Constraints for Argument Filterings*

Harald Zankl, Nao Hirokawa, and Aart Middeldorp

Institute of Computer Science, University of Innsbruck, Austria

Abstract. The dependency pair method is a powerful method for automatically proving termination of rewrite systems. When used with traditional simplification orders like LPO and KBO, argument filterings play a key role. In this paper we propose an encoding of argument filterings in propositional logic. By incorporating propositional encodings of simplification orders, the search for suitable argument filterings is turned into a satisfiability problem. Preliminary experimental results show that our logic-based approach is significantly faster than existing implementations.

1 Introduction

The problem of proving termination of systems, processes and programs arises naturally in many areas of computer science. In this paper we are concerned with termination of term rewrite systems (TRSs), a formal model of computation that underlies functional programming and which is heavily used in symbolic computation and theorem proving.

Termination is undecidable in general but for term rewriting many powerful methods have been developed in the past decades. In this paper we are concerned with the dependency pair method [1], a relatively new and very powerful method which is used in almost every automatic termination prover. In this method termination problems are transformed into collections of ordering constraints, which are solved recursively by traditional techniques like the lexicographic path order (LPO), the Knuth-Bendix order (KBO), and polynomial interpretations. When using strictly monotone simplification orders like LPO and KBO, a significant increase in termination proving power is obtained by using argument filterings to simplify the ordering constraints. Finding a suitable argument filtering is a challenging search problem [9].

After recapitulating the main theorem underlying the dependency pair method in Section 2 we propose a simple encoding of argument filterings in propositional logic in Section 3. Propositional encodings of simplification orders [3, 11, 13, 15] can easily be incorporated, resulting in a propositional formula with the property that any satisfying assignment corresponds to an argument filtering and the parameters of the encoded order which solve the constraints and vice-versa. We describe such a combination with the embedding order in Section 4 and

* This research is supported by FWF (Austrian Science Fund) project P18763. Some of the results in this paper were first announced in [14].

with KBO in Section 5. In order to test the effectiveness of our approach, we implemented this combination on top of the recursive SCC algorithm of [9]. For satisfiability checking we used the state-of-the-art SAT solver MiniSat [5]. The results are compared with the divide and conquer algorithm implemented in T$_\text{T}$T [10] and described in Section 6. In Section 7 we recast a result concerning the interplay of argument filterings and usable rules [12] as a propositional formula, resulting in a free and fast implementation which adds significant power.

2 Dependency Pairs

We assume basic knowledge of term rewriting [2] and the dependency pair method [1, 6, 8, 12]. We just state the main theorem underlying the method and, because it plays an important role in the paper, recall the definition of argument filterings.

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 $\pi(f) = [i_1, \dots, i_m]$.

Theorem 1. *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 $(\geqslant, >)$ such that $\pi(\mathcal{U}(\mathcal{C}) \cup \mathcal{C}) \subseteq \geqslant$ and $\pi(\mathcal{C}) \cap > \neq \varnothing$.* □

In this paper we reformulate the above theorem as a satisfiability problem in propositional logic for specific reduction pairs. In Section 4 we address the embedding order and in Section 5 we address KBO, but first we explain how to represent argument filterings in propositional logic.

3 Representing Argument Filterings

Definition 2. *Let $\mathcal{F}$ be a signature. The set of propositional variables $\{X_f \mid f \in \mathcal{F}\} \cup \{X_f^i \mid f \in \mathcal{F}$ and $1 \leqslant i \leqslant \mathsf{arity}(f)\}$ is denoted by $\mathcal{X}_\mathcal{F}$. Let π be an argument filtering for $\mathcal{F}$. The induced assignment α_π is defined as follows:*

$$\alpha_\pi(X_f) = \begin{cases} \mathsf{true} & \text{if } \pi(f) = [i_1, \dots, i_m] \\ \mathsf{false} & \text{if } \pi(f) = i \end{cases} \quad \text{and} \quad \alpha_\pi(X_f^i) = \begin{cases} \mathsf{true} & \text{if } i \in \pi(f) \\ \mathsf{false} & \text{if } i \notin \pi(f) \end{cases}$$

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

Definition 3. *An assignment α for $\mathcal{X}_\mathcal{F}$ is said to be* argument filtering consistent *if for every n-ary function symbol $f \in \mathcal{F}$ such that $\alpha \not\vDash X_f$ there is a unique $i \in \{1, \dots, n\}$ such that $\alpha \vDash X_f^i$.*

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

Definition 4. *The propositional formula* $\mathsf{AF}(\mathcal{F})$ *is defined as*

$$\bigwedge_{f \in \mathcal{F}} \left(X_f \vee \bigvee_{i=1}^{\mathsf{arity}(f)} \left(X_f^i \wedge \bigwedge_{j \neq i} \neg X_f^j \right) \right).$$

Lemma 5. *An assignment* α *for* $\mathcal{X}_\mathcal{F}$ *is argument filtering consistent if and only if* $\alpha \vDash \mathsf{AF}(\mathcal{F})$. □

Definition 6. *Let* α *be an argument filtering consistent assignment for* $\mathcal{X}_\mathcal{F}$. *The argument filtering* π_α *is defined as follows:*

$$\pi_\alpha(f) = \begin{cases} [i \mid \alpha \vDash X_f^i] & \text{if } \alpha \vDash X_f, \\ i & \text{if } \alpha \nvDash X_f \text{ and } \alpha \vDash X_f^i \end{cases}$$

for all function symbols $f \in \mathcal{F}$.

Example 7. Consider a signature consisting of two binary function symbols f and g. The assignment α with $\alpha(X_\mathsf{f}) = \alpha(X_\mathsf{f}^2) = \alpha(X_\mathsf{g}^1) = \mathsf{true}$ and $\alpha(X_\mathsf{f}^1) = \alpha(X_\mathsf{g}) = \alpha(X_\mathsf{g}^2) = \mathsf{false}$ is argument filtering consistent. The induced argument filtering π_α consists of $\pi_\alpha(\mathsf{f}) = [2]$ and $\pi_\alpha(\mathsf{g}) = 1$.

4 Embedding

When reformulating Theorem 1 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 $\pi(\mathcal{U}(\mathcal{C}) \cup \mathcal{C}) \subseteq \geqslant$ and $\pi(\mathcal{C}) \cap > \neq \varnothing$ in Theorem 1. In Section 5 we consider KBO, which is a bit more challenging.

Definition 8. *The* embedding *order* $\trianglerighteq_\mathsf{emb}$ *is defined on terms as follows:* $s \trianglerighteq_\mathsf{emb} t$ *if* $s = t$ *or* $s = f(s_1, \ldots, s_n)$ *and either* $s_i \trianglerighteq_\mathsf{emb} t$ *for some* i *or* $t = f(t_1, \ldots, t_n)$ *and* $s_i \trianglerighteq_\mathsf{emb} t_i$ *for all* i. *The strict part is denoted by* $\triangleright_\mathsf{emb}$.

In the following we define propositional formulas $\ulcorner s \triangleright_\mathsf{emb}^\pi t \urcorner$ and $\ulcorner s \trianglerighteq_\mathsf{emb}^\pi t \urcorner$ which, in conjunction with $\mathsf{AF}(\mathcal{F})$, 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 $\ulcorner s =^\pi t \urcorner$ that represents all argument filterings which make s and t equal. (In the sequel we assume that $\wedge$ binds stronger than $\vee$.)

Definition 9. *Let s and t be terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. We define a propositional formula $\ulcorner s =^{\pi} t \urcorner$ over $\mathcal{X}_{\mathcal{F}}$ by induction on s and t. If $s \in \mathcal{V}$ then*

$$\ulcorner s =^{\pi} t \urcorner = \begin{cases} \top & \text{if } s = t, \\ \bot & \text{if } t \in \mathcal{V} \text{ and } s \neq t, \\ \neg X_g \wedge \bigvee\limits_{j=1}^{m} \big(X_g^j \wedge \ulcorner s =^{\pi} t_j \urcorner\big) & \text{if } t = g(t_1,\ldots,t_m). \end{cases}$$

Let $s = f(s_1,\ldots,s_n)$. If $t \in \mathcal{V}$ then

$$\ulcorner s =^{\pi} t \urcorner = \neg X_f \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i =^{\pi} t \urcorner\big).$$

If $t = g(t_1,\ldots,t_m)$ with $f \neq g$ then

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

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

$$\ulcorner s =^{\pi} t \urcorner = \neg X_f \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i =^{\pi} t_i \urcorner\big) \vee X_f \wedge \bigwedge_{i=1}^{n} \big(X_f^i \rightarrow \ulcorner s_i =^{\pi} t_i \urcorner\big).$$

Definition 10. *Let s and t be terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. We define propositional formulas $\ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner$ and $\ulcorner s \trianglerighteq^{\pi}_{\mathsf{emb}} t \urcorner = \ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner \vee \ulcorner s =^{\pi} t \urcorner$ over $\mathcal{X}_{\mathcal{F}}$ by induction on s and t. If $s \in \mathcal{V}$ then $\ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner = \bot$. Let $s = f(s_1,\ldots,s_n)$. If $t \in \mathcal{V}$ then*

$$\ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner = X_f \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i \trianglerighteq^{\pi}_{\mathsf{emb}} t \urcorner\big) \vee \neg X_f \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i \rhd^{\pi}_{\mathsf{emb}} t \urcorner\big).$$

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

$$X_f \wedge \Big(X_g \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i \trianglerighteq^{\pi}_{\mathsf{emb}} t \urcorner\big) \vee \neg X_g \wedge \bigvee_{j=1}^{m} \big(X_g^j \wedge \ulcorner s \rhd^{\pi}_{\mathsf{emb}} t_j \urcorner\big)\Big)$$

and

$$\neg X_f \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i \rhd^{\pi}_{\mathsf{emb}} t \urcorner\big).$$

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

$$\ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner = X_f \wedge \Bigg(\bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i \trianglerighteq^{\pi}_{\mathsf{emb}} t \urcorner\big) \vee \bigwedge_{i=1}^{n} \big(X_f^i \rightarrow \ulcorner s_i \trianglerighteq^{\pi}_{\mathsf{emb}} t_i \urcorner\big) \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i \rhd^{\pi}_{\mathsf{emb}} t_i \urcorner\big)\Bigg) \vee \neg X_f \wedge \bigvee_{i=1}^{n} \big(X_f^i \wedge \ulcorner s_i \rhd^{\pi}_{\mathsf{emb}} t_i \urcorner\big).$$

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

Lemma 11. *Let $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. If α is an assignment for $\mathcal{X}_{\mathcal{F}}$ such that $\alpha \vDash \ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner \wedge \mathsf{AF}(\mathcal{F})$ 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 $\alpha_\pi \vDash \ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner \wedge \mathsf{AF}(\mathcal{F})$.* □

We conclude this section by stating the propositional formulation of the termination criterion of Theorem 1 specialized to embedding.

Theorem 12. *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*

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

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

5 Knuth-Bendix Order

Our approach extends naturally to propositional encodings of other simplification orders [3, 11, 13, 15]. The encoding of LPO as a satisfiability problem has been pioneered by Kurihara and Kondo [11]. A more efficient encoding is described in [3]. An encoding of the multiset path order (MPO) is given in [13]. In [15] we described how to encode KBO as a satisfiability problem. In this section we integrate the encoding of KBO with argument filterings.

KBO is parameterized by two main components: a precedence and an admissible weight function. A precedence is a proper order $>$ on a signature. 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 $c \in \mathcal{F}$. The admissibility condition states that $f > g$ for all $g \in \mathcal{F} \setminus \{f\}$ whenever f is a unary function symbol with $w(f) = 0$. The weight of a term t is defined as follows: $w(t) = w_0$ if t is a variable and $w(f) + w(t_1) + \cdots + w(t_n)$ if $t = f(t_1, \ldots, t_m)$.

Definition 13. *Let $>$ be a 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) $t \in \mathcal{V}$ and $s = f^n(t)$ for a unary function symbol f and $n > 0$, or
  (2) $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$, and there exists an $i \in \{1, \ldots, n\}$ such that $s_j = t_j$ for all $1 \leqslant j < i$ and $s_i >_{\mathrm{kbo}} t_i$, or
  (3) $s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, and $f \succ g$.

Following [11], to encode the precedence the set of propositional variables $\mathcal{X}_\mathcal{F}$ is extended.

Definition 14. *Let $\mathcal{F}$ be a signature. The union of $\mathcal{X}_\mathcal{F}$ and $\{Y_{fg} \mid f, g \in \mathcal{F}$ and $f \neq g\}$ is denoted by $\mathcal{Y}_\mathcal{F}$.*

Our aim is to define a formula

$$\ulcorner s >^\pi_{\mathrm{kbo}} t \urcorner \wedge \mathsf{AF}(\mathcal{F}) \wedge \mathsf{PO}(\mathcal{F}) \wedge \mathsf{ADM}^\pi(\mathcal{F})$$

that is satisfiable if and only if there exist an argument filtering π, a precedence $>$, and an admissible weight function (w, w_0) such that $\pi(s) >_{\mathrm{kbo}} \pi(t)$. The conjunct $\mathsf{PO}(\mathcal{F})$ will ensure that the assignment for the variables in $\mathcal{Y}_\mathcal{F} \setminus \mathcal{X}_\mathcal{F}$ corresponds to a proper order on the signature. In [11] this is done by directly encoding transitivity and asymmetry. A more efficient encoding in which function symbols are mapped to natural numbers in binary representation is described in [3]. Our implementation follows the latter approach. The conjunct $\mathsf{ADM}^\pi(\mathcal{F})$ takes care of the admissibility condition (Definition 19).

Below we define the conjunct $\ulcorner s >^\pi_{\mathrm{kbo}} t \urcorner$. The basic idea is to adapt $\ulcorner s \rhd^\pi_{\mathsf{emb}} t \urcorner$ by incorporating the recursive definition of $>_{\mathrm{kbo}}$. The following definitions, taken from [3, 15], are needed to deal with the weight function in propositional logic.

We fix the number k of bits that is available for representing natural numbers in binary. Let $a < 2^k$. We denote by $\boldsymbol{a} = \langle a_k, \ldots, a_1 \rangle$ the binary representation of a where a_k is the most significant bit. The operations $>$, $=$, and $\geqslant$ are defined as follows:

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

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. We define $\ulcorner (\boldsymbol{f}, \varphi) + (\boldsymbol{g}, \psi) \urcorner$ as $(\boldsymbol{s}, \varphi \wedge \psi \wedge \gamma \wedge \sigma)$ 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 and $\oplus$ denotes exclusive or. The condition

$\neg c_k$ prevents a possible overflow. We define $\ulcorner(\boldsymbol{f}, \varphi) > (\boldsymbol{g}, \psi)\urcorner$ as $\ulcorner \boldsymbol{f} > \boldsymbol{g}\urcorner \wedge \varphi \wedge \psi$. 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.

With the above definitions in mind, we now focus on the propositional encoding of $>^{\pi}_{\mathrm{kbo}}$. First we take care of the non-duplication check of variables.

Definition 15. *The formula* $\mathsf{ND}^{\pi}(s,t)$ *is inductively defined as follows. If* $s \in \mathcal{V}$ *then* $\mathsf{ND}^{\pi}(s,t)$ *can safely be set to* $\bot$ *because* $\ulcorner s >^{\pi}_{\mathrm{kbo}} t\urcorner$ *will evaluate to* $\bot$ *anyway. If* $s \notin \mathcal{V}$ *and* $t \in \mathcal{V}$ *then* $\mathsf{ND}^{\pi}(s,t) = \ulcorner s \trianglerighteq^{\pi}_{\mathsf{emb}} t\urcorner$. *If* $s = f(s_1, \ldots, s_n)$ *and* $t = g(t_1, \ldots, t_m)$ *then*

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

with

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

The idea behind the recursive definition of $|s,c|_x$ 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 16. Consider the rule $l = \mathsf{f}(x, \mathsf{g}(y)) \to \mathsf{f}(x,y) = r$. Using two bits to represent numbers, the formula $\mathsf{ND}^{\pi}(l,r)$ evaluates to

$$\ulcorner(\langle 0, X^1_{\mathsf{f}}\rangle, \top) \geqslant (\langle 0, X^1_{\mathsf{f}}\rangle, \top)\urcorner \wedge \ulcorner(\langle 0, X^2_{\mathsf{f}} \wedge X^1_{\mathsf{g}}\rangle, \top) \geqslant (\langle 0, X^2_{\mathsf{f}}\rangle, \top)\urcorner$$

which says that for x there are more or less no constraints but for y we know that whenever the second argument of f is not deleted then also g must retain its argument.

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

Definition 17. *We define* $w^{\pi}(t)$ *as* $w'_{\pi}(t, \top)$ *with*

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

Here $d \cdot \boldsymbol{g}$ *stands for* $\langle d \wedge g_k, \ldots, d \wedge g_1\rangle$.

Definition 18. *Let s and t be terms. We define propositional formulas*

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

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

$$\ulcorner s >^{\pi}_{\mathrm{kbo}'} t \urcorner = X_f \wedge X_g \wedge Y_{fg} \vee \neg X_g \wedge \bigvee_{j=1}^{m} (X^j_g \wedge \ulcorner s >^{\pi}_{\mathrm{kbo}} t_j \urcorner) \vee$$
$$\neg X_f \wedge \bigvee_{i=1}^{n} (X^i_f \wedge \ulcorner s_i >^{\pi}_{\mathrm{kbo}} t \urcorner).$$

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

$$\ulcorner s >^{\pi}_{\mathrm{kbo}'} t \urcorner = X_f \wedge \langle s_1, \ldots, s_n \rangle >^{\mathsf{lex},\pi,f}_{\mathrm{kbo}} \langle t_1, \ldots, t_n \rangle \vee \neg X_f \wedge \bigvee_{i=1}^{n} (X^i_f \wedge \ulcorner s_i >^{\pi}_{\mathrm{kbo}} t_i \urcorner).$$

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

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

if $n > 0$.

Note that $\ulcorner s >^{\pi}_{\mathrm{kbo}'} t \urcorner$ corresponds to the definition of $>_{\mathrm{kbo}}$ in the case of equal weights (Definition 13). The peculiar looking equation $\ulcorner s >^{\pi}_{\mathrm{kbo}'} t \urcorner = \ulcorner s \rhd^{\pi}_{\mathsf{emb}} t \urcorner$ 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 19. *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.*

$$\ulcorner \boldsymbol{w_0} > \boldsymbol{0} \urcorner \wedge \bigwedge_{f \in \mathcal{F}} \big(\mathsf{constant}(f) \to \ulcorner \boldsymbol{f} \geqslant \boldsymbol{w_0} \urcorner\big) \wedge$$
$$\bigwedge_{f \in \mathcal{F}} \big(\ulcorner \boldsymbol{f} = \boldsymbol{0} \urcorner \wedge \mathsf{unary}(f) \to \bigwedge_{g \in \mathcal{F}, f \neq g} (X_g \to Y_{fg})\big)$$

with

$$\mathsf{constant}(f) = X_f \wedge \bigwedge_{i=1}^{\mathsf{arity}(f)} \neg X^i_f$$

and

$$\mathsf{unary}(f) = X_f \wedge \bigvee_{i=1}^{\mathsf{arity}(f)} (X^i_f \wedge \bigwedge_{i \neq j} \neg X^j_f).$$

We are now ready to state the propositional encoding of the termination criterion of Theorem 1 specialized to KBO.

Theorem 20. *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*

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

is satisfiable then there are an argument filtering π, a precedence $>$, and an admissible weight function (w, w_0) such that $\pi(\mathcal{U}(\mathcal{C}) \cup \mathcal{C}) \subseteq \geqslant_{\mathrm{kbo}}$ 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 20 holds if we don't put a bound on the number k of bits used for the representation of the weights. Of course, to get a finite formula we fix k in advance, which makes the approach incomplete. This is however not a serious problem in practice (cf. [15]).

6 Experimental Results

We implemented the encodings of the previous sections on top of the recursive SCC algorithm with the divide and conquer approach described in [9] for combining constraints in the termination prover T$_{\mathsf{T}}$T. The generated propositional formulas are tested for satisfiability with the state-of-the-art SAT solver MiniSat after applying Tseitin's translation to obtain a CNF. The propositional formulas in Sections 4 and 5 are written in a way to make them easily understandable for humans. Concerning efficiency however there are quite some useful optimizations which result in a large speedup. Consider e.g. the case of equal function symbols in Definition 9. The original formula

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

can be expressed more concisely as

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

since we know that $\mathsf{AF}(\mathcal{F})$ must hold anyway. Also the rules of commutativity, distributivity, etc. can considerably decrease the size of the generated formulas.

The results of our experiments are summarized in the tables below. We used a timeout of 60 seconds for each of the 865 TRSs in the 2006 edition of the Termination Problem Data Base. All tests were performed on a server equipped with an Intel® Xeon™ processor running at a CPU rate of 2.40GHz and 512MB

Table 1. Embedding and KBO.

	embedding			KBO			
	AProVE	TTT	sat	TTT	sat(2)	sat(3)	sat(4)
solved	194	194	194	279	271	317	322
timeout	12	6	0	135	2	3	8
time (in seconds)	735	417	150	8786	1172	1636	2181

of system memory. In Table 1 we compare our implementation of Theorems 12 and 20 with the divide and conquer algorithm of TTT described in [9]. For the embedding order we also tested AProVE [7].[1] The integers given as argument to sat denote the number of bits used to represent natural numbers in binary.

We implemented also an LPO version of Theorems 12 and 20. We refrain from describing it here as it has been (independently) done in [4]. We anticipate that by incorporating the advanced optimizations to minimize the size of the generated formulas sketched in the long version of the latter paper, the times in the sat columns can be reduced further.

An interesting possibility of the logic-based approach is that one can try different reduction pairs without having to worry about a strategy to control the order and time spent on each pair; just add the encoding of a different reduction pair with or without argument filterings as a new disjunct at the appropriate place in the overall formula. For instance, when using both KBO and LPO in TTT for a cycle in the dependency graph, one must specify the order in which they are tried. That this can make a difference can be seen from the data in Table 2. Here TTT(LK) (TTT(KL)) means that LPO is tried before (after) KBO, for each cycle that is generated during the recursive SCC algorithm. The numbers in *italics* in the sat columns are explained in the next section.

Table 2. KBO and LPO in parallel.

	TTT(LK)	TTT(KL)	sat(2)		sat(3)		sat(4)	
solved	310	295	305	*337*	338	*369*	343	*377*
timeout	121	136	6	*9*	9	*11*	14	*16*
time	7025	9025	1664	*1940*	2076	*2351*	2623	*2898*

[1] Since AProVE crashes when a stack overflow occurs, which happens frequently with KBO, we didn't manage to obtain data for the KBO columns.

7 Extensions

Allowing quasi-precedences in the encoding of KBO with argument filterings is an easy task (cf. [15]). Other precedence-based orders like the multiset path order are also easily handled (cf. [13]).

The propositional framework is perfectly suited to recast existing termination criteria in order to eliminate the often considerable effort to implement these criteria. Consider e.g. the following reformulation of a technique due to [12] for computing a restricted set of usable rules based on a given argument filtering.

Theorem 21. *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 $(\geqslant, >)$ such that $\pi(\mathcal{U}(\mathcal{C}, \pi) \cup \mathcal{C}) \subseteq \geqslant$ and $\pi(\mathcal{C}) \cap > \neq \varnothing$.* □

Rather than giving an explicit definition of the set $\mathcal{U}(\mathcal{C}, \pi)$ we encode the constraint $\pi(\mathcal{U}(\mathcal{C}, \pi) \cup \mathcal{C}) \subseteq \geqslant$ as the conjunction of[2]

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

and

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

Here U_f is a new propositional variable for every defined and every dependency pair symbol f.

Example 22. Consider the TRS consisting of the four rules

$$\begin{aligned} \mathsf{sum}(x, [\,]) &\to x & 0 + y &\to y \\ \mathsf{sum}(x, y :: z) &\to \mathsf{sum}(x + y, z) & \mathsf{s}(x) + y &\to \mathsf{s}(x + y) \end{aligned}$$

For the dependency pair $\mathsf{SUM}(x, y :: z) \to \mathsf{SUM}(x+y, z)$ none of the rewrite rules is usable under an argument filtering π with $\pi(\mathsf{SUM}) = [2]$ and the dependency pair simplifies to $\mathsf{SUM}(y :: z) \to \mathsf{SUM}(z)$ which can be oriented by $\rhd_{\mathsf{emb}}$ from left to right. Exactly this observation is mirrored in the last conjunction of the advanced usable rules formula that suggests that if a rule is used ($U_{root(l)}$ evaluates to *true*) then a defined symbol f occurring in the right hand side of the rule gives rise to further usable rules if this symbol f "remains" after applying the argument filtering. In the example we have the subformula $U_{\mathsf{SUM}} \to (X^1_{\mathsf{SUM}} \to U_+)$ which says that if the first argument of SUM is not deleted by the argument filtering then U_+ is set to *true* and $+$ gives rise to usable rules.

So by simply adding to the above constraint the encodings of the other (side) conditions we get essentially for free an implementation of a more powerful usable rule criterion than the one currently available in T$_\mathsf{T}$T (which amounts to the condition $\pi(\mathcal{U}(\mathcal{C}) \cup \mathcal{C}) \subseteq \geqslant$ in Theorem 1). Doing this for the KBO and LPO combination produces the numbers in *italics* in Table 2.

[2] Independently, in [4] a similar encoding is presented.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
3. M. Codish, V. Lagoon, and P. Stuckey. Solving partial order constraints for LPO termination. In *Proc. of the 17th International Conference on Rewriting Techniques and Applications*, volume 4098 of *LNCS*, pages 4–18, 2006.
4. M. Codish, P. Schneider-Kamp, V. Lagoon, R. Thiemann, and J. Giesl. Automating dependency pairs using SAT solvers. In *Proc. of the 8th International Workshop on Termination*, pages 60–63, 2006. Extended version to appear in Proc. of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LNCS, 2006.
5. N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. of the 6th International Conference on Theory and Applications of Satisfiability Testing*, volume 2919 of *LNCS*, pages 502–518, 2004.
6. J. Giesl, T. Arts, and E. Ohlebusch. Modular termination proofs for rewriting using dependency pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.
7. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. of the 3rd International Joint Conference on Automated Reasoning*, LNAI, pages 281–286, 2006.
8. N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In *Proc. of the 15th International Conference on Rewriting Techniques and Applications*, volume 3091 of *LNCS*, pages 249–268, 2004.
9. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1,2):172–199, 2005.
10. N. Hirokawa and A. Middeldorp. Tyrolean termination tool. In *Proc. of the 16th International Conference on Rewriting Techniques and Applications*, volume 3467 of *LNCS*, pages 175–184, 2005.
11. M. Kurihara and H. Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In *Proc. of the 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 3029 of *LNCS*, pages 827–837, 2004.
12. R. Thiemann, J. Giesl, and P. Schneider-Kamp. Improved modular termination proofs using dependency pairs. In *Proc. of the 2nd International Joint Conference on Automated Reasoning*, volume 3097 of *LNAI*, pages 75–90, 2004.
13. H. Zankl. BDD and SAT techniques for precedence based orders. Master's thesis, University of Innsbruck, 2006. Available at `http://cl-informatik.uibk.ac.at/HZ.pdf`.
14. H. Zankl, N. Hirokawa, and A. Middeldorp. Constraints for argument filterings. In *Proc. of the 8th International Workshop on Termination*, pages 50–54, 2006.
15. H. Zankl and A. Middeldorp. KBO as a satisfaction problem. In *Proc. of the 8th International Workshop on Termination*, pages 55–59, 2006. Full version available at `http://arxiv.org/abs/cs.SC/0608032`.