRSs for our approach together with the required time to compute the subformulas and their number.

We conclude with an example that can be proved KBO terminating with $k+1$ but not with k bits. This gives evidence that our approach is not complete in theory.

Example 1. Consider the parameterized TRS consisting of the two rules

$$\mathsf{f}(\mathsf{g}(x,y)) \to \mathsf{g}(\mathsf{f}(x),\mathsf{f}(y)) \qquad\qquad \mathsf{s}^n(x) \to \mathsf{f}(x)$$

with $n = 2^k$. 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 the largest element in the precedence. Therefore the second rule can only be handled when the weight of s is strictly larger than zero. It follows that the minimum weight of $\mathsf{s}^n(x)$ is $n+1 = 2^k+1$, which requires $k+1$ bits.

1.6 Conclusion

In this note we presented an efficient approach to determine KBO termination of TRSs. Although the algorithm is not complete in theory, in practice it is much faster and more powerful than any existing implementation for KBO termination.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
2. M. Codish, V. Lagoon, and P. Stuckey. Solving partial order constraints for LPO termination. In *Proc. 17th RTA*, volume 4098 of *LNCS*, 2006. To appear.
3. J. Dick, J. Kalmus, and U. Martin. Automating the Knuth-Bendix ordering. *Acta Infomatica*, 28:95–119, 1990.
4. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. 3th IJCAR*, LNAI, 2006. To appear.
5. N. Hirokawa and A. Middeldorp. Tyrolean termination tool. In *Proc. 16th International Conference on Rewriting Techniques and Applications*, volume 3467 of *LNCS*, pages 175–184, 2005.
6. K. Korovin and A. Voronkov. Orienting rewrite rules with the Knuth-Bendix order. *Information and Computation*, 183:165–186, 2003.
7. M. Kurihara and H. Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In *Proc. 17th IEA/AIE*, volume 3029 of *LNCS*, pages 827–837, 2004.
8. H. Zankl, N. Hirokawa, and A. Middeldorp. Constraints for argument filterings. This volume.