Reducing right hand sides for termination

Hans Zantema

Department of Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
h.zantema@tue.nl

Abstract. We propose two transformations on term rewrite systems (TRSs) based on reducing right hand sides: one related to the transformation order and a variant of dummy elimination. Under mild conditions we prove that the transformed system is terminating if and only if the original one is terminating. Both transformations are very easy to implement, and make it much easier to prove termination of some TRSs automatically.

1 Introduction

Developing techniques for proving termination of TRSs is a challenging research area already for a long time. In recent years the emphasis in this area has shifted towards implementation: for new techniques to prove termination it is no longer sufficient that they can be used to prove termination of particular TRSs in theory, but also tools should be able to use these techniques to prove termination fully automatically. Several tools have been developed for this goal, and there is a yearly competition in which all of these tools are applied to an extensive set of examples (TPDB, the termination problem data base), and compared, see

`http://www.lri.fr/~marche/termination-competition/`.

In this paper we present two transformations on TRSs for which termination of the original TRS can be concluded from termination of the transformed TRS. Since these transformations are very easy to implement and proving termination of the transformed TRS by standard techniques is often much simpler than proving termination of the original TRS, they are very suitable to be used as preprocessing steps before using any of the tools.

Both transformations do not change left hand sides, and reduce right hand sides. In the first transformation, related to the *transformation ordering*, [3], this is done by rewriting right hand sides using the same TRS. So here it is assumed that at least one right hand side of a rule is not in normal form. In the second transformation, being a variant of *dummy elimination*, [7], the right hand sides are decomposed with respect to a special symbol (a *dummy symbol*) that occurs in a right hand side but in no left hand side.

The technique of rewriting right hand sides was considered before in [10], but there it was required that the whole TRS is non-overlapping, while our requirements are much weaker. Our approach is based on the transformation ordering from [3], presented in a more abstract setting in [15]. The first approach to implement this was described in [12]. In order to use this technique for rewriting right hand sides we had to adjust the underlying theory. In this paper all required theory is included.

For our present variant of dummy elimination the main theorem states that the original TRS is terminating if the transformed TRS is terminating, just as in [7]. However, in case of left-linearity we also have the converse, as we prove in this paper, by which our present variant is often stronger than the earlier version from [7].

For string rewriting the techniques described in this paper have been implemented in TORPA: Termination of Rewriting Proved Automatically, see

`http://www.win.tue.nl/~hzantema/torpa.html`, a tool developed by the author, being the winner in the above mentioned competition in the category of string rewriting, both in 2004 and 2005.

For term rewriting the techniques described in this paper have been implemented in the tool TPA: Termination Proved Automatically, written by Adam Koprowski, see

`http://www.win.tue.nl/tpa`.

In the above mentioned termination competition this tool was third among 6 participants in the category of term rewriting. This was before both techniques of this paper were implemented.

The organization of this paper is as follows. First in Section 2 the preliminaries are given, both for abstract rewriting and term rewriting. Then in Section 3 the theory and implementation of the technique of rewriting right hand sides is presented. Next, Section 4 treats dummy elimination, first for term rewriting and then for string rewriting. To derive the result for string rewriting from the result for term rewriting in Subsection 4.2 we prove and apply a general theorem. A TRS having no symbols of arity greater than one is transformed to an SRS simply by ignoring all parentheses and variable symbols. The theorem states that the TRS is terminating if and only if the SRS is terminating. Finally, in Section 5 we give some conclusions.

2 Preliminaries

2.1 Abstract rewriting

In the following R, S and T are arbitrary binary relations on a fixed set. In the applications they will correspond to rewrite relations of TRSs. We write a dot symbol for relational composition, i.e., one has $t(R \cdot S)t'$ if and only if there exists a t'' such that tRt'' and $t''St'$. We write R^+ for the transitive closure of R and R^* for the reflexive transitive closure of R, and we write R^{-1} for the inverse of R. Further we write $R \subseteq S$ if tRt' implies tSt'. Clearly, if $R \subseteq S$ then $R \cdot T \subseteq S \cdot T$ and $T \cdot R \subseteq T \cdot S$.

Using these notations confluence of a relation R, written as $\mathsf{CR}(R)$, can be expressed shortly as $(R^{-1})^* \cdot R^* \subseteq R^* \cdot (R^{-1})^*$. Similarly, local confluence of a relation R, written as $\mathsf{WCR}(R)$, can be expressed as $R^{-1} \cdot R \subseteq R^* \cdot (R^{-1})^*$.

We write $\infty(t, R)$ if there exists an infinite sequence $tRt_1Rt_2Rt_3R\cdots$. Such an infinite sequence is called an *infinite R-reduction*. A relation R is called *terminating* on t, written as $\mathsf{SN}(t, R)$, if not $\infty(t, R)$. A relation R is called *terminating*, written as $\mathsf{SN}(R)$, if it is terminating on every t, i.e., no infinite R-reduction exists at all.

For a terminating relation R we can apply *induction on R*, i.e. if for all elements t we can prove

$$(\forall t' : (tRt' \Rightarrow P(t'))) \Rightarrow P(t)$$

then we may conclude that the property $P(t)$ holds for all t. The assumption $\forall t' : (tRt' \Rightarrow P(t'))$ is called the *induction hypothesis*.

We write R/S for $S^* \cdot R \cdot S^*$. For instance, $(R/S)^+$ describes a sequence of $R \cup S$-steps containing at least one R-step, so

$$(R/S)^+ \;=\; S^* \cdot R \cdot (R \cup S)^* \;=\; (R \cup S)^* \cdot R \cdot S^*.$$

2.2 Term rewriting

Write $\mathsf{Var}(t)$ for the set of variables in a term t. A *rewrite rule* is a pair of terms (ℓ, r), written as $\ell \to r$, such that ℓ is not a variable and $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$. The terms ℓ, r are called the *left hand side* (lhs) and the *right hand side* (rhs) of the rule $\ell \to r$, respectively. A rule $\ell \to r$ is called *left-linear* if every variable occurs at most once in ℓ. A rule $\ell \to r$ is called *non-erasing* if $\mathsf{Var}(r) = \mathsf{Var}(\ell)$.

A *term rewrite system* (TRS) is defined to be a set of rewrite rules. A TRS is called left-linear if all its rules are left-linear. A TRS is called non-erasing if all its rules are non-erasing.

A term t rewrites to a term u w.r.t. a TRS $\mathcal{R}$, notation $t \to_{\mathcal{R}} u$, if there is a rule $\ell \to r$ in $\mathcal{R}$, a context C and a substitution σ such that $t = C[\ell\sigma]$ and $u = C[r\sigma]$. A TRS $\mathcal{R}$ is said to be terminating, confluent or locally confluent (notation: $\mathsf{SN}(\mathcal{R}), \mathsf{CR}(\mathcal{R}), \mathsf{WCR}(\mathcal{R})$) if the corresponding property holds for the binary relation $\to_{\mathcal{R}}$ on terms. Basic techniques to prove termination of TRSs include recursive path order ([5]) and polynomial interpretations ([4]). More envolved techniques in which TRSs are first transformed before basic techniques are applied include semantic labelling ([14]) and dependency pairs ([1]). For an overview of techniques for proving termination of TRSs see [15]. For a general introduction to rewriting see [2].

The TRS $\mathcal{E}mb$ is defined to consist of all rules of the shape $f(x_1, \ldots, x_n) \to x_i$. A rule $\ell \to r$ is called *self-embedding* if $r \to^*_{\mathcal{E}mb} \ell$. A TRS $\mathcal{R}$ is called *simply terminating* if $\mathcal{R} \cup \mathcal{E}mb$ is terminating. It is obvious that a TRS containing a self-embedding rule is not simply terminating. It is well-known ([13, 15]) that termination of a TRS can not be proved by recursive path order or polynomial interpretations if the TRS is not simply terminating.

Two non-variable terms t, u are said to have overlap if there are substitutions σ, τ such that either $t'\sigma = u\tau$ for a non-variable subterm t' of t, or $t\sigma = u'\tau$ for a non-variable subterm u' of u. Two rules $\ell_1 \to r_1$ and $\ell_2 \to r_2$ are said to have non-trivial overlap if either the rules are distinct and ℓ_1 and ℓ_2 have overlap, or the rules are equal and there are substitutions σ, τ such that $t'\sigma = \ell_1\tau$ for a non-variable proper subterm t' of ℓ_1. Here properness is essential to exclude the trivial overlap caused by $\ell_1\sigma = \ell_1\tau$ for $\sigma = \tau$. It is well-known that $\mathsf{WCR}(\mathcal{R})$ holds if no two (possibly equal) rules of $\mathcal{R}$ have non-trivial overlap.

3 Rewriting right hand sides

3.1 The theory

Lemma 1. *Let S, T be binary relations satisfying*

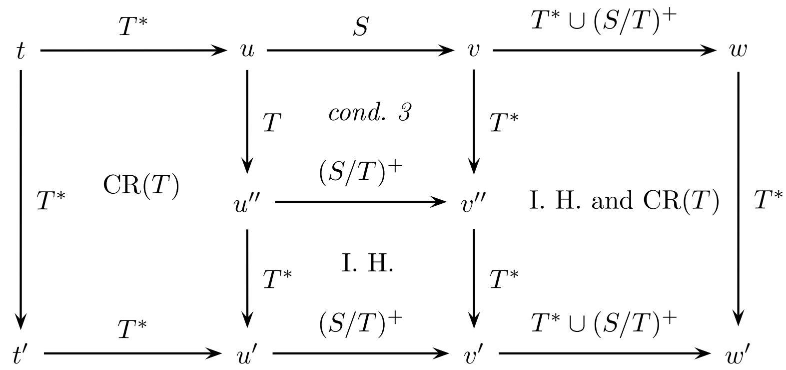
1. *$S \cup T$ is terminating,*
2. *T is locally confluent, and*
3. *$T^{-1} \cdot S \subseteq (S/T)^+ \cdot (T^{-1})^*$.*

Then $(T^{-1})^* \cdot (S/T)^+ \subseteq (S/T)^+ \cdot (T^{-1})^*$.

Proof. We prove by induction on $S \cup T$ that for every t the following holds:

Let $t'(T^{-1})^* t (S/T)^+ w$. Then there exists w' satisfying $t'(S/T)^+ w' (T^{-1})^* w$.

First observe that $(S/T)^+ = T^* \cdot S \cdot (S \cup T)^*$. Since $(S \cup T)^* = T^* \cup (S/T)^+$, we obtain u, v satisfying $tT^*uSv(T^* \cup (S/T)^+)w$. Since T is terminating by *1* and locally confluent by *2*, we have CR(T) by Newman's Lemma: T is confluent. Since tT^*t', tT^*u and CR(T) we obtain u' satisfying $t'T^*u'$ and uT^*u'. If $u' = u$ then we may choose $w' = w$ indeed satisfying $t'(S/T)^+w'(T^{-1})^*w$ and we are done. In the remaining case we have u'' satisfying $uTu''T^*u'$. Applying condition *3* to $u''T^{-1}uSv$ yields v'' satisfying $u''(S/T)^+v''(T^{-1})^*v$. Since tT^*uTu'' we may apply the induction hypothesis to u'', yielding v' satisfying $u'(S/T)^+v'(T^{-1})^*v''$. Now we have vT^*v' and either vT^*w or $v(S/T)^+w$. In the first case CR(T) yields w' satisfying $v'T^*w'(T^{-1})^*w$, in the second case the induction hypothesis applied to v yields w' satisfying $v'(S/T)^+w'(T^{-1})^*w$. In all cases we have $t'T^*u'(S/T)^+v'(T^* \cup (S/T)^+)w'$, so $t'(S/T)^+w'$, and wT^*w', and we are done. Summarized in a picture:



∎

Theorem 2. *Let R, S, T be binary relations satisfying*

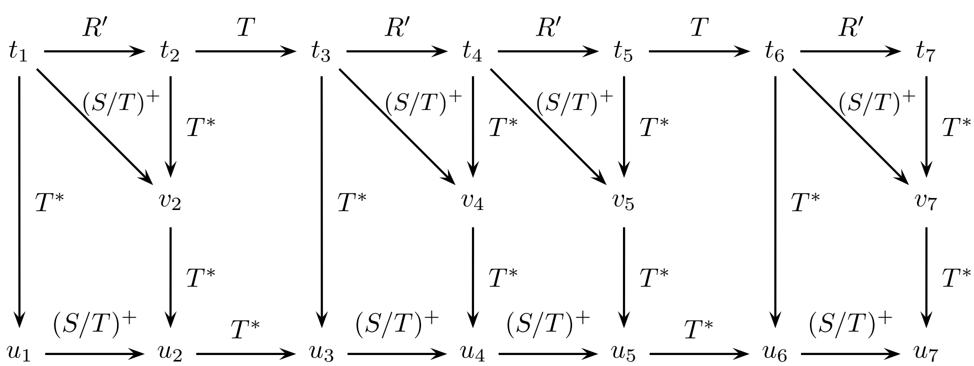
1. *$S \cup T$ is terminating,*
2. *T is locally confluent, and*
3. *$T^{-1} \cdot S \subseteq (S/T)^+ \cdot (T^{-1})^*$.*
4. *$R \subseteq ((S/T)^+ \cdot (T^{-1})^*) \cup T$.*

Then R is terminating.

Proof. Assume R is not terminating. So there is an infinite R-reduction, i.e., a sequence $t_1, t_2, t_3, \ldots$ such that $t_i R t_{i+1}$ for all $i = 1, 2, 3, \ldots$. Write $R' = (S/T)^+ \cdot (T^{-1})^*$. Let $u_1 = t_1$, and define u_i for $i = 2, 3, 4, \ldots$ satisfying $t_i T^* u_i$ in the following way:

– If $t_{i-1} T t_i$ then choose u_i such that $t_i T^* u_i$ and $u_{i-1} T^* u_i$. This can be done since T is confluent, following from Newman's Lemma and conditions *1, 2*.
– Otherwise, by condition *4* we have $t_{i-1} R' t_i$, i.e., there exists v_i satisfying $t_{i-1}(S/T)^+ v_i (T^{-1})^* t_i$. By Lemma 1 we may choose u_i such that $u_{i-1}(S/T)^+ u_i$ and $t_i T^* v_i T^* u_i$.

A typical initial part of this construction is sketched in the following picture:



Since T is terminating by condition *1*, the second case $t_{i-1} R' t_i$ occurs infinitely often. So we have $u_{i-1}(S/T)^+ u_i$ for infinitely many values of i, while for the other values of i we have $u_{i-1} T^* u_i$. Hence $u_1 \to u_2 \to u_3 \to \cdots$ is an infinite $S \cup T$-reduction, contradicting condition *1*, concluding the proof. ∎

Theorem 2 is closely related to the underlying theory in the transformation ordering, [3]. A generalization of this underlying theory is expressed in Theorem 6.5.16 in [15]. In fact this Theorem 6.5.16 coincides with the present Theorem 2 where conditions *3, 4* are replaced by

3'. $T^{-1} \cdot S \subseteq T^* \cdot S \cdot (S \cup T \cup T^{-1})^*$,

4'. $R \subseteq (S/(T \cup T^{-1})^+$

Condition *3'* is a strict weakening of condition *3*. However conditions *4'* and *4* are incomparable, since in condition *4* it is allowed that some R-step is only a single T-step and in condition *4'* it is not. In our application this is essential, therefore this new Theorem 2 was developed rather than applying the earlier result.

Next we apply Theorem 2 to term rewriting. In order to do so we first give a lemma analyzing how applications of non-overlapping rewrite rules commute.

Lemma 3. *Let $\ell_i \to r_i$ be rewrite rules for $i = 1, 2$ for which ℓ_1 and ℓ_2 do not have overlap, and having rewrite relations $\to_1, \to_2$, respectively. Let C be a context and σ, τ be substitutions such that $C[\ell_1\sigma] = \ell_2\tau$. Then $\ell_2 = C_2[x]$ for some context C_2 and some variable x, for which the two reducts $C[r_1\sigma]$ and $r_2\tau$ of $C[\ell_1\sigma] = \ell_2\tau$ satisfy*

$$C[r_1\sigma] \to_1^{n-1} \cdot \to_2 \cdot \leftarrow_1^k r_2\tau,$$

where n, k are the numbers of occurrences of x in ℓ_2, r_2, respectively.

Proof. Since there is no overlap between ℓ_1 and ℓ_2 we can write $C = C_2[D]$ where $\ell_2 = C_2[x]$ for contexts C_2, D and some variable x for which $x\tau = D[\ell_1\sigma]$. Define τ' by $x\tau' = D[r_1\sigma]$, and $y\tau' = y\tau$ for $y \neq x$. By applying the reduction $x\tau \to_1 x\tau'$ to the occurrences of $x\tau$ corresponding to the other $n-1$ occurrences of x in $\ell_2 = C_2[x]$, we obtain $C[r_1\sigma] \to_1^{n-1} \ell_2\tau'$. Conversely we obtain $r_2\tau \to_1^k r_2\tau'$ since x occurs k times in r_2. Combining these observations yields

$$C[r_1\sigma] \to_1^{n-1} \ell_2\tau' \to_2 r_2\tau' \leftarrow_1^k r_2\tau,$$

proving the lemma. □

To sketch a typical example of how Lemma 3 applies, consider $\ell_1 \to r_1$ to be the rule $a \to b$ and $\ell_2 \to r_2$ to be the rule $f(x, x) \to g(x, x, x)$. Let $C = f(\Box, a)$ and $x\tau = a$; since ℓ_1 does not contain variables, σ plays no role. Indeed we have $C[\ell_1\sigma] = f(a, a) = \ell_2\tau$, and

$$C[r_1\sigma] = f(b, a) \to_1 f(b, b) \to_2 g(b, b, b) \leftarrow_1^3 g(a, a, a) = r_2\tau.$$

Now we are ready to give the main theorem.

Theorem 4. *Let $\mathcal{R}$ be a TRS for which a right hand side is not in normal form, i.e., $\mathcal{R}$ contains a rule $\ell \to r$ and a rule of the shape $\ell' \to C[\ell\sigma]$. Assume that $\ell \to r$ is left-linear and non-erasing, $\mathsf{WCR}(\{\ell \to r\})$, and there is no overlap between ℓ and the lhs of any rule of $\mathcal{R} \setminus \{\ell \to r\}$. Let $\mathcal{R}'$ be obtained from $\mathcal{R}$ by replacing the rule $\ell' \to C[\ell\sigma]$ by $\ell' \to C[r\sigma]$. Then $\mathcal{R}$ is terminating if and only if $\mathcal{R}'$ is terminating.*

Proof. The 'only if'-part is immediate from the observation that every $\mathcal{R}'$-step can be mimicked by one or two $\mathcal{R}$-steps: an $\mathcal{R}'$-step applying the rule $\ell' \to C[r\sigma]$ is mimicked by first applying the $\mathcal{R}$-rule $\ell' \to C[\ell\sigma]$ and then the $\mathcal{R}$-rule $\ell \to r$, all other $\mathcal{R}'$-steps are $\mathcal{R}$-steps themselves. It remains to prove the 'if'-part.

First note that the rules $\ell \to r$ and $\ell' \to C[\ell\sigma]$ are distinct, since otherwise the rule $\ell \to C[C[\ell\sigma]\sigma]$ contained in $\mathcal{R}'$ is not terminating.

We apply Theorem 2 for the binary relations

- T being the rewrite relation of the single rule $\ell \to r$,
- S being the rewrite relation of $\mathcal{R}' \setminus \{\ell \to r\}$, and
- R being the rewrite relation of the TRS $\mathcal{R}$.

Termination of the TRS $\mathcal{R}$ is proved by checking all four conditions of Theorem 2.

Condition *1* holds since $S \cup T$ is the rewrite relation of $\mathcal{R}'$ and $\mathcal{R}'$ is assumed to be terminating.

Condition *2* holds by assumption.

For proving condition *3* assume that $uT^{-1}tSv$, i.e., a term t rewrites by T to u and by S to v. We distinguish three cases:

- The T-redex is in parallel with the S-redex. Then we have
$$(u,v) \ \in \ S \cdot T^{-1} \ \subseteq \ (S/T)^{+} \cdot (T^{-1})^{*}.$$
- The T-redex is above the S-redex. Then we apply Lemma 3 for $\ell_2 \to r_2$ being $\ell \to r$, so $\to_2 \, = \, T$, and $\to_1 \, \subseteq \, S$. This yields
$$v = C[r_1\sigma] \ \to_1^{n-1} \cdot \to_2 \ \cdot \ \leftarrow_1^{k} \ r_2\tau = u,$$
where n, k are the numbers of occurrences of x in ℓ, r, respectively. Since $\ell \to r$ is left-linear and non-erasing, we have $k > 0$ and $n = 1$. So
$$(v,u) \ \in \to_2 \cdot \leftarrow_1^{+},$$
hence $(u,v) \in \to_2^{+} \cdot \leftarrow_1 \subseteq \ S^{+} \cdot T^{-1} \ \subseteq \ (S/T)^{+} \cdot (T^{-1})^{*}$.
- The S-redex is above the T-redex. Then we apply Lemma 3 for $\ell_1 \to r_1$ being $\ell \to r$, so $\to_1 \, = \, T$, and $\to_2 \, \subseteq \, S$. This yields
$$u \ = \ C[r_1\sigma] \ \to_1^{*} \cdot \to_2 \ \cdot \ \leftarrow_1^{*} \ r_2\tau \ = \ v,$$
so $(u,v) \ \in \to_1^{*} \cdot \to_2 \cdot \leftarrow_1^{*} \subseteq \ T^{*} \cdot S \cdot (T^{-1})^{*} \ \subseteq \ (S/T)^{+} \cdot (T^{-1})^{*}$.

In all cases we proved $(u,v) \ \in \ (S/T)^{+} \cdot (T^{-1})^{*}$, concluding condition *3*.

Condition *4* is verified by considering all three possibilities for an $\mathcal{R}$-rewrite step $t \to u$.

- If $t \to u$ is an application of the rule $\ell \to r$ then tTu.
- If $t \to u$ is an application of the rule $\ell' \to C[\ell\sigma]$ then $tS \cdot T^{-1}u$ where the S-step is an application of the rule $\ell' \to C[r\sigma]$.
- If $t \to u$ is an application of another rule, then this rule is in $\mathcal{R}' \ \{\ell \to r\}$, so tSu.

In all three cases we conclude $(t,u) \ \in \ (S \cdot (T^{-1})^{*}) \cup T \ \subseteq \ ((S/T)^{+} \cdot (T^{-1})^{*}) \ \cup \ T$, concluding condition *4*. □

One can wonder whether all conditions of Theorem 4 are essential. Indeed they are, as is shown by the following four examples.

In the first example let $\mathcal{R}$ consist of the two rules
$$f(x) \to a, \ \ b \to f(b).$$
Let $\ell \to r$ be the first rule, applicable to the right hand side of the second rule. Then $\mathcal{R}'$ consisting of the rules $f(x) \to a$, $b \to a$ is terminating, while $\mathcal{R}$ is not, and all conditions of Theorem 4 hold except for non-erasingness of $\ell \to r$.

In the second example let $\mathcal{R}$ consist of the two rules
$$a \to b, \ \ a \to a.$$
Let $\ell \to r$ be the first rule, applicable to the right hand side of the second rule. Then $\mathcal{R}'$ consisting of two copies of the rule $a \to b$ is terminating, while $\mathcal{R}$ is not, and all conditions of Theorem 4 hold except for the non-overlappingness condition.

In the third example let $\mathcal{R}$ consist of the three rules

$$f(f(x)) \to g(x), \quad h(x) \to f(f(x)), \quad g(f(a)) \to h(h(a)).$$

Let $\ell \to r$ be the first rule, applicable to the right hand side of the second rule. Then $\mathcal{R}'$ consisting of the three rules

$$f(f(x)) \to g(x), \quad h(x) \to g(x), \quad g(f(a)) \to h(h(a))$$

is terminating by recursive path order using the precedence $f > h > g$. However, $\mathcal{R}$ is not terminating due to the reduction

$$h(h(a)) \to h(f(f(a))) \to f(f(f(f(a)))) \to f(g(f(a))) \to f(h(h(a))).$$

All conditions of Theorem 4 hold except for $\mathsf{WCR}(\{\ell \to r\})$.

In the last example let $\mathcal{R}$ consist of the four rules

$$f(x, x) \to g(x), \quad a \to b, \quad a \to c, \quad f(b, c) \to f(a, a).$$

Let $\ell \to r$ be the first rule, applicable to the right hand side of the last rule. Then $\mathcal{R}'$ is terminating, while $\mathcal{R}$ is not due to the reduction $f(b, c) \to f(a, a) \to f(a, c) \to f(b, c)$. All conditions of Theorem 4 hold except for left-linearity of $\ell \to r$.

A possible generalization of Theorem 4 would be in weakening the restriction of non-overlap to a restriction of critical pairs having a particular kind of common reduct. Moreover, even in case this critical pair condition does not hold one can think of extending T by the normalized versions of the corresponding critical pairs, introducing a kind of completion as in [3, 12]. However, in this paper we want to concentrate on very simple criteria not involving branching choices as is introduced in searching for common reducts of critical pairs in typically non-confluent TRSs.

A variant of Theorem 4 was given by Gramlich in [10]:

Theorem 5. *Let $\mathcal{R}$ be a non-overlapping TRS for which a right hand side is not in normal form, i.e., $\mathcal{R}$ contains a rule $\ell \to r$ and a rule of the shape $\ell' \to C[\ell\sigma]$. Assume that $\ell \to r$ is non-erasing. Let $\mathcal{R}'$ be obtained from $\mathcal{R}$ by replacing the rule $\ell' \to C[\ell\sigma]$ by $\ell' \to C[r\sigma]$. Then $\mathcal{R}$ is terminating if and only if $\mathcal{R}'$ is terminating.*

So here we do not have a left-linearity requirement for $\ell \to r$ any more, but the full TRS $\mathcal{R}$ is required to be non-overlapping, while our Theorem 4 only requires non-overlappingness involving the rule $\ell \to r$. In typical applications to TRSs describing arithmetic and having a rule $p(s(x)) \to x$ to be applied to some rhs, Theorem 5 is only applicable if the TRS is non-overlapping. So this approach fails as soon as overlapping combinations of usual rules like

$$x - 0 \to x, \quad s(x) - s(y) \to x - y, \quad x - x \to 0$$

occur. Our Theorem 4 still applies directly if the TRS contains rules of this shape. Therefore we think that in practice our Theorem 4 is more powerful than Theorem 5.

3.2 Implementation

We propose to use Theorem 4 as a pre-processing phase for any tool for proving termination of TRSs as follows. Let $\mathcal{R}$ be any finite TRS for which termination has to be proved.

Basic procedure:
Check if $\mathcal{R}$ can be written as

$$\mathcal{R} \;=\; \mathcal{R}_0 \;\cup\; \{\ell \to r,\; \ell' \to C[\ell\sigma]\}$$

where $\ell \to r$ is left-linear and non-erasing, and has no non-trivial overlap with any rule of R.
If so, then replace $\mathcal{R}$ by $\mathcal{R}_0 \;\cup\; \{\ell \to r,\; \ell' \to C[r\sigma]\}$, and start again.

Checking whether $\mathcal{R}$ can be written in the given way is straightforward: simply check whether any rhs can be rewritten, and if so, check whether the corresponding rewrite rule $\ell \to r$ is left-linear and non-erasing, and has no non-trivial overlap with any rule of R.

From Theorem 4 it follows that for every step in the basic procedure the replaced TRS is terminating if and only if the original TRS is terminating; note that local confluence of the single rule $\ell \to r$ follows from the property that $\ell \to r$ has no non-trivial overlap with itself. So also if the basic procedure contains a number of steps the resulting TRS is terminating if and only if the original TRS is terminating.

In case $\mathcal{R}$ is terminating, then the basic procedure is terminating too. This can be seen as follows. Assume that the procedure goes on forever, repectively yielding TRSs $\mathcal{R}_1 = \mathcal{R}$, $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$, …. Since every $\mathcal{R}_{i+1}$-step can be mimicked by one or two $\mathcal{R}_i$-steps, as we saw in the 'only if'-part of the proof of Theorem 4, we conclude that $\to_{\mathcal{R}_i} \subseteq \to^+_{\mathcal{R}}$ for every $i = 1, 2, 3, \ldots$. Since $\mathcal{R}_{i+1}$ is obtained from $\mathcal{R}_i$ by applying $\to_{\mathcal{R}_i}$ to a rhs of $\mathcal{R}_i$, we can also obtain $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$ by applying $\to^+_{\mathcal{R}}$ to one of the rhs's. Since there are only finitely many rules, but inifinitely many steps from $\mathcal{R}_i$ to $\mathcal{R}_{i+1}$, there is some rhs of the original TRS $\mathcal{R}$ on which $\to^+_{\mathcal{R}}$ is applied infinitely often, contradicting termination of $\mathcal{R}$.

Unfortunately, in case $\mathcal{R}$ is not terminating then it can be the case that the basic procedure is terminating neither. For instance, if $\mathcal{R}$ consists of the two rules $a \to a,\; b \to a$ then the basic procedure can be applied again yielding $\mathcal{R}$ after one step. This process may go on forever. More generally, let $\mathcal{R}$ be a TRS of the shape $\mathcal{R}_0 \cup \{\ell \to r\}$ in which r admits an infinite $\mathcal{R}_0$-reduction. Then the basic procedure applied to $\mathcal{R}$ may go on forever. A simple way to get a robust implementation of the basic procedure that terminates on every TRS is to put some upper bound on the total number of steps of the basic procedure.

String rewriting can be seen as a particular case of term rewriting in which all symbols have arity 1. Since in this case variables and parentheses are redundant, they are usually omitted.

A tool for automatically proving termination of string rewriting is called TORPA: Termination of Rewriting Proved Automatically, see

`http://www.win.tue.nl/~hzantema/torpa.html`

This tool has been developed by the author. After having ideas in mind for years the actual implementation started in July 2003. Earlier versions of TORPA have been described in [16] (version 1.1), in [17] (version 1.2) and in [18] (version 1.3). The extensive paper [18] also contains a full treatment of all the underlying theory.

Our basic procedure for reducing right hand sides was implemented for string rewriting in the newest version of TORPA, version 1.4 This version of the TORPA tool participated in the termination competition in 2005, and was the winner among the eight participants in the string rewriting category, see

`http://www.lri.fr/~marche/termination-competition/2005/`.

In the text generated by TORPA our technique is called *transformation order*. As an example we consider the string rewriting system consisting of the following five rules

$$f0 \to s0,\;\; d0 \to 0,\;\; ds \to ssdps,\;\; fs \to dfps,\;\; ps \to e,$$

where e represents the empty string. This system describes computation of powers of 2: think of s being successor, p being predecessor, d being doubling and f being exponentiation.[1] It is easy to observe that fs^n0 rewrites to its normal form $s^{2^n}0$ for every $n = 0, 1, 2, \ldots$. The normal form of f^n0 has super-exponential size and requires a super-exponential number of steps to be computed. Note that the system is not simply terminating: both the third and the fourth rule are self-embedding. TORPA yields the following termination proof:

```
TORPA 1.4 is applied to the string rewriting system
f 0  -> s 0
d 0  -> 0
d s  -> s s d p s
f s  -> d f p s
p s  -> e
Choose polynomial interpretation f: lambda x.x+1, rest identity
remove: f 0  -> s 0
Remaining rules:
    d 0  -> 0
    d s  -> s s d p s
    f s  -> d f p s
    p s  -> e

Transformation order: apply rule 4 on rhs of rule 2, result:
d 0  -> 0
d s  -> s s d
f s  -> d f p s
p s  -> e

Transformation order: apply rule 4 on rhs of rule 3, result:
d 0  -> 0
d s  -> s s d
f s  -> d f
p s  -> e


Choose polynomial interpretation p: lambda x.x+1, rest identity
remove: p s  -> e
Remaining rules:
    d 0  -> 0
    d s  -> s s d
    f s  -> d f

Terminating by recursive path order with precedence:
  d>s  f>d
```

For term rewriting (not restricting to string rewriting) our basic procedure has been implemented in the tool TPA, written by Adam Koprowski, see

`http://www.win.tue.nl/tpa`.

In the above mentioned termination competition this tool was third among 6 participants in the category of term rewriting. This was before our basic procedure was implemented.

Let $\mathcal{R}$ be any TRS and let $\mathcal{R}'$ be the result of applying the basic procedure to $\mathcal{R}$. From many examples we observe that proving termination of $\mathcal{R}'$ is much simpler than proving termination of $\mathcal{R}$ directly. We should like to have evidence that it is never the other way around. Since the notion of 'simpler' depends on the unspecified set of techniques to be used, it is hard to make this claim solid. However, by construction we have $\to_{\mathcal{R}'} \subseteq \to_{\mathcal{R}}^{+}$, and the left hand sides of $\mathcal{R}$

[1] The fact that the constant 0 may be treated here as a unary symbol will be justified by Theorem 12

are equal to the left hand sides of $\mathcal{R}'$. Under these conditions for all techniques known by us it is very unlikely that proving termination of $\mathcal{R}'$ may be harder than proving termination of $\mathcal{R}$. Therefore applying our basic procedure as a pre-processing before trying any other tool for proving termination will often increase the power of the tool, and probably never decrease it.

One may wonder whether it is natural to have right hand sides not being in normal form. Of course this is hard to answer since there is no precise definition of natural. To our knowledge the most extensive list of termination problems in term rewriting is TPDB, the Termination Problem Data Base, see

`http://www.lri.fr/~marche/tpdb/`.

This database was used in the above mentioned competition. Restricted to term rewriting (excluding string rewriting, being a separate category) it contains 773 TRSs for which the problem of termination has been posed. They are from a wide scala of origins and application areas. Therefore we think it makes sense to consider these TRSs to get an impression of the applicability of our technique. It turns out that among these 773 TRSs there are 98 TRSs for which not only a rhs is not in normal form, but also the extra conditions are satisfied. So for these TRSs our basic procedure is applicable. For several of them, proving termination of the transformed system is much simpler than proving termination of the original system. For instance, consider the classical TRS describing computation of factorials consisting of the following rules

$$\begin{array}{rlcrl} p(s(x)) & \to x & \qquad & *(0,y) & \to 0 \\ \mathsf{fact}(0) & \to s(0) & & *(s(x),y) & \to +(*(x,y),y) \\ \mathsf{fact}(s(x)) & \to *(s(x),\mathsf{fact}(p(s(x)))) & & +(x,0) & \to x \\ & & & +(x,s(y)) & \to s(+(x,y)). \end{array}$$

This TRS is `D33/21.trs` in the TRS category of TPDB. In the 2005 competition only two of the six participating tools were able to prove termination of this TRS. Note that it is not simply terminating since the rule $\mathsf{fact}(s(x)) \to *(s(x),\mathsf{fact}(p(s(x))))$ is self-embedding. However, by applying our basic procedure this self-embedding rule is replaced by $\mathsf{fact}(s(x)) \to *(s(x),\mathsf{fact}(x))$, which is replaced again by $\mathsf{fact}(s(x)) \to +(*(x,\mathsf{fact}(x)),\mathsf{fact}(x))$. Termination of the resulting TRS is easily concluded by the recursive path order using the precedence $\mathsf{fact} > * > + > s$. Using our basic procedure, the tool TPA was able to prove termination of 6 more TRSs in TPDB than without it, including this factorial system:

```
Reduce right hand side of rule (3) using rule (1) to obtain the
following TRS:
(1) p(s(x)) -> x
(2) fact(0) -> s(0)
(3) fact(s(x)) -> *(s(x),fact(x))
(4) *(0,y) -> 0
(5) *(s(x),y) -> +(*(x,y),y)
(6) +(x,0) -> x
(7) +(x,s(y)) -> s(+(x,y))

Reduce right hand side of rule (3) using rule (5) to obtain the
following TRS:
(1) p(s(x)) -> x
(2) fact(0) -> s(0)
(3) fact(s(x)) -> +(*(x,fact(x)),fact(x))
(4) *(0,y) -> 0
(5) *(s(x),y) -> +(*(x,y),y)
(6) +(x,0) -> x
(7) +(x,s(y)) -> s(+(x,y))
```

```
All the rules of this TRS can be oriented with RPO with the
following precedence:

fact > s
+ > s
fact > *
fact > +
* > +
```

4 Dummy elimination

In the basic procedure based on Theorem 4 right hand sides are rewritten. So a part of such an rhs matches with an lhs. In this section we consider the opposite: we consider rhs's containing symbols that do not occur in lhs's at all. These symbols are called *dummy symbols*. Again we keep the lhs's and reduce the rhs's, but this reduction is done completely different then before: the dummy symbol now is used to split up the rhs into several smaller rhs's, each generating a rule in the transformed TRS, with its lhs kept unchanged. This approach was studied before in [7] and was called *dummy elimination*. An earlier version already appeared in [13]. The main theorem was that if the TRS after applying dummy elimination is terminating, then the original TRS is terminating too. In general the converse is not true. Here we present a modification of dummy elimination for which the same property holds, but for which in case of left-linearity also the converse holds (the transformed TRS is terminating if and only if the original TRS is). Moreover, our new version is more powerful to be used in tools for automatically proving termination of TRSs or SRSs.

4.1 Dummy elimination for term rewriting

Before giving precise definitions first we give an example sketching the general idea. Consider the TRS $\mathcal{R}$ consisting of the single rule

$$f(g(x)) \;\rightarrow\; f(a(g(x))).$$

Here the symbol a is a dummy symbol: it does not occur in any lhs. Intuitively this means that this dummy symbol does not play an essential role in further reductions of the term, and further reductions can be localized as being either affecting the part above the dummy symbol or affecting the part below it. This can be formalized by decomposing the right hand sides into smaller terms in which the dummy acts as a separator. In this case this means that the term $f(h(g(x)))$ is decomposed into two terms $f(\diamond)$ and $b(g(x))$, where $\diamond$ is a fresh constant and b is a fresh unary symbol. The lhs's remain the same. The result is the transformed system $DE_a(\mathcal{R})$, in this example consisting of the two rules

$$\begin{aligned} f(g(x)) &\rightarrow f(\diamond) \\ f(g(x)) &\rightarrow b(g(x)). \end{aligned}$$

The main result states that $DE_a(\mathcal{R})$ is terminating if and only if $\mathcal{R}$ is terminating. The value of this transformation for proving termination automatically is already clear from this example: $\mathcal{R}$ is self-embedding, hence not simply terminating, but its termination follows from termination of $DE_a(\mathcal{R})$ by recursive path order choosing the precedence $f > b$, $g > \diamond$.

In order to give a precise definition for dummy elimination we need some auxiliary definitions. We fix one dummy symbol a of a TRS $\mathcal{R}$. Let n be the arity of a. Choose a fresh constant $\diamond_a$ and a fresh unary symbol b_a, i.e., $\diamond_a$ and b_a do not occur in $\mathcal{R}$. As long as a is fixed, we omit the subscripts, simply writing b and $\diamond$. For any term t we define inductively a term $\mathsf{cap}_a(t)$ and a set of terms $\mathsf{dec}_a(t)$:

$$\begin{array}{lll}
\mathsf{cap}_a(x) & = x & \text{for all } x \in \mathsf{Var}, \\
\mathsf{cap}_a(f(t_1,\ldots,t_k)) & = f(\mathsf{cap}_a(t_1),\ldots,\mathsf{cap}_a(t_k)) & \text{for all } f,\ f \neq a \\
\mathsf{cap}_a(a(t_1,\ldots,t_n)) & = \diamond & \\
\mathsf{dec}_a(x) & = \emptyset & \text{for all } x \in \mathsf{Var}, \\
\mathsf{dec}_a(f(t_1,\ldots,t_k)) & = \bigcup_{i=1}^{k} \mathsf{dec}_a(t_i) & \text{for all } f,\ f \neq a \\
\mathsf{dec}_a(a(t_1,\ldots,t_n)) & = \bigcup_{i=1}^{n} (\mathsf{dec}_a(t_i) \cup \{b(\mathsf{cap}_a(t_i))\}). &
\end{array}$$

Roughly speaking we decompose a term t by using the symbol a as a separator, where occurrences of a are replaced by $\diamond$ and arguments of a are marked by the symbol b. Now the term $\mathsf{cap}_a(t)$ is the rootmost part of this decomposition, while $\mathsf{dec}_a(t)$ is the set of all other parts in this decomposition. Now we define the TRS $DE_a(\mathcal{R})$ for any TRS $\mathcal{R}$ having a as a dummy symbol by

$$DE_a(\mathcal{R}) = \{\ell \to u \mid u = \mathsf{cap}_a(r) \vee u \in \mathsf{dec}_a(r) \text{ for a rule } \ell \to r \in \mathcal{R}\}.$$

The transformation DE_a is called *dummy elimination.*

Theorem 6. *Let a be a dummy symbol in a TRS $\mathcal{R}$ for which $DE_a(\mathcal{R})$ is terminating. Then $\mathcal{R}$ is terminating too.*

Before proving this theorem we give an example slightly more complicated than the one given above, and we recall the earlier dummy elimination theorem. Let the TRS $\mathcal{R}$ consist of the two rules

$$\begin{array}{l}
f(g(x)) \to f(a(g(a(x, f(x))), g(f(x)))) \\
g(f(x)) \to g(g(a(f(x), g(g(x))))).
\end{array}$$

Then $DE_a(\mathcal{R})$ consists of the rules

$$\begin{array}{ll}
f(g(x)) \to f(\diamond) & f(g(x)) \to b(g(f(x))) \\
f(g(x)) \to b(g(\diamond)) & g(f(x)) \to g(g(\diamond)) \\
f(g(x)) \to b(x) & g(f(x)) \to b(f(x)) \\
f(g(x)) \to b(f(x)) & g(f(x)) \to b(g(g(x)))
\end{array}$$

Indeed Theorem 6 is helpful for proving termination of $\mathcal{R}$, since both rules of $\mathcal{R}$ are self-embedding and termination of $DE_a(\mathcal{R})$ is easily proved by recursive path order, choosing the precedence $f > g > b > \diamond$.

In the earlier version from [7] the symbol b was omitted. More precisely, for a TRS $\mathcal{R}$ having a as a dummy symbol the TRS $E(\mathcal{R})$ was defined exactly as $DE_a(\mathcal{R})$, with the only difference that $\mathsf{dec}_a(a(t_1,\ldots,t_n))$ was defined to be $\bigcup_{i=1}^{n}(\mathsf{dec}_a(t_i) \cup \{\mathsf{cap}_a(t_i)\})$ rather than $\bigcup_{i=1}^{n}(\mathsf{dec}_a(t_i) \cup \{b(\mathsf{cap}_a(t_i))\})$. As a consequence, the TRS $E(\mathcal{R})$ is obtained from $DE_a(\mathcal{R})$ by removing all symbols b from it. As the main result we recall:

Theorem 7. *Let a be a dummy symbol in a TRS $\mathcal{R}$ for which $E(\mathcal{R})$ is terminating. Then $\mathcal{R}$ is terminating too.*

For a proof of Theorem 7 we refer to [7] or [6], where a slightly more general version has been treated. An alternative proof has been given in [11], where even the restriction of the dummy not occurring in left hand sides has been weakened slightly. A generalization of this result to rewriting modulo equations has been given in [8].

Now we give the proof of Theorem 6.

Proof. Let a be a dummy symbol of arity n in a TRS $\mathcal{R}$ for which $DE_a(\mathcal{R})$ is terminating. Assume $\mathcal{R}$ is not terminating, so admits an infinite reduction. We define a transformation Φ on terms and TRSs replacing every a by $a(b(-),\ldots,b(-))$, more precisely:

$$\begin{array}{lll}\Phi(x) & = x & \text{for all } x \in \mathsf{Var},\\ \Phi(f(t_1,\ldots,t_k)) & = f(\Phi(t_1),\ldots,\Phi(t_k)) & \text{for all } f \text{ with } f \neq a\\ \Phi(a(t_1,\ldots,t_n)) & = a(b(\Phi(t_1)),\ldots,b(\Phi(t_n))) & \\ \Phi(\mathcal{R}) & = \{\Phi(\ell) \to \Phi(r) \mid \ell \to r \in \mathcal{R}\}. & \end{array}$$

From this definition it is straightforwardly proved that if $t \to_{\mathcal{R}} u$, then $\Phi(t) \to_{\Phi(\mathcal{R})} \Phi(u)$. So the assumed infinite $\mathcal{R}$ reduction transforms by Φ to an infinite $\Phi(\mathcal{R})$ reduction.

On the other hand the symbol a is still a dummy symbol in $\Phi(\mathcal{R})$. By construction we have $E(\Phi(\mathcal{R})) = DE_a(\mathcal{R})$, which was assumed to be terminating. Hence by Theorem 7 we conclude termination of $\Phi(\mathcal{R})$, contradiction. □

The way we want to use dummy elimination in proving termination automatically is as follows: if termination of $\mathcal{R}$ has to be proved, and $\mathcal{R}$ has a dummy symbol a, then apply DE_a to $\mathcal{R}$, and proceed with the search for termination proofs on $DE_a(\mathcal{R})$. For this approach to be useful we should also like to have the converse of Theorem 6: $\mathcal{R}$ is terminating if and only if $DE_a(\mathcal{R})$ is terminating. This is seen as follows: if $\mathcal{R}$ is terminating but $DE_a(\mathcal{R})$ is not, then trying to prove termination of $DE_a(\mathcal{R})$ will always fail. For instance, let $\mathcal{R}$ consist of the two rules

$$f(g(x)) \to g(f(f(x))), \quad g(f(x)) \to g(a(g(g(x)))).$$

Then indeed $\mathcal{R}$ is terminating, but trying to prove this by proving termination of $E(\mathcal{R})$ consisting of the three rules

$$f(g(x)) \to g(f(f(x))), \quad g(f(x)) \to g(\diamond), \quad g(f(x)) \to g(g(x))$$

will fail since $E(\mathcal{R})$ is not terminating due to

$$f(f(g(x))) \to_{E(\mathcal{R})} f(g(f(f(x)))) \to_{E(\mathcal{R})} f(g(g(f(x)))) \to_{E(\mathcal{R})} g(\underbrace{f(f(g}(f(x))))).$$

Note that Theorem 4 does not apply here due to overlap between the rules. We conclude that the earlier version of dummy elimination E rather than DE_a fails due to the fact that the desired 'if and only if' property does not hold.

Next we show that in case of left-linearity, the desired 'if and only if' property holds for DE_a. First we need two lemmas.

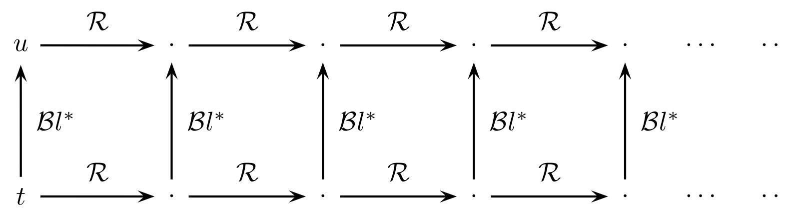
Let $\mathcal{B}l$ be the TRS defined to consist of all rules of the shape $f(x_1,\ldots,x_n) \to \diamond$, for all symbols f of arity $n \geq 0$.

Lemma 8. *Let $\mathcal{R}$ be a left-linear TRS in which the constant $\diamond$ and the unary symbol b do not occur in any lhs.*

1. *If $t \to^*_{\mathcal{B}l} u$ and $\infty(u,\mathcal{R})$, then $\infty(t,\mathcal{R})$.*
2. *If $\infty(C[b(t)],\mathcal{R})$ for any context C and any term t, then either $\infty(C[\diamond],\mathcal{R})$ or $\infty(t,\mathcal{R})$.*

Proof. Part *1.*

Let t,u,v be terms satisfying $t \to_{\mathcal{B}l} u \to_{\mathcal{R}} v$. Then u is obtained from t by replacing any subterm by $\diamond$. So the redex of $u \to_{\mathcal{R}} v$ is either above or parallel to this occurrence of $\diamond$. Since $\mathcal{R}$ is linear and $\diamond$ does not occur in the lhs of the correpending rule in $\mathcal{R}$, the TRS $\mathcal{R}$ could also be applied directly to t yielding $t \to_{\mathcal{R}} \cdot \to^*_{\mathcal{B}l} v$. Hence we conclude $\to_{\mathcal{B}l} \cdot \to_{\mathcal{R}} \subseteq \to_{\mathcal{R}} \cdot \to^*_{\mathcal{B}l}$. Using this property one easily proves $\to^*_{\mathcal{B}l} \cdot \to_{\mathcal{R}} \subseteq \to_{\mathcal{R}} \cdot \to^*_{\mathcal{B}l}$, applying induction on the number of $\to_{\mathcal{B}l}$-steps. Using this inclusion the infinite $\mathcal{R}$-reduction starting in u is transformed to an infinite $\mathcal{R}$-reduction starting in t, as is sketched in the following picture:



Part *2*. We prove the more general claim for multiple hole contexts:

> Let C be a multihole context for which $\mathsf{SN}(C[\diamond,\dots,\diamond],\mathcal{R})$ and $\infty(C[b(t_1),\dots,b(t_n)],\mathcal{R})$. Then $\infty(t_i,\mathcal{R})$ for some $i = 1,\dots,n$.

We prove this for all contexts C satisfying $\mathsf{SN}(C[\diamond,\dots,\diamond],\mathcal{R})$, by induction on $\mathcal{R}$. Consider the infinite $\mathcal{R}$-reduction starting in $C[b(t_1),\dots,b(t_n)]$. If all redex positions are below C then every step is in one of the n positions $C[b(t_1),\dots,b(t_n)]$, so at least one of the $b(t_i)$ is rewritten infinitely often. Since b does not occur in any lhs, the same holds for t_i and we are done. In the remaining case the infinite $\mathcal{R}$-reduction is of the shape

$$C[b(t_1),\dots,b(t_n)] \;\to^*_{\mathcal{R}}\; C[b(u_1),\dots,b(u_n)] \;\to_{\mathcal{R}}\; D[b(v_1),\dots,b(v_k)] \;\to^*_{\mathcal{R}}\; \cdots,$$

where $t_i \to^*_{\mathcal{R}} u_i$ for every i, $C[\diamond,\dots,\diamond] \to_{\mathcal{R}} D[\diamond,\dots,\diamond]$, and for every j there exists i such that $v_j = u_i$. Here we use left-linearity of $\mathcal{R}$ and non-occurrence of b's in lhs's. Since $C[\diamond,\dots,\diamond] \to_{\mathcal{R}} D[\diamond,\dots,\diamond]$ we may apply the induction hypothesis to $D[b(v_1),\dots,b(v_k)]$, yielding j satisfying $\infty(v_j,\mathcal{R})$. Since there exists i such that $v_j = u_i$ we obtain $t_i \to^*_{\mathcal{R}} u_i = v_j \to^{\infty}_{\mathcal{R}}$, so $\infty(t_i,\mathcal{R})$. ☐

Lemma 9. *Let t be any term and a any symbol. Then*

1. $t \to^*_{\mathcal{B}l} \mathsf{cap}_a(t)$, *and*
2. *for every $v \in \mathsf{dec}_a(t)$ the terms t, v can be written as $t = C[t']$ and $v = b(v')$, where $t' \to^*_{\mathcal{B}l} v'$.*

Proof. By induction on the structure of t, straightforward from the definitions of cap_a and dec_a. ☐

Theorem 10. *Let $\mathcal{R}$ be a left-linear terminating TRS having a dummy symbol a. Then $DE_a(\mathcal{R})$ is terminating.*

Proof. We prove that $\mathsf{SN}(t, DE_a(\mathcal{R}))$ for every term t, by induction on $\mathcal{R}$. So the induction hypothesis states that $\mathsf{SN}(w, DE_a(\mathcal{R}))$ for every term w satisfying $t \to^+_{\mathcal{R}} w$.

Assume that t admits an infinite $DE_a(\mathcal{R})$-reduction

$$t \;\to_{DE_a(\mathcal{R})}\; u \;\to_{DE_a(\mathcal{R})}\; \cdot \;\to_{DE_a(\mathcal{R})}\; \cdots .$$

For the step $t \;\to_{DE_a(\mathcal{R})}\; u$ we distinguish two cases, implied by the definition of $DE_a(\mathcal{R})$.

- $t = C[\ell\sigma]$ and $u = C[\mathsf{cap}_a(r)\sigma]$ for some context C, some substitution σ and some rule $\ell \to r$ in $\mathcal{R}$. Let $v = C[r\sigma]$. By part *1* of Lemma 9 we conclude that $r \to^*_{\mathcal{B}l} \mathsf{cap}_a(r)$, so $v = C[r\sigma] \to^*_{\mathcal{B}l} C[\mathsf{cap}_a(r)\sigma] = u$. Since $\infty(u, DE_a(\mathcal{R}))$ and $DE_a(\mathcal{R})$ is left-linear and has no $\diamond$ or b symbols in lhs's, we may apply part *1* of Lemma 8, yielding $\infty(v, DE_a(\mathcal{R}))$, contradicting the induction hypothesis.
- $t = C[\ell\sigma]$ and $u = C[v\sigma]$ for some context C, some substitution σ, $v \in \mathsf{dec}_a(r)$, and some rule $\ell \to r$ in $\mathcal{R}$. By part *2* of Lemma 9 we obtain $r = C'[r']$ and $v = b(v')$, where $r' \to^*_{\mathcal{B}l} v'$. By part *2* of Lemma 8 we may distinguish two cases based on the infinite $DE_a(\mathcal{R})$-reduction of $u = C[v\sigma = C[b(v'\sigma)]$:

- $\infty(C[\diamond], DE_a(\mathcal{R}))$. Since $C[r\sigma] \to_{\mathcal{B}l} C[\diamond]$ we conclude $\infty(C[r\sigma], DE_a(\mathcal{R}))$ from part *1* of Lemma 8.
- $\infty(v'\sigma, DE_a(\mathcal{R}))$. Since $r' \to^*_{\mathcal{B}l} v'$ we may apply part *1* of Lemma 8, yielding $\infty(r'\sigma, DE_a(\mathcal{R}))$. Since $C[r\sigma] = C[C'\sigma[r'\sigma]]$ we obtain $\infty(C[r\sigma], DE_a(\mathcal{R}))$.

In both cases we obtain $\infty(w, DE_a(\mathcal{R}))$ for $w = C[r\sigma]$ satisfying $t = C[\ell\sigma] \to_{\mathcal{R}} C[r\sigma] = w$, contradicting the induction hypothesis.

□

Left-linearity is essential in Theorem 10 as is shown by the following example. Let $\mathcal{R}$ consist of the single rule

$$f(x,x) \;\to\; f(a(c), a(d)).$$

Then $\mathcal{R}$ is terminating, but $DE_a(\mathcal{R})$ is not since it contains the non-terminating rule $f(x,x) \to f(\diamond, \diamond)$.

For proving termination of a TRS $\mathcal{R}$ containing a dummy symbol automatically we propose always to try proving termination of $DE_a(\mathcal{R})$ first. For non-left-linear TRSs this may fail even if $\mathcal{R}$ is terminating as was shown by the above example. However, even then it may be a good strategy first to search for some time for a termination proof of $DE_a(\mathcal{R})$, since often termination proofs for $DE_a(\mathcal{R})$ are substantially simpler than direct termination proofs for $\mathcal{R}$.

In case a TRS contains more than one dummy symbol it is a natural question how to proceed. It turns out that the order of applying the corresponding DE operations does not influence the result, e.g., if both a_1 and a_2 are dummy symbols in $\mathcal{R}$, then $DE_{a_1}(DE_{a_2}(\mathcal{R})) = DE_{a_2}(DE_{a_1}(\mathcal{R}))$. In constructing this combined dummy elimination we can apply it for all dummy symbols in one run, introducing a fresh constant $\diamond_a$ and a fresh unary symbol b_a for every dummy symbol a. So in case a TRS contains more than one dummy symbol we propose always to proceed by this combined dummy elimination.

The best tool at the moment for proving TRS termination is AProVE, see

`http://aprove.informatik.rwth-aachen.de/`

We give two examples now showing that our DE-strategy would make sense for AProVE. The first TRS consists of two rules

$$f(f(g(g(x)))) \to g(g(g(f(f(f(x)))))), \quad f(x) \to a(x,x).$$

AProVE fails to prove termination of this TRS. However, after applying DE_a the resulting TRS consisting of the rules

$$f(f(g(g(x)))) \to g(g(g(f(f(f(x)))))), \quad f(x) \to \diamond, \quad f(x) \to b(x)$$

is proved to be terminating by AProVE in a fraction of a second.

As the second example consider the TRS consisting of the rules

$$\begin{array}{ll} f(g(x)) \to f(h(h(a(h(h(g(f(x)))))))) & f(g(x)) \to g(g(f(h(x)))) \\ f(h(x)) \to h(g(f(x))) & g(h(x)) \to h(g(x)) \\ h(f(x)) \to g(g(h(h(a(f(x)))))) & f(x) \to g(g(h(x))). \end{array}$$

Again AProVE fails to prove termination of this TRS, but by applying DE_a the two rules containing a in their rhs's are replaced by

$$f(g(x)) \to f(h(h(\diamond))), \quad f(g(x)) \to b(h(h(g(f(x))))),$$

$$h(f(x)) \to g(g(h(h(\diamond)))), \quad h(f(x)) \to b(f(x)),$$

resulting in a TRS for wich termination is proved easily, e.g., by recursive path order choosing the precedence $f > g > h > b > \diamond$. In the tool TPA dummy elimination has been implemented, and indeed for this TRS TPA yields the following proof, in which the fresh symbols are called `#0` and `#1` rather than $\diamond$ and b:

```
Eliminate dummy symbol <a>, to obtain the following TRS:
(1a) f(g(x)) -> f(h(h(#0)))
(1b) f(g(x)) -> #1(h(h(g(f(x)))))
(2) f(h(x)) -> h(g(f(x)))
(3a) h(f(x)) -> g(g(h(h(#0))))
(3b) h(f(x)) -> #1(f(x))
(4) f(g(x)) -> g(g(f(h(x))))
(5) g(h(x)) -> h(g(x))
(6) f(x) -> g(g(h(x)))

Use following polynomial interpretation:
[f(x)] = x + 1
rest default

  Remove rules with left hand side strictly bigger than
right hand side: (3a), (6)

All the rules of this TRS can be oriented with RPO with the
following precedence:
f > g
f > h
f > #1
g > h
g > #0
h > #1
```

In particular this last example is of interest with respect to the following. In [9] it was proved that if $DE_a(\mathcal{R})$ is DP simply terminating then $\mathcal{R}$ is DP simply terminating too. Here roughly speaking DP simple termination means that termination can be proved by the dependency pair technique using argument filtering and a simplification order. In [9] it is claimed that this theorem implies that *using dummy elimination as a preprocessing step to the dependency pair technique does not have any advantage*. However, our latter example convincingly shows the converse: here no dependency pair transformation was required, but if the dependency pair transformation had been applied to the resulting system, a straightforward termination proof only using recursive path order would have been found easily too.

We conclude this section by the termination proof generated by TPA of `D33/31.trs` in the TRS category of TPDB:

```
(1) :(:(x,y),z) -> :(x,:(y,z))
(2) :(+(x,y),z) -> +(:(x,z),:(y,z))
(3) :(z,+(x,f(y))) -> :(g(z,y),+(x,a))

Eliminate dummy symbol <g>, to obtain the following TRS:
(1) :(:(x,y),z) -> :(x,:(y,z))
(2) :(+(x,y),z) -> +(:(x,z),:(y,z))
(3a) :(z,+(x,f(y))) -> :(#0,+(x,a))
(3b) :(z,+(x,f(y))) -> #1(z)
(3c) :(z,+(x,f(y))) -> #1(y)
```

```
Use following polynomial interpretation:
[:(x,y)] = xy
[+(x,y)] = x + y + 1
rest default

Remove rules with left hand side strictly bigger than
right hand side: (2), (3b)-(3c)

Use following polynomial interpretation:
[f(x)] = x + 1
rest default

Remove rules with left hand side strictly bigger than
right hand side: (3a)

All the rules of this TRS can be oriented with RPO with the
following precedence:
Status: :: Lex-LR,
Precedence: empty
```

4.2 Dummy elimination for string rewriting

For term rewriting we believe that the operation DE_a is the most natural and most powerful variant of dummy elimination, due to the combination of Theorem 6 and Theorem 10. However, for string rewriting there is a drawback: due to the introduction of the constant $\diamond_a$ for a string rewriting system (SRS) $\mathcal{R}$, being a TRS over a signature only containing unary symbols, the transformed system $DE_a(\mathcal{R})$ is not an SRS any more.

This can be solved by defining a variant DE'_a of DE_a, where the only difference is that $\diamond_a$ is a unary symbol rather than a constant. In this way more symmetry between $\diamond_a$ and b_a is introduced. To express this symmetry in the notation, we will write $a\$$ instead of $\diamond_a$, and $\$a$ instead of b_a. As usual, we will identify a term $a_1(a_2(\cdots(a_n(x))\cdots))$ with the string $a_1 a_2 \cdots a_n$, by simply ignoring all parentheses and the variable symbol. So the single variable x in term notation is written as the empty string λ in string notation. Now for a dummy symbol a in an SRS $\mathcal{R}$ we define

$$DE'_a(\mathcal{R}) = \{\ell \to u \mid u = \mathsf{cap}'_a(r) \vee u \in \mathsf{dec}'_a(r) \text{ for a rule } \ell \to r \in \mathcal{R}\},$$

where

$$\begin{array}{ll} \mathsf{cap}'_a(\lambda) & = \lambda \\ \mathsf{cap}'_a(fs) & = f\,\mathsf{cap}'_a(s) \text{ for all symbols } f \text{ with } f \neq a \text{ and all strings } s \\ \mathsf{cap}'_a(as) & = a\$ \\ \mathsf{dec}'_a(\lambda) & = \emptyset \\ \mathsf{dec}'_a(fs) & = \mathsf{dec}'_a(s) \quad \text{for all symbols } f \text{ with } f \neq a \text{ and all strings } s \\ \mathsf{dec}'_a(as) & = \mathsf{dec}'_a(s) \cup \{\$a(\mathsf{cap}'_a(s))\}. \end{array}$$

First we give an example. Let the SRS $\mathcal{R}$ consist of the single rule $bb \to bab$. Since it is self-embedding, it is not simply terminating. However, termination of $DE'_a(\mathcal{R})$ consisting of the two rules

$$b\,b \;\to\; b\,a\$,\;\; b\,b \;\to\; \$a\,b,$$

is trivial by counting the number of b-symbols.

The main theorem about DE'_a is the following.

Theorem 11. *Let $\mathcal{R}$ be an SRS having a dummy symbol a. Then $\mathcal{R}$ is terminating if and only if $DE'_a(\mathcal{R})$ is terminating.*

In order to prove Theorem 11 we need a general theorem relating termination of TRSs over constants and unary symbols, and SRSs.

The function ϕ is defined on terms over constants and unary symbols, yielding strings, is defined as follows:

$$\phi(x) = \lambda,\ \ \phi(c) = c\ \ \phi(f(t)) = f\phi(t)$$

for all variables x, all constants c and all unary symbols f. A TRS $\mathcal{R}$ over constants and unary symbols is mapped to an SRS $\phi(\mathcal{R})$ as follows:

$$\phi(\mathcal{R}) \;=\; \{\ \phi(\ell) \to \phi(r) \mid \ell \to r \in \mathcal{R}\ \}.$$

Theorem 12. *Let $\mathcal{R}$ be a TRS over constants and unary symbols. Then $\mathcal{R}$ is terminating if and only if $\phi(\mathcal{R})$ is terminating.*

Proof. (sketch)

The 'if'-part is obvious since an infinite $\mathcal{R}$-reduction is lifted to an infinite $\phi(\mathcal{R})$-reduction by adding postfixes.

For the converse fix x to be a variable, and define inductively:

$$\begin{array}{lll}
\mathsf{head}(\lambda) & = x & \\
\mathsf{head}(fs) & = f(\mathsf{head}(s)) & \text{for all unary symbols } f \text{ and all strings } s\\
\mathsf{head}(cs) & = c & \text{for all constants } c \text{ and all strings } s\\
\mathsf{tail}(\lambda) & = \emptyset & \\
\mathsf{tail}(fs) & = \mathsf{tail}(s) & \text{for all unary symbols } f \text{ and all strings } s\\
\mathsf{tail}(cs) & = \{\mathsf{head}(s)\} \cup \mathsf{tail}(s) & \text{for all constants } c \text{ and all strings } s
\end{array}$$

Define Φ from strings to multisets of terms by

$$\Phi(s) \;=\; \{\mathsf{head}(s)\} \cup \mathsf{tail}(s).$$

So if c, d are constants and f, g, h are unary symbols then

$$\Phi(cfgdghfch) \;=\; \{c, f(g(d)), g(h(f(c))), h(x)\}.$$

One checks that for $s \to_{\phi(\mathcal{R})} s'$ applying a rule $\phi(\ell) \to \phi(r)$ for $\ell \to r \in \mathcal{R}$ in which both or none of ℓ and r contain a variable, then

$$\Phi(s') \;=\; \Phi(s) \setminus \{t\}\ \cup \{u\}$$

for $t \in \Phi(s)$ and $t \to_{\mathcal{R}} u$. In the remaining applications of $\phi(\ell) \to \phi(r)$ for $\ell \to r \in \mathcal{R}$ we have that ℓ contains a variable and r does not. For this case we can prove

$$\Phi(s') \;=\; \Phi(s) \setminus \{t\}\ \cup \{u, u'\}$$

for $t \in \Phi(s)$ and $t \to_{\mathcal{R}} u$ and u' is a subterm of t. If $\mathcal{R}$ is terminating then it is well-known that the union of $\to_{\mathcal{R}}$ and the subterm relation is a well-founded order. Assume that $\phi(\mathcal{R})$ admits an infinite reduction. Then by the above properties this transforms by Φ to an infinite decreasing sequence with respect to the multiset lifting of this well-founded order, contradiction. ☐

This proof sketch of Theorem 12 was based on a discussion with Juergen Giesl, René Thiemann and Peter Schneider-Kamp. An alternative proof eliminating the multiset argument was worked out by René Thiemann.

The impact of Theorem 12 goes far beyond dummy elimination. In fact Theorem 12 states that proving termination of string rewriting is equivalent to termination of term rewriting as long as no symbols of arity higher than one occur.

Now we are ready to prove Theorem 11.

Proof. (of Theorem 11)

Since an SRS is left-linear, by Theorem 6 and Theorem 10 we conclude that $\mathcal{R}$ is terminating if and only if $DE_a(\mathcal{R})$ is terminating. By Theorem 12 this holds if and only if $\phi(DE_a(\mathcal{R}))$ is terminating. By construction $\phi(DE_a(\mathcal{R}))$ and $DE'_a(\mathcal{R})$ coincide, up to renaming of $\diamond_a$ to $a\$$ and b_a to $\$a$. □

The transformation DE'_a for string rewriting has been implemented in TORPA. As an example, applying TORPA, version 1.4, to the SRS consisting of the four rules

$$ad \to dbcbd, \quad a \to bbfbb, \quad bdb \to ad, \quad df \to bd$$

yields:

```
Apply dummy elimination, result:
a d  -> d b c$
a d  -> $c b d
a  -> b b f b b
b d b  -> a d
d f  -> b d
Choose polynomial interpretation
a: lambda x.x+72
d: lambda x.4x
b: lambda x.x+16
c$: lambda x.x+1
f: lambda x.x+5
$c: lambda x.x+1
remove: a d  -> d b c$
remove: a d  -> $c b d
remove: a  -> b b f b b
remove: b d b  -> a d
remove: d f  -> b d
Terminating since no rules remain.
```

Next we give the result of TORPA on the same example we considered before:

```
f g  -> f h h a h h g f
f h  -> h g f
h f  -> g g h h a f
f g  -> g g f h
g h  -> h g
f  -> g g h

Apply dummy elimination, result:
f g  -> f h h a$
f g  -> $a h h g f
f h  -> h g f
h f  -> g g h h a$
h f  -> $a f
f g  -> g g f h
g h  -> h g
f  -> g g h
```

```
Choose polynomial interpretation f: lambda x.x+1, rest identity
remove: h f  -> g g h h a$
remove: f  -> g g h
Choose polynomial interpretation
f: lambda x.4x
g: lambda x.x+5
h: lambda x.x+2
a$: lambda x.x+1
$a: lambda x.x+1
remove: f g  -> $a h h g f
remove: f h  -> h g f
remove: h f  -> $a f
remove: f g  -> g g f h
Choose polynomial interpretation g: lambda x.x+1, rest identity
remove: f g  -> f h h a$
Choose polynomial interpretation:
g: lambda x.10x,  rest lambda x.x+1
remove: g h  -> h g
Terminating since no rules remain.
```

5 Conclusions

We described two techniques to transform a given TRS to another one, in such a way that termination of the given TRS can be concluded from termination of the transformed one, and proving termination of the transformed TRS is often easier than proving termination of the given TRS directly.

Both techniques are easy to implement, and have the nice property that no choice has to be made, so never an explosion of the search space will be caused, and no heuristics have to be developed. On the other hand both techniques have a drawback: they are only applicable for a restricted class of TRSs. For rewriting right hand sides a rhs is required not being in normal form, and for dummy elimination a dummy symbol is required, i.e., a symbol occurring in a rhs but in no lhs. However, both our techniques may be applied not only as a preprocessing, but also in proofs consisting of several transformations of TRSs. If in a proof search the remaining proof obligation is finding a termination proof for some TRS, then both our techniques may be applied, even if they are not applicable to the original TRS.

One may wonder when to apply these techniques. Our proposal is: nearly always. For rewriting right hand sides we proved that the original TRS is terminating if and only if the transformed TRS is terminating, and we are not aware of TRSs for which termination of the transformed TRS is harder to prove than termination of the original TRS, while the converse often occurs. For left-linear TRSs the same can be said for our version of dummy elimination. So the only situation where the effect may be negative is for dummy elimination for non-left-linear TRSs. Indeed for the single rule $f(x,x) \;\to\; f(a(c),a(d))$ we saw that dummy elimination should not be applied, since then the transformed TRS is not terminating while the original one is.

Also combinations of both techniques described in this paper make sense. We conclude by the result of TORPA on the SRS consisting of the rules

$$bb \to cd,\; cc \to ddd,\; c \to g,\; dd \to cf,\; ddd \to gc,\; f \to ag,\; g \to dab,\; gg \to bc.$$

```
Transformation order: apply rule 6 on rhs of rule 4, result:
b b  -> c d
c c  -> d d d
c  -> g
d d  -> c a g
d d d  -> g c
f  -> a g
g  -> d a b
g g  -> b c

Apply dummy elimination, result:
b b  -> c d
c c  -> d d d
c  -> g
d d  -> c a$
d d  -> $a g
d d d  -> g c
f  -> a$
f  -> $a g
g  -> d a$
g  -> $a b
g g  -> b c
```

and for the resulting system a proof is found only using polynomials.

Acknowledgment

We want to thank Adam Koprowski for implementing the techniques described here in his tool TPA, providing the output of TPA on the given examples, and for careful proofreading this paper.

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. F. Bellegarde and P. Lescanne. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing*, 1(2):79–96, 1990.
4. A. Ben-Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9:137–159, 1987.
5. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
6. M. Ferreira. *Termination of Term Rewriting – Well-Foundedness, Totality, and Transformations*. PhD thesis, University of Utrecht, 1995.
7. M. Ferreira and H. Zantema. Dummy elimination: Making termination easier. In *Proceedings of the 10th International Conference on Fundamentals of Computation Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 243–252, Berlin, 1995. Springer-Verlag.
8. M. C. F. Ferreira. Dummy elimination in equational rewriting. In H. Ganzinger, editor, *Proceedings of the 7th Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 1996.
9. J. Giesl and A. Middeldorp. Eliminating dummy elimination. In *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, volume 1831 of *Lecture Notes in Computer Science*, pages 309–323. Springer, 2000.
10. Bernhard Gramlich. Simplifying termination proofs for rewrite systems by preprocessing. In Maurizio Gabrielli and Frank Pfenning, editors, *Proc. 2nd Int. Conf. on Principles and Practice of Declarative Programming (PPDP 2000), Montreal, Canada*, pages 139–150, Montreal, Canada, September 2000. ACM Press.

11. A. Middeldorp, H. Ohsaki, and H. Zantema. Transforming termination by self-labelling. In *Proceedings of the 13th International Conference on Automated Deduction*, volume 1104 of *Lecture Notes in Artificial Intelligence*, pages 373–386, Berlin, 1996. Springer-Verlag.
12. J. Steinbach. Automatic termination proofs with transformation orderings. In J. Hsiang, editor, *Proceedings of the 6th Conference on Rewriting Techniques and Applications*, volume 914 of *Lecture Notes in Computer Science*, pages 11–25. Springer, 1995.
13. H. Zantema. Termination of term rewriting: Interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.
14. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.
15. H. Zantema. Termination. In *Term Rewriting Systems, by Terese*, pages 181–259. Cambridge University Press, 2003.
16. H. Zantema. Termination of string rewriting proved automatically. Technical Report CS-report 03-14, Eindhoven University of Technology, 2003. Available via `http://www.win.tue.nl/inf/onderzoek/en_index.html` .
17. H. Zantema. TORPA: termination of rewriting proved automatically. 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 95–104. Springer, 2004.
18. H. Zantema. Termination of string rewriting proved automatically. *Journal of Automated Reasoning*, 2005. accepted.