Matrix Interpretations for Proving Termination of Term Rewriting

Endrullis, J.; Waldmann, J.; Zantema, H.

published in
Journal of Automated Reasoning
2008

DOI (link to publisher)
10.1007/s10817-007-9087-9

document version
Publisher's PDF, also known as Version of record

Matrix Interpretations for Proving Termination of Term Rewriting

Jörg Endrullis[1], Johannes Waldmann[2], and Hans Zantema[3]

[1] Department of Computer Science, Vrije Universiteit Amsterdam
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
joerg@few.vu.nl

[2] Hochschule für Technik, Wirtschaft und Kultur (FH) Leipzig
Fb IMN, PF 30 11 66, D-04251 Leipzig, Germany
waldmann@imn.htwk-leipzig.de

[3] Department of Computer Science, Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
h.zantema@tue.nl

Abstract. We present a new method for automatically proving termination of term rewriting. It is based on the well-known idea of interpretation of terms where every rewrite step causes a decrease, but instead of the usual natural numbers we use vectors of natural numbers, ordered by a particular non-total well-founded ordering. Function symbols are interpreted by linear mappings represented by matrices. This method allows to prove termination and relative termination. A modification of the latter in which strict steps are only allowed at the top, turns out to be helpful in combination with the dependency pair transformation.

By bounding the dimension and the matrix coefficients, the search problem becomes finite. Our implementation transforms it to a Boolean satisfiability problem (SAT), to be solved by a state-of-the-art SAT solver. Our implementation performs well on the Termination Problem Data Base: better than 5 out of 6 tools that participated in the 2005 termination competition in the category of term rewriting.

1 Introduction

The annual Termination Competition [2] has given a new drive to the quest for automated methods to obtain termination proofs for term rewriting.

The termination provers do apply established methods (path orderings, dependency pairs, interpretations, labellings) as well as new methods (RFC match bounds). Two insights are that general methods can be restricted to special cases, gaining efficiency without loosing too much power, and combining methods may lead to strong improvements. We present here one such phenomenon: termination proofs from interpretations into a well-founded monotone algebra. This is a well-known general theme, but our point is

– the special choice of the algebra, and
– the special implementation of how to find suitable interpretations.

The carrier of the algebra consists of vectors of natural numbers on which we define a well-founded ordering that is not total. Each function symbol is interpreted by a suitable linear mapping. This method allows to prove termination and relative termination. It has been proposed for string rewriting by Hofbauer and Waldmann [11]. In the present paper, we discuss its extension to term rewriting and a modification that allows to prove relative top-termination, i.e., a variant of relative termination where the strict steps are only allowed on top level. The latter is very helpful when using the dependency pair transformation. In order to cover the two-sorted nature of the dependency pair transformation, our monotone algebra setting is presented many-sorted.

We have implemented the method by bounding the dimension and the matrix coefficients, resulting in a search problem with a finite but typically huge search space. This is solved by transforming this finite search problem to a SAT problem, and using the state-of-the-art SAT solver SatELiteGTI, [3]. This performs surprisingly well on the Termination Problem Data Base, see section 7.

The main part of the paper is organized as follows. We present a many-sorted monotone algebra framework for relative termination and relative top-termination in Section 3, generalizing earlier results on monotone algebras. Then we choose the matrix instance of this framework in Section 4. Later, we combine this with the Dependency Pair method in Section 5. Our implementation is described in Section 6 and its performance is discussed in Section 7.

Our methods are illustrated by examples. They are kept simple for the sake of presentation. Nevertheless none of them can be proved terminating by any of the programs that participated in the Termination Competition 2005 [2].

2 Preliminaries

Let S be a non-empty set of sorts, and let Σ be an S-sorted signature, being a set of operation symbols each having a fixed arity in $S^* \times S$. An *S-sorted set* A is defined to consist of a set A_s for every $s \in S$. For an S-sorted set $\mathcal{X}$ of variable symbols let $\mathcal{T}(\Sigma, \mathcal{X})$ be the S-sorted set of terms over Σ and $\mathcal{X}$, that is, the smallest S-sorted set satisfying

– $x_s \in \mathcal{T}(\Sigma, \mathcal{X})_s$ for all $x_s \in \mathcal{X}_s$, and
– if the arity of $f \in \Sigma$ is $((s_1, \dots, s_n), s)$ and $t_i \in \mathcal{T}(\Sigma, \mathcal{X})_{s_i}$ for $i = 1, \dots, n$, then $f(t_1, \dots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})_s$.

A *term rewriting system* (TRS) R over $\Sigma, \mathcal{X}$ is a S-sorted set in which for every $s \in S$ the set R_s consists of pairs $(\ell, r) \in \mathcal{T}(\Sigma, \mathcal{X})_s \times \mathcal{T}(\Sigma, \mathcal{X})_s$, for which $\ell \notin \mathcal{X}_s$ and all variables in r occur in ℓ. Pairs (ℓ, r) are called *rewrite rules* of sort s and are usually written as $\ell \to r$.

An S-sorted relation $\to$ over an S-sorted set A is defined to be an S-sorted set for which $\to_s \subseteq A_s \times A_s$ for every $s \in S$.

A substitution $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$ is defined by a map $\sigma_s : \mathcal{X}_s \to \mathcal{T}(\Sigma, \mathcal{X})_s$ for every $s \in S$. These extend to terms in the obvious way.

For a TRS R the (S-sorted) *top rewrite relation* $\overset{top}{\rightarrow}_R$ on $\mathcal{T}(\Sigma, \mathcal{X})$ is defined by $t \overset{top}{\rightarrow}_{R,s} u$ if and only if there is a rewrite rule $\ell \rightarrow r \in R_s$ and a substitution $\sigma : \mathcal{X} \rightarrow \mathcal{T}(\Sigma, \mathcal{X})$ such that $t = \ell\sigma$ and $u = r\sigma$. The (S-sorted) *rewrite relation* $\rightarrow_R$ is defined to be the smallest S-sorted relation satisfying

- if $t \overset{top}{\rightarrow}_R u$ then $t \rightarrow_R u$, and
- if $t_i \rightarrow_{R,s_i} u_i$ and $t_j = u_j$ for $j \neq i$, then $f(t_1, \ldots, t_n) \rightarrow_{R,s} f(u_1, \ldots, u_n)$ for every $f \in \Sigma$ of arity $((s_1, \ldots, s_n), s)$ and every $i = 1, \ldots, n$.

For S-sorted binary relations we write $\cdot$ for sort-wise relation composition, and * for sort-wise transitive reflexive closure.

An S-sorted relation $\rightarrow$ is called *well-founded* or *terminating* if for no $s \in S$ an infinite sequence $t_1, t_2, t_3, \ldots$ exists such that $t_i \rightarrow_s t_{i+1}$ for all $i = 1, 2, 3, \ldots$.

A TRS R is called *terminating* if $\rightarrow_R$ is well-founded. Termination is also called *strong normalization*; therefore the property of R being terminating is written as $\mathsf{SN}(R)$.

A binary relation $\rightarrow_1$ is called *terminating relative to* a binary relation $\rightarrow_2$, written as $\mathsf{SN}(\rightarrow_1 / \rightarrow_2)$, if for no $s \in S$ an infinite sequence $t_1, t_2, t_3, \ldots$ exists such that

- $t_i \rightarrow_{1,s} t_{i+1}$ for infinitely many values of i, and
- $t_i \rightarrow_{2,s} t_{i+1}$ for all other values of i.

We use the notation $\rightarrow_1 / \rightarrow_2$ to denote $\rightarrow_2^* \cdot \rightarrow_1 \cdot \rightarrow_2^*$; it is easy to see that $\mathsf{SN}(\rightarrow_1 / \rightarrow_2)$ coincides with well-foundedness of $\rightarrow_1 / \rightarrow_2$. We write $\mathsf{SN}(R/S)$ as a shorthand for $\mathsf{SN}(\rightarrow_R / \rightarrow_S)$, and we write $\mathsf{SN}(R_{top}/S)$ as a shorthand for $\mathsf{SN}(\overset{top}{\rightarrow}_R / \rightarrow_S)$.

3 Monotone Algebras

A Σ-algebra $(A, [\cdot])$ is defined to consist of a S-sorted set A, and for every $f \in \Sigma$ a function $[f] : A_{s_1} \times \cdots \times A_{s_n} \rightarrow A_s$, where $((s_1, \ldots, s_n), s)$ is the arity of f. This function $[f]$ is called the *interpretation* of f.

Let $\alpha_s : \mathcal{X}_s \rightarrow A_s$ for every $s \in S$; this collection of maps α_s is written as $\alpha : \mathcal{X} \rightarrow A$. We define the term evaluation $[\cdot, \alpha] : \mathcal{T}(\Sigma, \mathcal{X}) \rightarrow A$ inductively by

$$[x, \alpha] = \alpha_s(x),$$
$$[f(t_1, \ldots, t_n), \alpha] = [f]([t_1, \alpha], \ldots, [t_n, \alpha])$$

for $f \in \Sigma$ and $x \in \mathcal{X}_s$.

Definition 1. *An operation $[f] : A_{s_1} \times \cdots \times A_{s_n} \rightarrow A_s$ is* monotone *with respect to an S-sorted binary relation $\rightarrow$ on A if for all $a_i, b_i \in A_{s_i}$ for $i = 1, \ldots, n$ with $a_i \rightarrow_{s_i} b_i$ for some i and $a_j = b_j$ for all $j \neq i$ we have*

$$[f](a_1, \ldots, a_n) \rightarrow_s [f](b_1, \ldots, b_n).$$

A weakly monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ *is a Σ-algebra $(A, [\cdot])$ equipped with two S-sorted relations $>$, $\gtrsim$ on A such that*

– *$>$ is well-founded;*
– *$> \cdot \gtrsim \subseteq >$;*
– *for every $f \in \Sigma$ the operation $[f]$ is monotone with respect to $\gtrsim$.*

An extended monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ *is a weakly monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ in which moreover for every $f \in \Sigma$ the operation $[f]$ is monotone with respect to $>$.*

The combination $>, \gtrsim$ is closely related to the notion of *reduction pair* in the dependency pair framework, e.g. in [8]. A crucial difference is that the relations in a reduction pair are relations on terms that are closed under substitutions, while in our setting they are relations on the arbitrary (many-sorted) set A.

In the sequel we often omit sort information, e.g. writing $[t, \alpha] > [u, \alpha]$ rather than $[t, \alpha] >_s [u, \alpha]$. A TRS given without sort information is assumed to be one-sorted, i.e., S consists of one element.

The one-sorted version of extended monotone algebra where $\gtrsim$ is left implicit by defining it as the union of $>$ and equality is called *well-founded monotone algebra* in [14,15]. A main theorem states that a TRS is terminating if and only if there is a well-founded monotone algebra $(A, [\cdot], >)$ such that $[\ell, \alpha] > [r, \alpha]$ for every rule $\ell \to r$ and every $\alpha : \mathcal{X} \to A$. First we show that for relative termination we have a similar characterization based on extended monotone algebras, but not on this earlier version of well-founded monotone algebras.

Theorem 1. *Let R, S be TRSs over a signature Σ. Then*

1. *$\mathsf{SN}(R/S)$ if and only if there exists an extended monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ such that $[\ell, \alpha] > [r, \alpha]$ for every rule $\ell \to r$ in R and $[\ell, \alpha] \gtrsim [r, \alpha]$ for every rule $\ell \to r$ in S, for every $\alpha : \mathcal{X} \to A$.*
2. *$\mathsf{SN}(R_{top}/S)$ if and only if there exists a weakly monotone Σ-algebra $(A, [\cdot], >, \gtrsim)$ such that $[\ell, \alpha] > [r, \alpha]$ for every rule $\ell \to r$ in R and $[\ell, \alpha] \gtrsim [r, \alpha]$ for every rule $\ell \to r$ in S, for every $\alpha : \mathcal{X} \to A$.*

Proof. For the 'if'-part of part *1* assume such an extended monotone algebra $(A, [\cdot], >, \gtrsim)$ exists; we have to prove $\mathsf{SN}(R/S)$. So assume an infinite reduction

$$t_1 \to_{R \cup S} t_2 \to_{R \cup S} t_3 \to_{R \cup S} \cdots$$

containing infinitely many R-steps. Choose $\alpha : \mathcal{X} \to A$ arbitrary. Due to monotonicity with respect to $>$ we obtain $[t_i, \alpha] > [t_{i+1}, \alpha]$ if $t_i \to_R t_{i+1}$, and due to monotonicity with respect to $\gtrsim$ we obtain $[t_i, \alpha] \gtrsim [t_{i+1}, \alpha]$ if $t_i \to_S t_{i+1}$. Since $> \cdot \gtrsim \subseteq >$ we obtain $> \cdot \gtrsim^* \subseteq >$, hence for $t_i \to_R t_{i+1} \to_S^* t_j$ we obtain $[t_i, \alpha] > [t_j, \alpha]$. Since there are infinitely many R-steps this gives rise to an infinite decreasing sequence with respect to $>$, contradicting well-foundedness.

The proof of the 'if'-part of part *2* is similar; now all $\to_R$-steps in the assumed infinite reduction are $\overset{top}{\to}_R$-steps, by which monotonicity with respect to $>$ is not required.

For the 'only if'-part assume $\mathsf{SN}(R/S)$, respectively $\mathsf{SN}(R_{top}/S)$, holds. Choose $A = \mathcal{T}(\Sigma, \mathcal{X})$, and $[f](t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ for all $f \in \Sigma$. Define $> = (\to_R / \to_S)^+$ and $\gtrsim = (\to_{R \cup S})^*$, respectively $> = (\overset{top}{\to}_R / \to_S)^+$ and

$\gtrsim\, = \,\to_S^*$. Then $(A, [\cdot], >, \gtrsim)$ satisfies all requirements; where well-foundedness of $>$ is concluded from the assumption $\mathsf{SN}(R/S)$, respectively $\mathsf{SN}(R_{top}/S)$. □

For the relations $>, \gtrsim$ we typically have in mind some more properties, like transitivity of both $>$ and $\gtrsim$, reflexivity of $\gtrsim$, and $\gtrsim \cdot > \cdot \gtrsim \,\subseteq\, > \,\subseteq\, \gtrsim$. However, from the proof of Theorem 1 we see that these properties are not essential.

For this characterization of relative termination the general notion of extended monotone algebra is essential: it does not hold for the restricted case where $\gtrsim$ coincides with the union of $>$ and equality. For instance, if R consists of the rule $f(f(x)) \to f(g(f(x)))$ and S consists of the rule $f(x) \to g(f(x))$ then $\mathsf{SN}(R/S)$ holds, but no extended monotone algebra exists in which $\gtrsim$ coincides with the union of $>$ and equality and the properties of Theorem 1 hold.

Now we arrive at the theorem for extended monotone algebras as we will use it for proving (relative) termination by matrix interpretations.

Theorem 2. *Let R, S be TRSs over a signature Σ.*

1. *Let $(A, [\cdot], >, \gtrsim)$ be an extended monotone Σ-algebra such that $[\ell, \alpha] \gtrsim [r, \alpha]$ for every rule $\ell \to r$ in $R \cup S$ and every $\alpha : \mathcal{X} \to A$. Let R' consist of all rules $\ell \to r$ from $R \cup S$ satisfying $[\ell, \alpha] > [r, \alpha]$ for every $\alpha : \mathcal{X} \to A$.*
   Then $\mathsf{SN}((R \setminus R')/(S \setminus R'))$ implies $\mathsf{SN}(R/S)$.
2. *Let $(A, [\cdot], >, \gtrsim)$ be a weakly monotone Σ-algebra such that $[\ell, \alpha] \gtrsim [r, \alpha]$ for every rule $\ell \to r$ in $R \cup S$ and every $\alpha : \mathcal{X} \to A$. Let R' consist of all rules $\ell \to r$ from R satisfying $[\ell, \alpha] > [r, \alpha]$ for every $\alpha : \mathcal{X} \to A$.*
   Then $\mathsf{SN}((R \setminus R')_{top}/S)$ implies $\mathsf{SN}(R_{top}/S)$.

Proof. For part *1* assume $\mathsf{SN}((R \setminus R')/(S \setminus R'))$. Take any infinite reduction with respect to $R \cup S$. From Theorem 1 part *1* we conclude $\mathsf{SN}(R'/(R \cup S))$, so this infinite reduction contains only finitely many R'-steps. So after removing a finite initial part, this reduction only consists of $(R \cup S) \setminus R'$-steps. Since $\mathsf{SN}((R \setminus R')/(S \setminus R'))$ this remaining part contains only finitely many $R \setminus R'$-steps. So the original infinite reduction contains only finitely many R-steps. Hence we proved $\mathsf{SN}(R/S)$.

For part *2* assume $\mathsf{SN}((R \setminus R')_{top}/S)$. Take any infinite reduction with respect to $\overset{top}{\to}_R \cup \to_S$. From Theorem 1 part *2* we conclude $\mathsf{SN}(R'_{top}/(R \cup S))$, so this infinite reduction contains only finitely many $\overset{top}{\to}_{R'}$-steps. So after removing a finite initial part, this reduction only consists of $\overset{top}{\to}_{R \setminus R'}$-steps and $\to_S$-steps. Since $\mathsf{SN}((R \setminus R')_{top}/S)$ this remaining part contains only finitely many $\overset{top}{\to}_{R \setminus R'}$-steps. So the original infinite reduction contains only finitely many $\overset{top}{\to}_R$-steps, proving $\mathsf{SN}(R_{top}/S)$. □

The basic way to apply Theorem 2 is as follows. If $\mathsf{SN}(R/S)$ (or $\mathsf{SN}(R_{top}/S)$) has to be proved then try to find an extended (or weakly) monotone Σ-algebra satisfying the conditions for which R' is not empty. Then the proof obligation is weakened to $\mathsf{SN}((R \setminus R')/(S \setminus R'))$ (or $\mathsf{SN}((R \setminus R')_{top}/S)$). For this we again apply Theorem 2 in the same way. This is repeated until $R \setminus R' = \emptyset$, for which the remaining proof obligation $\mathsf{SN}((R \setminus R')/(S \setminus R'))$ (or $\mathsf{SN}((R \setminus R')_{top}/S))$ trivially holds. Proving termination rather than relative termination is a special case of this approach: then S is empty in $\mathsf{SN}(R/S)$.

Application of Theorem 2 is well-known for the case where A consists of the natural numbers, or natural numbers ≥ 2, and all functions $[f]$ are polynomials, and $>$ and $\gtrsim$ have their usual meaning. For part *1* strict monotonicity is required, while for part *2* weak monotonicity is sufficient. In this polynomial case $\gtrsim$ coincides with the union of $>$ and equality. In the matrix interpretations in the vector algebras considered in this paper, this is not the case for dimensions > 1.

4 Matrix Interpretations

In this paper we focus on interpretations based on matrices. For the basic version this means that we fix a dimension d and construct a one-sorted extended monotone algebra $(A, [\cdot], >, \gtrsim)$ in which $A = \mathbf{N}^d$. Without any complication this extends to the many-sorted setting in which every sort has its own dimension. To keep the presentation simple here we restrict to the one-sorted case.

The relations $>$ and $\gtrsim$ on A are defined as follows:

$$(v_1, \ldots, v_d) > (u_1, \ldots, u_d) \iff v_1 > u_1 \wedge v_i \geq u_i \text{ for } i = 2, 3, \ldots, d,$$

$$(v_1, \ldots, v_d) \gtrsim (u_1, \ldots, u_d) \iff v_i \geq u_i \text{ for } i = 1, 2, \ldots, d.$$

All requirements for $>$ and $\gtrsim$ from Definition 1 trivially hold. Note that $\gtrsim$ does not coincide with the union of $>$ and equality.

For the interpretation $[c]$ of a symbol $c \in \Sigma$ of arity 0 we choose any element of A. For the interpretation $[f]$ of a symbol $f \in \Sigma$ of arity $n \geq 1$ we choose n matrices $F_1, F_2, \ldots, F_n$ over $\mathbf{N}$, each of size $d \times d$, such that the upper left elements $(F_i)_{1,1}$ are positive for all $i = 1, 2, \ldots, n$, and a vector $\boldsymbol{f} \in \mathbf{N}^d$. Now we define

$$[f](\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}) \ = \ F_1\boldsymbol{v_1} + \cdots + F_n\boldsymbol{v_n} + \boldsymbol{f}$$

for all $\boldsymbol{v_1}, \ldots, \boldsymbol{v_n} \in A$. One easily checks that f is monotonic with respect to $\gtrsim$. Due to positiveness of the upper left matrix elements we also conclude that f is monotonic with respect to $>$. So by choosing all $[f]$ of this shape all requirements of an extended monotone algebra are fulfilled.

In order to apply Theorem 2, part *1*, we should be able to check whether $[\ell, \alpha] \gtrsim [r, \alpha]$ or $[\ell, \alpha] > [r, \alpha]$ for all $\alpha : \mathcal{X} \to A$, for given rewrite rules $\ell \to r$. Let $x_1, \ldots, x_k$ be the variables occurring in ℓ, r. Then due to the linear shape of the functions $[f]$ we can compute matrices $L_1, \ldots, L_k, R_1, \ldots, R_k$ and vectors $\boldsymbol{l}, \boldsymbol{r}$ such that

$$[\ell, \alpha] \ = \ L_1\boldsymbol{x_1} + \cdots + L_k\boldsymbol{x_k} + \boldsymbol{l}$$

and

$$[r, \alpha] \ = \ R_1\boldsymbol{x_1} + \cdots + R_k\boldsymbol{x_k} + \boldsymbol{r}$$

where $\alpha(x_i) = \boldsymbol{x_i}$ for $i = 1, \ldots, k$.

For matrices $B, C \in \mathbf{N}^{d \times d}$ write

$$B \gtrsim C \iff \forall i, j : (B)_{i,j} \geq (C)_{i,j}.$$

The following lemma states how the conditions of Theorem 2 can be checked.

Lemma 1. *Let $L_1, \ldots, L_k, R_1, \ldots, R_k$ and $\boldsymbol{l}, \boldsymbol{r}$ correspond to a rewrite rule $\ell \to r$ as described above. Then*

– $[\ell, \alpha] \gtrsim [r, \alpha]$ *for every $\alpha : \mathcal{X} \to A$ if and only if*

$$L_i \gtrsim R_i \text{ for } i = 1, \ldots, k, \text{ and } \boldsymbol{l} \gtrsim \boldsymbol{r},$$

– $[\ell, \alpha] > [r, \alpha]$ *for every $\alpha : \mathcal{X} \to A$ if and only if*

$$L_i \gtrsim R_i \text{ for } i = 1, \ldots, k, \text{ and } \boldsymbol{l} \gtrsim \boldsymbol{r}, \text{ and } l_1 > r_1.$$

So for applying Theorem 2, part *1*, we fix a dimension d and choose matrices F_i and vectors $\boldsymbol{f}$ for all $f \in \Sigma$. Next for every rule $\ell \to r \in R \cup S$ we check whether $L_i \gtrsim R_i$ for $i = 1, \ldots, k$ and $\boldsymbol{l} \gtrsim \boldsymbol{r}$. If so, then we may remove all rules moreover satisfying $l_1 > r_1$. After having done so we may continue by choosing new matrices, or by any other technique for proving (relative) termination.

Note that for our matrix interpretations after choosing the interpretation checking whether a left hand side is greater (or greater or equal) than a right hand side is decidable due to Lemma 1, in contrast to non-linear polynomial interpretations.

Example 1. Consider the TRS consisting of the following rule.

$$\mathrm{h}(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)) \to \mathrm{h}(\mathrm{g}(u, \mathrm{s}(z)), \mathrm{g}(\mathrm{s}(y), x))$$

We choose $A = \mathbf{N}^2$ together with the symbol interpretations:

$$[\mathrm{h}](\boldsymbol{x_0}, \boldsymbol{x_1}) = \begin{pmatrix} 3 & 1 \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{x_0} + \begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix} \cdot \boldsymbol{x_1} + \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$
$$[\mathrm{g}](\boldsymbol{x_0}, \boldsymbol{x_1}) = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{x_0} + \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \cdot \boldsymbol{x_1}$$
$$[\mathrm{s}](\boldsymbol{x_0}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \boldsymbol{x_0} + \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

Let $\alpha : \mathcal{X} \to A$ be arbitrary; write $\alpha(x) = \boldsymbol{x}$, $\alpha(y) = \boldsymbol{y}$, $\alpha(z) = \boldsymbol{z}$ and $\alpha(u) = \boldsymbol{u}$. Then we obtain

$$[\mathrm{h}(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)), \alpha]$$
$$=$$
$$\begin{pmatrix} 7 & 3 \\ 2 & 1 \end{pmatrix} \cdot \boldsymbol{x} + \begin{pmatrix} 5 & 1 \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{y} + \begin{pmatrix} 5 & 1 \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{z} + \begin{pmatrix} 7 & 3 \\ 2 & 1 \end{pmatrix} \cdot \boldsymbol{u} + \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$
$$>$$
$$\begin{pmatrix} 7 & 3 \\ 2 & 1 \end{pmatrix} \cdot \boldsymbol{x} + \begin{pmatrix} 5 & 1 \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{y} + \begin{pmatrix} 5 & 1 \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{z} + \begin{pmatrix} 7 & 3 \\ 2 & 1 \end{pmatrix} \cdot \boldsymbol{u} + \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$
$$=$$
$$[\mathrm{h}(\mathrm{g}(u, \mathrm{s}(z)), \mathrm{g}(\mathrm{s}(y), x)), \alpha].$$

By Theorem 2 we conclude that the system is terminating.

Just as in this example, in general we conclude $[\ell,\alpha] > [r,\alpha]$ for arbitrary $\alpha : \mathcal{X} \to A$ if we have a strict decrease in the first vector coefficient, and $\geq$ for all matrix coefficients and all other vector coefficients.

We conclude this section by an example of relative termination.

Example 2. Define R, S as follows; we want to prove $\mathsf{SN}(R/S)$.

$$\begin{aligned} R &= \{\ \mathrm{f}(\mathrm{a},\mathrm{g}(y),z) \to \mathrm{f}(\mathrm{a},y,\mathrm{g}(y)),\ \mathrm{f}(\mathrm{b},\mathrm{g}(y),z) \to \mathrm{f}(\mathrm{a},y,z),\ \mathrm{a} \to \mathrm{b}\ \} \\ S &= \{\ \mathrm{f}(x,y,z) \to \mathrm{f}(x,y,\mathrm{g}(z))\ \}. \end{aligned}$$

We choose the following symbol interpretations:

$$[\mathrm{a}] = \begin{pmatrix}1\\0\end{pmatrix} \qquad [\mathrm{b}] = \begin{pmatrix}0\\0\end{pmatrix}$$

$$[\mathrm{f}](\boldsymbol{x_0},\boldsymbol{x_1},\boldsymbol{x_2}) = \begin{pmatrix}1&0\\0&0\end{pmatrix}\cdot\boldsymbol{x_0} + \begin{pmatrix}1&2\\0&0\end{pmatrix}\cdot\boldsymbol{x_1} + \begin{pmatrix}1&0\\0&0\end{pmatrix}\cdot\boldsymbol{x_2} + \begin{pmatrix}0\\0\end{pmatrix}$$

$$[\mathrm{g}](\boldsymbol{x}) = \begin{pmatrix}1&0\\1&1\end{pmatrix}\cdot\boldsymbol{x} + \begin{pmatrix}0\\1\end{pmatrix}$$

Thereby all rules in $R \cup S$ are weakly decreasing, i.e. all matrix coefficients in the left hand side are greater or equal to the corresponding coefficients in the right hand side. Moreover, all upper left matrix coefficients are nonzero and the rules in R are strictly decreasing in the first coefficient. Hence by Theorem 2 all rules from R may be removed proving $\mathsf{SN}(R/S)$.

5 Top Reduction and Dependency Pairs

For a one-sorted TRS R a symbol $f \in \Sigma$ is called a *defined symbol* if f is the root symbol of a left hand side of a rule of R. For every defined symbol $f \in \Sigma$ a new marked symbol $f_\#$ is added having the same arity as f. If $f(s_1,\ldots,s_n) \to C[g(t_1,\ldots,t_m)]$ is a rule in R and g is a defined symbol of R, then the rewrite rule $f_\#(s_1,\ldots,s_n) \to g_\#(t_1,\ldots,t_m)$ is called a *dependency pair* of R. The TRS consisting of all dependency pairs of R is denoted by $\mathsf{DP}(R)$. We consider these TRSs R and $\mathsf{DP}(R)$ to be S-sorted for $S = \{s,\#\}$, and every $f \in \Sigma$ has arity $((s,\ldots,s),s)$ and its marked version $f_\#$ has arity $((s,\ldots,s),\#)$.

The main theorem about dependency pairs is the following, due to Arts and Giesl, [1].

Theorem 3. *Let R be a one-sorted TRS. Then* $\mathsf{SN}(R)$ *if and only if* $\mathsf{SN}(\mathsf{DP}(R)_{top}/R)$.

We will use this theorem for proving $\mathsf{SN}(R)$ by proving $\mathsf{SN}(\mathsf{DP}(R)_{top}/R)$ using part *2* of Theorem 2.

For doing so by matrix interpretations we fix a dimension d as before and construct a weakly monotone algebra $(A,[\cdot],>,\gtrsim)$ in which $A_s = \mathbf{N}^d$ and $A_\# = \mathbf{N}$. The relation $\gtrsim$ on $A_s = \mathbf{N}^d$ is defined as before:

$$(v_1,\ldots,v_d) \gtrsim (u_1,\ldots,u_d) \iff v_i \gtrsim u_i \text{ for all } i = 1,2,\ldots,d;$$

the relation $\gtrsim$ on $A_\# = \mathbf{N}$ is the usual $\geq$ on $\mathbf{N}$. However, for $>$ on $A_s = \mathbf{N}^d$ we now choose another relation as before: we choose $>$ to be the empty relation. The relation $>$ on $A_\# = \mathbf{N}$ is the usual $>$ on $\mathbf{N}$. All requirements for $>$ and $\gtrsim$ from Definition 1 trivially hold.

For the interpretation $[f]$ of a symbol $f \in \Sigma$ of arity $n \geq 0$ we define

$$[f](\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}) \;=\; F_1\boldsymbol{v_1} + \cdots + F_n\boldsymbol{v_n} + \boldsymbol{f}$$

for n matrices $F_1, F_2, \ldots, F_n$ over $\mathbf{N}$ of size $d \times d$, and a vector $\boldsymbol{f} \in \mathbf{N}^d$. Note that now we do not require any more that the upper left elements of the matrices are positive. For the interpretation $[f_\#]$ of a marked symbol $f_\#$ corresponding to f of arity $n \geq 0$ we define

$$[f_\#](\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}) \;=\; \boldsymbol{f_1}\boldsymbol{v_1} + \cdots + \boldsymbol{f_n}\boldsymbol{v_n} + c_f$$

for n row vectors $\boldsymbol{f_1}, \ldots, \boldsymbol{f_n}$ over $\mathbf{N}$ of size d, and a constant $c_f \in \mathbf{N}$. Here $\boldsymbol{f_i}\boldsymbol{v_i}$ denotes the inner product, corresponding to matrix multiplication of a row vector by a column vector.

As before $[f]$ is monotonic with respect to $\gtrsim$, and monotonicity with respect to $>$ is trivial since $>$ is empty. The same holds for $f_\#$. By choosing all $[f]$ and $f_\#$ of this shape all requirements of a weakly monotone algebra are fulfilled.

In order to apply Theorem 2, part *2*, for rules in R we check whether $[\ell, \alpha] \gtrsim [r, \alpha]$ for all $\alpha : \mathcal{X} \to A$ for given rewrite rules as before. Checking whether $[\ell, \alpha] > [r, \alpha]$ for all α is only required for rules $\ell \to r$ in $\mathsf{DP}(R)$ being of sort $\#$. This restriction can be written as $\boldsymbol{l}\boldsymbol{x} + c_l > \boldsymbol{r}\boldsymbol{x} + c_r$ for every vector $\boldsymbol{x}$ over $\mathbf{N}$, being equivalent to $\boldsymbol{l} \gtrsim \boldsymbol{r} \ \wedge\ c_l > c_r$. Similarly, for rules $\ell \to r$ in $\mathsf{DP}(R)$ the requirement $[\ell, \alpha] \gtrsim [r, \alpha]$ for all α is equivalent to $\boldsymbol{l} \gtrsim \boldsymbol{r} \ \wedge\ c_l \gtrsim c_r$.

It is also possible to keep the treatment of $\mathsf{SN}(\mathsf{DP}(R)_{top}/R)$ one-sorted on vectors of size d, choosing $>$ to be the strict part of $\gtrsim$. However, then the search space is much bigger since for every $f_\#$ n matrices of size $d \times d$ plus a vector have to be chosen, instead of n vectors of size d plus a constant, where n is the arity of f. Every termination proof in this one-sorted setting also yields a termination proof in the two-sorted setting as presented here, with the same bound on matrix- and vector elements. This can be seen as follows. If there is a proof in the one-sorted setting then for at least one dependency pair the interpretation of the lhs strictly exceeds the interpretation of the rhs. Since $>$ is the strict part of $\gtrsim$, there is at least one dimension in which strict inequality appears. Then by eliminating all other dimensions an interpretation in our two-sorted setting is found by which this particular dependency pair can be removed. By repeating the argument, the full termination proof in the one-sorted setting can be mimicked in our two-sorted setting. So the two-sorted approach is as powerful but yields much smaller search spaces, by which this two-sorted approach is preferred.

Example 3. Consider the TRS consisting of the following rule.

$$\mathrm{g}(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)) \to \mathrm{g}(\mathrm{g}(y, z), \mathrm{g}(x, \mathrm{s}(u)))$$

Using the dependency pairs transformation we get 3 dependency pairs:

1. $\mathrm{g}_\#(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)) \to \mathrm{g}_\#(\mathrm{g}(y, z), \mathrm{g}(x, \mathrm{s}(u)))$
2. $\mathrm{g}_\#(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)) \to \mathrm{g}_\#(y, z)$
3. $\mathrm{g}_\#(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)) \to \mathrm{g}_\#(x, \mathrm{s}(u))$

The dependency pairs 2 and 3 can easily be removed by counting the symbols. That is using $[\mathrm{g}_\#](x, y) = [\mathrm{g}](x, y) = 1 + x + y$ and $[\mathrm{s}](x) = x + 1$ as polynomial interpretation over $\mathbf{N}$. So the original rule and the first dependency pair remain. We choose the following interpretation with dimension $d = 2$ (i.e. $A_s = \mathbf{N}^2$, $A_\# = \mathbf{N}$).

$$\begin{aligned}
[\mathrm{g}_\#](\boldsymbol{x_0}, \boldsymbol{x_1}) &= (1,\ 0) \cdot \boldsymbol{x_0} + (0,\ 1) \cdot \boldsymbol{x_1} \\
[\mathrm{g}](\boldsymbol{x_0}, \boldsymbol{x_1}) &= \begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} \cdot \boldsymbol{x_0} + \begin{pmatrix} 1\,0 \\ 0\,1 \end{pmatrix} \cdot \boldsymbol{x_1} \\
[\mathrm{s}](\boldsymbol{x_0}) &= \begin{pmatrix} 1\,0 \\ 0\,0 \end{pmatrix} \cdot \boldsymbol{x_0} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}
\end{aligned}$$

For the original rule $\mathrm{g}(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)) \to \mathrm{g}(\mathrm{g}(y, z), \mathrm{g}(x, \mathrm{s}(u)))$ we obtain

$$\begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} \cdot \boldsymbol{x} + \begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} \cdot \boldsymbol{y} + \begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} \cdot \boldsymbol{z} + \begin{pmatrix} 1\,0 \\ 0\,1 \end{pmatrix} \cdot \boldsymbol{u} + \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\gtrsim$$

$$\begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} \cdot \boldsymbol{x} + \begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} \cdot \boldsymbol{y} + \begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} \cdot \boldsymbol{z} + \begin{pmatrix} 1\,0 \\ 0\,0 \end{pmatrix} \cdot \boldsymbol{u} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and for the remaining dependency pair $\mathrm{g}_\#(\mathrm{g}(\mathrm{s}(x), y), \mathrm{g}(z, u)) \to \mathrm{g}_\#(\mathrm{g}(y, z), \mathrm{g}(x, \mathrm{s}(u)))$ we obtain

$$(1,\ 0) \cdot \boldsymbol{x} + (1,\ 0) \cdot \boldsymbol{y} + (1,\ 0) \cdot \boldsymbol{z} + (0,\ 1) \cdot \boldsymbol{u} + (\mathbf{1})$$

$$>$$

$$(1,\ 0) \cdot \boldsymbol{x} + (1,\ 0) \cdot \boldsymbol{y} + (1,\ 0) \cdot \boldsymbol{z} + (0,\ 0) \cdot \boldsymbol{u} + (\mathbf{0}).$$

So all rules are weakly decreasing and the dependency pair is strictly decreasing and thus can be removed. Hence the system is terminating.

Note that the given interpretation cannot be used to prove termination directly by Lemma 1. All the upper left matrix elements are nonzero, but $(1,1)^T \not> (1,0)^T$. In Section 7 we will see that in experiments it often happens similarly that this dependency pair approach succeeds where the basic matrix approach from Section 4 fails.

6 Implementation

The method described in the previous sections has been implemented as follows.

The basic algorithm finds a matrix interpretation that allows to remove rules from a termination problem. It is called repeatedly until all rules have been removed.

Algorithm *Remove*:

- inputs
  - a pair of rewrite systems (R, S) over signature Σ
  - a flag $f \in \{\text{Full}, \text{Top}\}$
  - numbers d, b, b'
- outputs a matrix interpretation $[\cdot]$ such that
  - if $f = \text{Full}$, then the interpretation fulfills the conditions of Theorem 2, part *1*, for a non-empty TRS R';
  - if $f = \text{Top}$, then the interpretation fulfills the conditions of Theorem 2, part *2*, for a non-empty TRS R';
  - the interpretation $[\cdot]$ uses matrices of dimension $d \times d$;
  - all the coefficients in the matrices in the interpretations of operation symbols are in the range $0 \ldots 2^b - 1$;
  - all the coefficients in the in the matrices in the interpretations of rules are in the range $0 \ldots 2^{b'} - 1$.

It may be useful to choose $b < b'$. For instance, if $b = 2$ and $b' = 3$ then the algorithm searches for matrices with coefficients < 4 as the interpretations of the operation symbols, but allows coefficients up to 7 in the matrices obtained by multiplying these basic matrices guided by the shape of the rules.

As described in Sections 4 and 5 the conditions for Theorem 2 give rise to constraints on coefficients in vectors and matrices that constitute the interpretations of the rules.

The implementation of the algorithm has two stages: the first stage produces a system I of inequalities, representing these constraints. The second stage solves this constraint system I by translation to a boolean satisfiability problem F.

We stress that the constraint system I consists of inequalities between polynomials of the unknowns. The maximal degree of these polynomials is the maximal depth of a term in a rewrite rule. The number of unknowns depends linearly on the size of the alphabet and quadratically on the dimension of the vector space we use. The number of the inequalities is quadratic in the dimension and linear in the number of rules. Because of the size and the non-linearity of this system, there is no hope for a feasible exact algorithm that solves it.

By putting the bounds b, b' on the range of the variables, the problem has become finite. This finite problem is translated into propositional logic. Each variable from I is then represented by a list of b boolean variables, giving its binary representation. To represent intermediate results (partial sums and products), we need additional constraint variables (translated into bit strings of length b').

Then the formula F is transformed into conjunctive normal form, and we call a SAT solver to find a satisfying assignment. We use SatELiteGTI, [3], the winner of last year's SAT competition. But our translators are not specific to that solver since we use a system-independent data exchange format. E. g. we checked with ZChaff ([12]) and got nearly identical results. Information about the 2005 SAT competition and these tools is obtained via `http://www.satcompetition.org/`.

From the satisfying assignment for F a satisfying assignment for the original system I is constructed. This gives the matrices and vectors for the symbol interpretations. The rule interpretations are re-calculated to double-check that all constraints do really hold and that indeed a nonempty set R' of rules can be removed according to Theorem 2.

If the solver does not find a satisfying assignment within a given time bound, the process is repeated by either giving larger bounds for the coefficients or larger dimension for the vector space.

While this gives the general idea, quite some effort has to be invested to organize the repeated attempts in such a manner that all potentially successful parameter combinations are actually tried within the given time bound. For instance, we start with the direct matrix method using dimension one with 5 bits for coefficients, followed by dimension two with 3 bits for coefficients, both 5 seconds time-out. Afterward we do a dependency pairs transformation and use matrix interpretations of dimension one, up to dimension 4, with 2 bits for coefficients, 3 bits for intermediates, increasing the time-out stepwise.

To give an impression of this search and the size of the resulting formula, consider the TRS consisting of the following rules.

$$\mathrm{h}(x, \mathrm{c}(y, z)) \to \mathrm{h}(\mathrm{c}(\mathrm{s}(y), x), z)$$

$$\mathrm{h}(\mathrm{c}(\mathrm{s}(x), \mathrm{c}(\mathrm{s}(0), y)), z) \to \mathrm{h}(y, \mathrm{c}(\mathrm{s}(0), \mathrm{c}(x, z)))$$

For smaller dimensions no solution is found, but by choosing dimension $d = 3$ and 2 bits per coefficient suitable interpretations are found by which the second rule can be removed. Termination of the remaining rule is easily shown by a one-dimensional interpretation.

For the main step in this proof, i.e., removing the second rule, the translation of the constraint problem needs 8.000 boolean variables and 40.000 propositional clauses. A satisfying assignment is found by SatELiteGTI in around 5 seconds on a current personal computer.

The translation of one binary multiplication (where the arguments have 3 bits and the result has 6 bits) needs about 150 clauses. One can exchange variables for clauses, to a certain extent.

We developed two independent implementations:

– as part of Matchbox [13], by Waldmann, written in Haskell, and
– as part of Jambox, [4], by Endrullis, written in Java.

This allows to double-check our results. We each use slightly different algorithms that produce formulas of different sizes. It is not automatically the case that the smaller formula is better for the solver. In some cases, the solver will find a solution for a larger formula earlier than for a smaller one.

7 Performance Measurements

In this section we will analyze the performance of the matrix method under various setting on the TRS part of the Termination Problem Database 2005 (TPDB). This problem set was the basis of the 2005 Termination Competition and is available via [2]. It consists of 773 TRS, among which 588 could be proved to be terminating by any of the six participating tools; the rest both contains non-terminating TRSs and TRSs for which the termination behavior is unknown or only established by a human.

By *direct method* we mean pure matrix interpretations, i.e. without usage of any other termination methods like dependency pairs. Likewise the method with dependency pairs stands for the combination of matrix interpretations with the dependency pairs framework. A huge amount of methods has been developed for the dependency pairs framework. In our implementation we restrict to the most basic methods, since our goal is to analyze the strength of the matrix method. In particular, we use dependency graph approximation and the usable rules criterion [5,8], the sub-term criterion [8], and compute strongly connected components as in [9]. Finally, dependency pairs + stands for the extension by the transformation of applicative TRSs into functional form as described in [6], and rewriting of right hand sides [16]. Both techniques are non-branching syntactic transformations, to be used as preprocessing.

We want to emphasize that we did not apply any of the following techniques: recursive path order, argument filtering and semantic labelling, as they were considered sometimes to be essential for any serious termination tool.

The following table presents our results.

method	dimension d	initial bits b	result bits b'	cumulative YES score
direct	1	4	5	141
direct	2	2	3	219
direct	3	3	4	225
dependency pairs	1	4	5	433
dependency pairs	2	1	2	503
dependency pairs	2	2	3	505
dependency pairs	3	2	3	507
dependency pairs	4	2	3	509
dependency pairs +	4	2	3	538

For these results we took the time limit of 1 minute, just like in the Termination Competition. However, this time was hardly ever consumed; the average computation time for all proofs is around 1 second. The full results, including all proofs generated by Jambox, are available via

`http://joerg.endrullis.de/ijcar06/`.

For the following 6 systems our approach found termination proofs where all participating tools in the 2005 competition failed: `TRCSR-Ex1-2-Luc02c-GM`, `TRCSR-Ex14-AEGL02-C`, `TRCSR-Ex1-GL02a-C`, `TRCSR-Ex4-7-15-Bor03-C`, `TRCSR-Ex49-GM04-C` and `TRCSR-Ex6-9-Luc02c-GM`. So by adding our approach the total score of 588 for all tools would increase to 594.

We also applied our approach the subcategory of relative termination in TPDB, on which the 2005 competition also run. The winner in this subcategory was TPA with a score of 23; our approach would have yielded a second place with a score of 20. Among these 20 proofs 10 are done with dimension one, 8 with dimension two and 2 with dimension three.

8 Conclusions

The idea of using matrix interpretations for termination proofs for string rewriting was developed by Hofbauer and Waldmann [11]. It allowed them to prove termination for $\{aa \to bc, bb \to ac, cc \to ab\}$. In this paper we showed how to extend this approach to term rewriting successfully. A crucial ingredient is taking linear combinations of matrix interpretations for symbols of arity > 1.

In the results on the benchmark database TPDB we see a big jump when increasing the dimension from 1 (representing linear polynomial interpretations) to 2. Increasing the dimension from 2 to higher values only yields a minor improvement, while then the sizes of the satisfiability formulas strongly increase. By adding the dependency pairs approach an enormous jump is achieved: then using only linear polynomial interpretations ($d = 1$) already reaches a score of 433 points. In the Termination Competition 2005 this would have been a remarkable third place. Finally, our highest score of 538 for dependency pairs + would have yielded a second place in this competition: still below the winning score of 576 for AProVE [7], but significantly better than the second score of 509 for TTT [10].

We like to stress that among the 538 TRSs for which termination was proved by our tool, for several (6) of them all six tools from the 2005 competition failed.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. Termination Competition. `http://www.lri.fr/~marche/termination-competition/`.
3. N. Eén and A. Biere. Effective preprocessing in sat through variable and clause elimination. In F. Bacchus and T. Walsh, editors, *Proc. 8th Int. Conf. Theory and Applications of Satisfiability Testing SAT 2005*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75, Berlin, 2005. Springer-Verlag.
4. J. Endrullis. Jambox: Automated termination proofs for string rewriting. `http://joerg.endrullis.de/`, 2005.
5. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2004)*, volume 3452 of *Lecture Notes in Computer Science*, pages 301–331. Springer, 2005.
6. J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proceedings of the 5th International Workshop on Frontiers of Combining Systems (FroCoS 2005)*, volume 3717 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 2005.
7. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with AProVE. In V. van Oostrom, editor, *Proceedings of the 15th Conference on Rewriting Techniques and Applications (RTA)*, volume 3091 of *Lecture Notes in Computer Science*, pages 210–220. Springer, 2004.

8. N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In V. van Oostrom, editor, *Proceedings of the 15th Conference on Rewriting Techniques and Applications (RTA)*, volume 3091 of *Lecture Notes in Computer Science*, pages 249–268. Springer, 2004.
9. N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199:172–199, 2005.
10. N. Hirokawa and A. Middeldorp. Tyrolean termination tool. In J. Giesl, editor, *Proceedings of the 16th Conference on Rewriting Techniques and Applications (RTA)*, volume 3467 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 2005.
11. D. Hofbauer and J. Waldmann. Proving termination with matrix interpretations. In F. Pfenning, editor, *Proceedings of the 17th Conference on Rewriting Techniques and Applications (RTA)*, Lecture Notes in Computer Science. Springer, 2006.
12. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference DAC 2001*, pages 530–535. ACM, 2001.
13. J. Waldmann. Matchbox: A tool for match-bounded string rewriting. In V. van Oostrom, editor, *Proceedings of the 15th Conference on Rewriting Techniques and Applications (RTA)*, pages 85–94, Berlin, 2004. Springer-Verlag.
14. H. Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.
15. H. Zantema. Termination. In *Term Rewriting Systems, by Terese*, pages 181–259. Cambridge University Press, 2003.
16. H. Zantema. Reducing right-hand sides for termination. In A. Middeldorp, V. van Oostrom, F. van Raamsdonk, and R. de Vrijer, editors, *Processes, Terms and Cycles: Steps on the Road to Infinity:Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*, volume 3838 of *Lecture Notes in Computer Science*, pages 173–197, Berlin, 2005. Springer-Verlag.