# Erasure for Termination Proofs

Hongwei Xi[1] * and Joachim Steinbach[2]

[1] Department of Computer Science and Engineering
Oregon Graduate Institute
P.O. Box 91000, Portland, OR 97291, USA
e-mail: hongwei@cse.ogi.edu

[2] Institut für Informatik
Technische Universität München
80290 München, Germany
e-mail: steinbac@in.tum.de

**Abstract.** We introduce a technique to facilitate termination proofs for term rewriting systems. We especially focus on innermost termination. The main features of this technique lie in its simplicity and effectiveness in practice. This work can be regarded as an application of the general notion *termination through transformation* to both termination and innermost termination proofs.

## 1 Introduction

It is a highly significant question to determine whether a term rewriting system (TRS) is terminating. In theorem proving, TRSs are widely used for a variety of purposes. For instance, it is often desirable to transform a set of equality rules into a TRS in order to reduce the search space. Also TRSs can be used for proving the termination of both functional and logic programs.

Though termination is an undecidable property of TRSs in general, there have been many techniques developed for facilitating termination proofs. Some surveys are given in [Der87,Ste95b]. As mentioned in [MOZ96], techniques for termination proofs can be generally classified into two categories.

– Basic techniques such as various path orderings [Pla78,KL80,Der82], Knuth-Bendix ordering [KB70], and polynomial interpretations [Lan79,BL87] that apply directly to a TRS.
– Transformational approaches which in general transform a TRS into another TRS such that the termination of the latter implies that of the former and the latter can be proven terminating more easily. For instance, transformation orderings [BL90,Ste95a], semantic labelling [Zan95] and freezing [Xi98] belong to this category. Also the dependency pair approach [AG97,AG98] can be loosely classified into this category since it transforms a TRS into a set of dependency pairs.

There are also various results on modular termination, which basically give the sufficient conditions on two terminating TRSs that imply the termination of their union. The importance of modularity results is evident. It is often true that new TRSs are formed on top of existing TRSs. With modularity results, it is possible to reduce the termination of new TRSs to that of the existing ones. In this paper, we adopt a transformational approach for establishing some results on modular termination and innermost termination. Given a TRS $\mathcal{R}$, we intend to split $\mathcal{R}$ into the union of $\mathcal{R}_1$ and $\mathcal{R}_2$, and then prove that the (innermost) termination of $\mathcal{R}_1$ implies that of $\mathcal{R}$ under some conditions.

We say that a TRS is innermost terminating if there is no infinite innermost rewriting sequence in this TRS. Roughly speaking, innermost rewriting means that we can rewrite a term only if all of its proper subterms are in normal form. To some extent, innermost rewriting can model the notion of call-by-value evaluation in functional programming, though there are usually some special rules for handling

* Partially supported by the United States Air Force Materiel Command (F19 628-96-C-0161) and the Department of Defense.

conditionals. Also it is proven in [AZ95] that the innermost termination of the TRS transformed from a logic program implies the termination of the logic program. Therefore, the study on innermost termination is of significant relevance to the study of termination of functional and logic programs. Moreover, there are also various results which relates innermost termination to termination [Gra95]. This allows us to reduce termination to innermost termination for some TRSs, where the latter is often easier to prove.

We now present an example to illustrate the erasure technique before going into further details. It is frequent to encounter hierarchical combination of TRSs when we transform functional programs into TRSs. The simple reason is that defined functions are used to define new functions. For instance, the following function `purge` defined in ML [MTHM97] removes all duplicates from a given (integer) list while the function `remove` deletes all the elements equal to some given value.

```
fun remove(x, nil) = nil
  | remove(x, cons(y, ys)) =
    if x = y then remove(x, ys) else cons(y, remove(x, ys))

fun purge(nil) = nil
  | purge(cons(x, xs)) = cons(x, purge(remove(x, xs)))
```

When proving termination of such a functional program, the following aspect must be taken into consideration:

> Usually, the programmer applies a semantic argument such as a measure function in order to show that the defined function is terminating. For example, the function `purge` is terminating because the length of the list $remove(x, ys)$ is not greater than that of $ys$. Note that it is in general an exceedingly difficult task to synthesize such a measure function from the structure of a program.

The program can be transformed into the following TRS $\mathcal{R}_{\mathrm{pg}}$ [1].

$$\begin{array}{lrcl}
(1) & remove(x, nil) & \to & nil \\
(2) & remove(x, cons(y, ys)) & \to & if(x = y, remove(x, ys), cons(y, remove(x, ys))) \\
(3) & purge(nil) & \to & nil \\
(4) & purge(cons(x, xs)) & \to & cons(x, purge(remove(x, xs)))
\end{array}$$

It seems difficult to prove the termination of this TRS with a syntactic approach. We can transform this TRS into the following TRS $\mathcal{R}^1_{\mathrm{pg}}$ with the erasure technique (ET) [2].

$$\begin{array}{lrcl}
(1') & nil & \to & nil \\
(2.1') & cons(y, ys) & \to & ys \\
(2.2') & cons(y, ys) & \to & cons(y, ys) \\
(3') & purge(nil) & \to & nil \\
(4') & purge(cons(x, xs)) & \to & cons(x, purge(xs))
\end{array}$$

In this case, we project a term beginning with $remove$ to the second argument of $remove$ and a term beginning with $if$ to either the second or the third argument of $if$. Under the recursive path ordering RPO with the precedence $purge \succ cons$, the rules (2.1′), (3′) and (4′) can be strictly ordered and the rules (1′) and (2.2′) can be ordered. We now informally argue that $\mathcal{R}_{\mathrm{pg}}$ is terminating. Suppose that there is an infinite innermost $\mathcal{R}_{\mathrm{pg}}$-rewriting sequence. We will show that this sequence induces an infinite $\mathcal{R}^1_{\mathrm{pg}}$-rewriting sequence. We then observe that this induced sequence cannot have infinitely many applications of those strictly ordered rules. Therefore, there is an infinite $\mathcal{R}^1_{\mathrm{pg}}$-rewriting sequence in which only applied rules are either (1′) or (2.2′). We will then prove this implies that there is an infinite innermost $\mathcal{R}_{\mathrm{pg}}$-rewriting sequence in which the only applied rules are either (1) or (2). This is a contradiction since the

[1] We omit the rules involving $=$ and $if$ at this moment.

[2] The following is slightly different from the actual application of ET for the purpose of presentation.

TRS consisting of rules (1) and (2) is easily proven to be terminating. Therefore, we conclude that $\mathcal{R}$ is innermost terminating. This argument will be substantiated in Section 3.

As already mentioned, most of the programmers use semantic arguments to prove termination. This is a powerful and flexible approach but it is also too semantic to be largely automated. On the other hand, the limited erasure technique is syntactic, and thus it is reasonable to expect that this approach can be combined with other approaches such as the freezing technique to facilitate automatic innermost termination proofs. However, we observe in practice [SX98] that it is even questionable to scale an approach as simple as RPOS, not mentioning other more involved techniques. Therefore, we expect that a more promising direction is to apply the erasure technique interactively. We shall make this point more clear with concrete examples.

This paper is organized as follows. In Section 2, we briefly explain the notations and introduce some basic concepts. We present the erasure technique (ET) for innermost termination proofs in Section 3 and establish the correctness of ET. This section constitutes the main contribution of the paper. We then mention some closely related work and conclude. We also present some examples in Appendix A, which can be of some assistance for the reader to understand the presented work if necessary.

## 2 Preliminaries

In general, we stick to the notations in [DJ91] though some minor modifications may occur. We briefly summarize the notations and develop some concepts needed later.

### 2.1 Basics

We fix a countably infinite set $\mathcal{X}$ of variables $x, y, \ldots$ and use $\mathcal{F}$ for a (finite) set of function symbols $f, g, \ldots$. Note that every function symbol $f$ is of a fixed arity $\mathcal{A}r(f)$ and $f$ is a constant if $\mathcal{A}r(f) = 0$. We assume that there is at least one constant in $\mathcal{F}$. Let $\mathcal{T}(\mathcal{F}, \mathcal{X})$ denote the set of terms over $\mathcal{F}$ and $\mathcal{X}$, and $\mathcal{T}(\mathcal{F})$ for the set of ground terms over $\mathcal{F}$. Given a term $t$, $\mathcal{V}ar(t)$ is the set of variables that occur in $t$. We use $l \to r$ for a rewrite rule, where we require $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. We use $\sigma$ for substitutions and $\mathrm{dom}(\sigma)$ for its (finite) domain. Also $t\sigma$ stands for the result of applying $\sigma$ to $t$.

**Definition 1.** Contexts $C$ *are defined as follows.*

1. *$[]$ is a context, and*
2. *$f(t_1, \ldots, t_{i-1}, C, t_{i+1}, \ldots, t_n)$ is a context if $\mathcal{A}r(f) = n$ and $C$ is a context.*

$C[t]$ is the term obtained from replacing the hole $[]$ in $C$ with term $t$.

A TRS $\mathcal{R}$ over $\mathcal{F}$ is a set of rewrite rules over $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A function symbol $f$ is an $\mathcal{R}$-defined function if $f$ is the root symbol of $l$ for some rewrite rule $l \to r$ in $\mathcal{R}$, and $f$ is a $\mathcal{R}$-constructor if it is not an $\mathcal{R}$-defined function. We often use $c$ for constructors.

Given a TRS $\mathcal{R}$, we write $t_1 \to_{\mathcal{R}} t_2$ if $t_1 = C[l\sigma]$ and $t_2 = C[r\sigma]$ and $l \to r$ is a rewrite rule in $\mathcal{R}$, and we may also write $t_1 \to_{\mathcal{R}} t_2 / \langle C, l \to r, \sigma \rangle$ to make this explicit. A term $t$ is in $\mathcal{R}$-normal form if there exists no $t'$ such that $t \to_{\mathcal{R}} t'$ holds. If $l \to r \in \mathcal{R}$ and all proper subterms of $l\sigma$ are in $\mathcal{R}$-normal form, we say $t_1 = C[l\sigma]$ rewrites to $t_2 = C[r\sigma]$ through *innermost* rewriting, and we use $\xrightarrow{i}_{\mathcal{R}}$ for such a rewriting relation. Also we use $t \to^{0/1} t'$ to mean that either $t = t'$ or $t \to t'$.

We use $\to^*$ for the transitive and reflexive closure of a relation $\to$. $\mathcal{R}$ is (innermost) terminating if there exists no infinite (innermost) $\mathcal{R}$-rewriting sequence. Given a substitution $\sigma$, $\sigma$ is $\mathcal{R}$-normal if $\sigma(x)$ is in $\mathcal{R}$-normal form for every $x \in \mathrm{dom}(\sigma)$. The following definition is less standard.

**Definition 2.** *Given a term $t$, $t$ is skeleton $\mathcal{R}$-normal if we always obtain terms in $\mathcal{R}$-normal form by replacing occurrences of variables in $t$ with terms in $\mathcal{R}$-normal form. Note that we do not have to replace occurrences of the same variable with the same terms. Similarly, $t$ is skeleton $\mathcal{R}$-terminating if we always obtain $\mathcal{R}$-terminating terms by replacing occurrences of variables in $t$ with $\mathcal{R}$-terminating terms.*

We have the following limited method to construct skeleton $\mathcal{R}$-normal terms.

**Proposition 1.** *Let $\mathcal{R}$ be a TRS.*

1. *Every variable is skeleton $\mathcal{R}$-normal.*
2. *$c(t_1, \ldots, t_n)$ is skeleton $\mathcal{R}$-normal if $c$ is an $\mathcal{R}$-constructor and $t_i$ are skeleton $\mathcal{R}$-normal for $i = 1, \ldots, n$.*

*Proof* This is straightforward by the definition. ∎

In other words, $\mathcal{R}$-constructor terms, that is, terms constructed from $\mathcal{R}$-constructors and variables, are skeleton $\mathcal{R}$-normal. Similarly, $\mathcal{R}$-constructor terms are also $\mathcal{R}$-terminating.

We use the notation $\succeq$ for a quasi ordering and $\succ$ for the strict part of $\succeq$. A reduction ordering is an ordering $\succeq$ such that its strict part $\succ$ is well-founded and both $\succeq$ and $\succ$ are compatible with the term structure and stable under substitutions. One of the most well-known and widely used reduction orderings is the *recursive path ordering* RPOS *with status* [Der82,KL80]. Please see [Ste95b] for further details.

*Remark 1.* We say that a rewrite rule $l \to r$ is *strictly* ordered under $\succeq$ if $l \succ r$, and $l \to r$ is ordered if $l \succeq r$.

### 2.2 Hierarchical Combination

**Definition 3.** *Given two TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$, we say $\mathcal{R}_1$ and $\mathcal{R}_2$ form a* hierarchical combination $\mathcal{R}_1 \cup \mathcal{R}_2$ *if no defined function symbols in $\mathcal{R}_2$ have appearances in $\mathcal{R}_1$. Given a term $t$, a subterm of $t$ is called an $\mathcal{R}_2$-subterm if the root symbol of the subterm is a $\mathcal{R}_2$-defined function symbol.*

Notice that hierarchical combination occurs naturally when we transform functional programs into TRSs: defined functions are used to define new functions.

We omit the proof of the following lemma since it is really a bit of folklore in term rewriting.

**Lemma 1.** *Suppose that two TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ form a hierarchical combination $\mathcal{R}$. We have the following.*

1. *If all $\mathcal{R}_2$-subterms of $t$ are in $\mathcal{R}$-normal form and $t \to_{\mathcal{R}} t'$, then all $\mathcal{R}_2$-subterms of $t'$ are in $\mathcal{R}$-normal form.*
2. *If $\mathcal{R}_1$ is terminating and all $\mathcal{R}_2$-subterms of $t$ are in $\mathcal{R}$-normal form, then $t$ is (innermost) $\mathcal{R}$-terminating, that is, there is no infinite (innermost) $\mathcal{R}$-rewriting sequence from $t$.*

In the following presentation, we may omit the prefix "$\mathcal{R}$-" if it is irrelevant or it is clear from the context which $\mathcal{R}$ we refer to.

### 2.3 Erasure

Generally speaking, $t_1$ is an erasure of $t_2$ if $t_1$ can be obtained from erasing some function symbols and subterms in $t_2$. In other words, $t_1$ embeds into $t_2$. However, it will soon be clear that some embedding may not be erasure.

For every function symbol $f$ in $\mathcal{F}$ with arity $n$, we associate with it the following rewrite rules for $i = 1, \ldots, n$.

$$\begin{array}{l} (f\text{-o-}i)\ f(x_1, \ldots, x_n) \to f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \\ (f\text{-p-}i)\ f(x_1, \ldots, x_n) \to x_i \end{array}$$

An $(f$-o-$i)$ rule is called an *omitting rule* and an $(f$-p-$i)$ a *projection rule*. Both of these rules are called erasure rules. Notice that an $(f$-o-$i)$ rule changes the arity of $f$. Also we say that $(f$-p-$i)$ is not argument-dropping if $\mathcal{A}r(f) = 1$. All other erasure rules are argument-dropping.

Given a set $\mathcal{S}$ of erasure rules in which there is *at most* one rule associated with $f$ for every $f \in \mathcal{F}$, we call $\mathcal{S}$ an erasure TRS. The $\mathcal{S}$-erasure of $t$ is the $\mathcal{S}$-normal form of $t$, which is alternatively defined as follows.

**Definition 4.** *Given an erasure TRS $\mathcal{S}$, we use $|t|_\mathcal{S}$ for the $\mathcal{S}$-erasure of $t$ and $\epsilon(t)_\mathcal{S}$ for the set of terms erased from $t$. In general, we omit the subscript $\mathcal{S}$ if there is no risk of confusion.*

$$|t| = \begin{cases} t & \textit{if } t \textit{ is a variable;} \\ f(|t_1|, \ldots, |t_{i-1}|, |t_{i+1}|, \ldots, |t_n|) & \textit{if } t = f(t_1, \ldots, t_n) \textit{ and } (f\text{-}o\text{-}i) \in \mathcal{S}; \\ |t_i| & \textit{if } t = f(t_1, \ldots, t_n) \textit{ and } (f\text{-}p\text{-}i) \in \mathcal{S}; \\ f(|t_1|, \ldots, |t_n|) & \textit{if } t = f(t_1, \ldots, t_n) \textit{ and otherwise.} \end{cases}$$

$$\epsilon(t) = \begin{cases} \emptyset & \textit{if } t \textit{ is a variable;} \\ \{t_i\} \cup \bigcup_{j \in \{1,\ldots,n\} \setminus \{i\}} \epsilon(t_j) & \textit{if } t = f(t_1, \ldots, t_n) \textit{ and } (f\text{-}o\text{-}i) \in \mathcal{S}; \\ \{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n\} \cup \epsilon(t_i) & \textit{if } t = f(t_1, \ldots, t_n) \textit{ and } (f\text{-}p\text{-}i) \in \mathcal{S}; \\ \bigcup_{j \in \{1,\ldots,n\}} \epsilon(t_j) & \textit{if } t = f(t_1, \ldots, t_n) \textit{ and otherwise.} \end{cases}$$

The erasure of rule $l \to r$ is $|l| \to |r|$, and the erasure of $\mathcal{R}$ is defined analogously. Note that the erasure of a rewrite rule may not always be a legal rewrite rule. For instance, the $\mathcal{S}$-erasure of $\mathit{if}(\mathit{false}, x, y) \to y$ is $x \to y$ for $\mathcal{S} = \{(\mathit{if}\text{-p-}2)\}$, which is illegal. Similarly, the erasure of a TRS may not be a legal TRS.

The erasure $|C|$ of a context $C$ can be defined in a straightforward manner. However, $|C|$ may not be a context since the hole $[]$ in $C$ may be erased away. In this case, we write $|C|[t]$ simply for $|C|$. Given a substitution $\sigma$, its erasure $|\sigma|$ is a substitution with the same domain and $|\sigma|(x) = |\sigma(x)|$ for every $x \in \mathrm{dom}(\sigma)$.

**Proposition 2.** *Given a context $C$, a term $t$ and a substitution $\sigma$, we have $|C[t]| = |C|[|t|]$ and $|t\sigma| = |t||\sigma|$.*

*Proof* This is straightforward from a structural induction on $C$ and $t$, respectively. ∎

**Lemma 2.** *Suppose that the erasure $|\mathcal{R}|$ of a TRS $\mathcal{R}$ is also a TRS. If $t_1 \to_\mathcal{R} t_2$, then $|t_1| \to_{|\mathcal{R}|}^{0/1} |t_2|$.*

*Proof* Assume $t_1 = C[l\sigma]$ and $t_2 = C[r\sigma]$ for some $\sigma$, where $l \to r \in \mathcal{R}$. If $|C|$ is *not* a context, then $|t_1| = |C| = |t_2|$. Otherwise, $|t_1| = |C|[|l||\sigma|]$ and $|t_2| = |C|[|r||\sigma|]$ by Proposition 2. Since $|l| \to |r| \in |\mathcal{R}|$, we have $|t_1| \to_{|\mathcal{R}|} |t_2|$. Clearly, if $|C|$ is a context, then $|t_1| \to_{|\mathcal{R}|} |t_2|$. ∎

Note that for every $f \in \mathcal{F}$ with arity $n$, we can introduce the following omitting rule, where $1 \leq i_1 < \ldots < i_k \leq n$.

$$(f\text{-o-}(i_1, \ldots, i_k)) \;\; f(x_1, \ldots, x_n) \to f(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_k-1}, x_{i_k+1}, \ldots, x_n)$$

In other words, this rule drops the subterms of $f(t_1, \ldots, t_n)$ at the positions $i_1, \ldots, i_k$. This rule is argument-dropping. Note that this is a *single* rule, which should not be regarded as a combination of several omitting rules. Also it should be clear that all the previous results involving erasure still hold in the presence of such omitting rules.

## 3 Erasure for Termination Proofs

The erasure technique (ET) is mainly to facilitate modular innermost termination proofs for TRSs. Notice that innermost termination implies termination for overlay TRSs [Gra95], and therefore this can also facilitate (classical) termination proofs. We also show that ET can be directly applied to (classical) termination proofs. The essential idea behind ET is *simulation* as presented in [Xi98]. In general, ET can be regarded as an application of the notion *termination through transformation* to both termination and innermost termination proofs.

### 3.1 Elementary Versions of ET

In this section, we establish some elementary versions of the erasure technique.

**Definition 5.** *Given a TRS $\mathcal{R}$, we say that $l \to r \in \mathcal{R}$ has a conservative erasure if $|l| \to |r|$ is a legal rewrite rule and $t$ is skeleton $\mathcal{R}$-normal for every $t \in \epsilon(r)$, that is, all subterms erased from $r$ are skeleton $\mathcal{R}$-normal. If all the rules in $\mathcal{R}$ have conservative erasures, then we say $\mathcal{R}$ has a conservative erasure $|\mathcal{R}|$.*

The next theorem is the most elementary one among those for ET which we will formulate and prove. Nonetheless, this theorem has largely captured the essential idea behind ET.

**Theorem 1.** *Assume that $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ has an erasure $\mathcal{R}' = \mathcal{R}'_1 \cup \mathcal{R}'_2$, where $\mathcal{R}'_i$ are the conservative erasures of $\mathcal{R}_i$ for $i = 1, 2$. Also assume that under some reduction ordering, every rule in $\mathcal{R}'_1$ can be ordered and every rule in $\mathcal{R}'_2$ can be strictly ordered, then the innermost termination of $\mathcal{R}_1$ implies the innermost termination of $\mathcal{R}$. In the case where all erasure rules are not argument-dropping, the termination of $\mathcal{R}_1$ implies that of $\mathcal{R}$.*

*Proof* Suppose that there exists an infinite innermost $\mathcal{R}$-rewriting sequence as follows:

$$t_1 \xrightarrow{i}_{\mathcal{R}} t_2 \xrightarrow{i}_{\mathcal{R}} \cdots \xrightarrow{i}_{\mathcal{R}} t_n \xrightarrow{i}_{\mathcal{R}} \cdots$$

where $t_i \to t_{i+1}/\langle C_i, l_i \to r_i, \sigma_i\rangle$ for some context $C_i$, rule $l_i \to r_i \in \mathcal{R}$ and substitution $\sigma_i$. We show that there is an infinite innermost $\mathcal{R}_1$-rewriting sequence.

Obviously, we can require that all proper subterms of $t_1$ be in $\mathcal{R}$-normal form since we are handling innermost rewriting. This implies that all terms in $\epsilon(t_1)$ are in $\mathcal{R}$-normal form. We now show inductively that this is true for all $t_i$ $(i = 1, 2, \ldots)$ by analyzing the difference between $\epsilon(t_i)$ and $\epsilon(t_{i+1})$. Let $t \in \epsilon(t_{i+1})$ and we have the following.

- $t$ is in $\epsilon(t_i)$. Then $t$ is in $\mathcal{R}$-normal form by induction hypothesis.
- $t$ is not in $\epsilon(t_i)$. Note $t_i = C_i[l_i\sigma_i]$ and $t_{i+1} = C_i[r_i\sigma_i]$. If $t$ contains $r_i\sigma_i$, then there must be some $s$ in $\epsilon(t_i)$ such that $s \to t$. This is impossible since all terms in $\epsilon(t_i)$ are in $\mathcal{R}$-normal form. Otherwise, $t$ is dropped from $r_i\sigma_i$. This means that $t$ either equals $s\sigma_i$ for some $s \in \epsilon(r)$ or $t$ is a subterm of $\sigma_i(x)$ for some $x \in \mathrm{dom}(\sigma_i)$. In the latter case, $t$ is obviously in $\mathcal{R}$-normal form since this is innermost rewriting. In the former case, $t$ is in $\mathcal{R}$-normal form since $s$ is skeleton $\mathcal{R}$-normal (note that $\mathcal{R}'$ is a conservative erasure of $\mathcal{R}$) and $\sigma$ is an $\mathcal{R}$-normal substitution.

Therefore, for $i = 1, 2, \ldots$, all terms in $\epsilon(t_i)$ are in $\mathcal{R}$-normal form. By Lemma 2, we have the following.

$$|t_1| \to^{0/1}_{|\mathcal{R}|} |t_2| \to^{0/1}_{|\mathcal{R}|} \cdots \to_{\mathcal{R}} |t_n| \to^{0/1}_{|\mathcal{R}|} \cdots$$

We now show that every $\to^{0/1}_{|\mathcal{R}|}$ step in this sequence is actually a $\to_{|\mathcal{R}|}$ step. It suffices to show that $|C_i|$ is always a context for $i = 1, 2, \ldots$. Suppose that $|C_i|$ is not a context. Then $l_i\sigma_i$ is a subterm of some term in $\epsilon(t_i)$. This is impossible since all terms in $\epsilon(t_i)$ are in $\mathcal{R}$-normal form. This implies that we actually have the following.

$$|t_1| \to_{|\mathcal{R}|} |t_2| \to_{|\mathcal{R}|} \cdots \to_{\mathcal{R}} |t_n| \to_{|\mathcal{R}|} \cdots$$

Since all rules in $\mathcal{R}'_1$ are ordered and all rules in $\mathcal{R}'_2$ are strictly ordered, there must be an $n$ such that all the rules applied after $|t_n|$ are in $\mathcal{R}'_1$. This implies that all the rules applied in the infinite innermost $\mathcal{R}$-rewriting sequence after $t_n$ are in $\mathcal{R}_1$, that is, we have an infinite innermost $\mathcal{R}_1$-rewriting sequence. Therefore, the innermost termination of $\mathcal{R}_1$ implies that of $\mathcal{R}$.

We now prove the second part of the theorem. Suppose that all the erasure rules are not argument-dropping. Then $|C|$ is a context for every context $C$. Therefore, $t_1 \to_{\mathcal{R}} t_2$ implies $|t_1| \to_{|\mathcal{R}|} |t_2|$ for every pair of terms $t_1$ and $t_2$. With the same argument as before, we can show that an infinite $\mathcal{R}$-rewriting sequence induces an infinite $\mathcal{R}_1$-rewriting sequence. Therefore, the termination of $\mathcal{R}_1$ implies that of $\mathcal{R}$. ■

Notice that we assume *no* relation between $\mathcal{R}_1$ and $\mathcal{R}_2$ in Theorem 1. This is an attractive feature in practice. Suppose that we intend to prove the termination of $\mathcal{R}$. We proceed to find a conservative erasure $\mathcal{R}'$ of $\mathcal{R}$ such that all rules in $\mathcal{R}'$ can be ordered under some reduction ordering. If there are rules in $\mathcal{R}'$ which can be strictly ordered, we remove them and use $\mathcal{R}'_1$ for the set of remaining rules. We can then find $\mathcal{R}_1 \subseteq \mathcal{R}$ such that $\mathcal{R}'_1$ is the conservative erasure of $\mathcal{R}_1$. In this way, we have reduced the innermost termination of $\mathcal{R}$ to that of $\mathcal{R}_1$. If $\mathcal{R}_1$ is empty, then we have proven that $\mathcal{R}$ is innermost terminating. Clearly, there is no need for splitting $\mathcal{R}$ before applying Theorem 1.

The following TRS $\mathcal{R}_{\text{wt}}$ is taken from [AG98]. Note that $m, n$ are variables, :: is the infix operator for *cons*, [] for *nil* and $[n]$ for $cons(n, nil)$. The function *weight* computes a weighted sum of natural numbers: $weight(n_0 :: n_1 :: \cdots :: n_k :: nil) = n_0 + \Sigma_{i=1}^{k} i * n_i$.

$$\begin{array}{lrl}
(1) & sum(s(m) :: x, n :: y) & \rightarrow sum(m :: x, s(n) :: y) \\
(2) & sum(0 :: x, y) & \rightarrow sum(x, y) \\
(3) & sum([], y) & \rightarrow y \\
(4) & weight([n]) & \rightarrow n \\
(5) & weight(m :: n :: x) & \rightarrow weight(sum(m :: n :: x, 0 :: x))
\end{array}$$

The last rule is self-embedding, and therefore the TRS cannot be proven terminating with a simplification ordering. Intuitively, $\mathcal{R}_{\text{wt}}$ is terminating because the length of $sum(m :: n :: x, 0 :: x)$ is less than that of $m :: n :: x$. We can use the erasure TRS $\mathcal{S} = \{$(sum-p-2),(s-p-1)$\}$ to capture this. The following TRS $\mathcal{R}'_{\text{wt}}$ is the $\mathcal{S}$-erasure of $\mathcal{R}_{\text{wt}}$.

$$\begin{array}{lrl}
(1') & n :: y & \rightarrow n :: y \\
(2') & y & \rightarrow y \\
(3') & y & \rightarrow y \\
(4') & weight([n]) & \rightarrow n \\
(5') & weight(m :: n :: x) & \rightarrow weight(0 :: x)
\end{array}$$

Notice that this is a conservative erasure. For instance, let $r$ be the right-hand side of rule (5), then $\epsilon(r)_{\mathcal{S}}$ is $\{m :: n :: x\}$, in which the term is skeleton $\mathcal{R}_{\text{wt}}$-normal. Clearly, $\mathcal{R}'_{\text{wt}}$ can be ordered under a $\mathsf{RPO}$. Since the rules (4′) and (5′) are strictly ordered, we delete them. Therefore, the innermost termination of $\mathcal{R}_{\text{sum}}$, which consists of the rules (1), (2) and (3), implies that of $\mathcal{R}_{\text{wt}}$ by Theorem 1. The termination of $\mathcal{R}_{\text{sum}}$ is readily proven with a $\mathsf{RPOS}$, and thus $\mathcal{R}_{\text{wt}}$ is innermost terminating. In this case $\mathcal{R}_{\text{wt}}$ is terminating since it is an overlay (actually non-overlapping) TRS.

On the other hand, if we can split a TRS into some hierarchical combination, then we can take advantage of Theorem 2 below, which is a generalized version of Theorem 1. We first present a definition very close to Definition 5.

**Definition 6.** *Let $\mathcal{R}$ be the hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$. We say that $l \rightarrow r \in \mathcal{R}$ has an $\mathcal{R}_2$-conservative erasure if $|l| \rightarrow |r|$ is a legal rewrite rule and $t$ is skeleton $\mathcal{R}_2$-normal for every $t \in \epsilon(r)$. If all rules in $\mathcal{R}$ have $\mathcal{R}_2$-conservative erasures, then we say $\mathcal{R}$ has an $\mathcal{R}_2$-conservative erasure $|\mathcal{R}|$.*

**Theorem 2.** *Let $\mathcal{S}$ be an erasure TRS and $\mathcal{R}$ be the hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2 = \mathcal{R}_{21} \cup \mathcal{R}_{22}$ such that $|\mathcal{R}_1|$ and $|\mathcal{R}_2|$ are $\mathcal{R}_2$-conservative. Assume that under some reduction ordering, the erasure of every rule in $\mathcal{R}_1$ and $\mathcal{R}_{21}$ can be ordered and the erasure of every rule in $\mathcal{R}_{22}$ can be strictly ordered, then the innermost termination of $\mathcal{R}_1 \cup \mathcal{R}_{21}$ implies the innermost termination of $\mathcal{R}$. In the case where all erasure rules in $\mathcal{S}$ are not argument-dropping, the termination of $\mathcal{R}_1 \cup \mathcal{R}_{21}$ implies the termination of $\mathcal{R}$.*

*Proof* This is very similar to the proof of Theorem 1. Suppose that there exists an infinite innermost $\mathcal{R}$-rewriting sequence as follows:

$$t_1 \xrightarrow{i}_{\mathcal{R}} t_2 \xrightarrow{i}_{\mathcal{R}} \cdots \xrightarrow{i}_{\mathcal{R}} t_n \xrightarrow{i}_{\mathcal{R}} \cdots$$

where $t_i \to t_{i+1}/\langle C_i, l_i \to r_i, \sigma_i \rangle$ for some context $C_i$, rule $l_i \to r_i \in \mathcal{R}$ and substitution $\sigma_i$. We show that there is an infinite innermost $\mathcal{R}_1$-rewriting sequence.

Obviously, we can require that all proper subterms of $t_1$ be in $\mathcal{R}$-normal form since we are handling innermost rewriting. This implies that all terms in $\epsilon(t_1)$ are in $\mathcal{R}_2$-normal form. We now show inductively that this is true for all $t_i$ ($i = 1, 2, \ldots$) by analyzing the difference between $\epsilon(t_i)$ and $\epsilon(t_{i+1})$. Let $t \in \epsilon(t_{i+1})$ and we have the following.

- $t$ is in $\epsilon(t_i)$. Then $t$ is in $\mathcal{R}_2$-normal form by induction hypothesis.
- $t$ is not in $\epsilon(t_i)$. Note $t_i = C_i[l_i\sigma_i]$ and $t_{i+1} = C_i[r_i\sigma_i]$. If $t$ contains $r_i\sigma_i$, then there must be some $s$ in $\epsilon(t_i)$ such that $s \to t$. Note $l_i \to r_i$ cannot be in $\mathcal{R}_2$ since $s$ must be in $\mathcal{R}_2$-normal form. Thus, $t$ is in $\mathcal{R}_2$-normal form by Lemma 1. Otherwise, $t$ is dropped from $r_i\sigma_i$. This means that $t$ either equals $s\sigma_i$ for some $s \in \epsilon(r)$ or $t$ is a subterm of $\sigma_i(x)$ for some $x \in \mathrm{dom}(\sigma_i)$. In the latter case, $t$ is obviously in $\mathcal{R}_2$-normal form since this is innermost rewriting. In the former case, $t$ is in $\mathcal{R}_2$-normal form since $s$ is skeleton $\mathcal{R}_2$-normal (note that $\mathcal{R}'$ is a $\mathcal{R}_2$-conservative erasure of $\mathcal{R}$) and $\sigma$ is an $\mathcal{R}_2$-normal (actually $\mathcal{R}$-normal) substitution.

Therefore, for $i = 1, 2, \ldots$, all terms in $\epsilon(t_i)$ are in $\mathcal{R}_2$-normal form. If $l_i \to r_i \in \mathcal{R}_2$, then $|C_i|$ must be a context since $l_i\sigma_i$ would be a subterm of some $t \in \epsilon(t_i)$ otherwise, which contradicts that all terms in $\epsilon(t_i)$ are $\mathcal{R}_2$-normal. Thus, if $t_i \to_{\mathcal{R}_2} t_2$ then $|t_1| \to_{|\mathcal{R}_2|} |t_2|$. Since all rules in $|\mathcal{R}_{22}|$ are strictly ordered under some reduction ordering and all rules in $|\mathcal{R}_1| \cup |\mathcal{R}_{21}|$ are ordered, there must be a $n$ such that for all $i > n$, $l_i \to r_i \notin \mathcal{R}_{22}$. This implies that we have an infinite innermost $\mathcal{R}$-rewriting sequence in which all applied rules are either from $\mathcal{R}_1$ or $\mathcal{R}_{21}$. Contrapositively, the innermost termination of $\mathcal{R}_1 \cup \mathcal{R}_{21}$ implies that of $\mathcal{R}$.

The second part of this theorem is really the same as that of Theorem 1. We thus omit the details. ■

We now present an application of Theorem 2. The following example is taken from the technical report version of [AG97].

Let $R_1$ be a TRS consisting of the following rules,

$$\begin{array}{rlrl} le(0,y) &\to true & pred(s(x)) &\to x \\ le(s(x),0) &\to false & minus(x,0) &\to x \\ le(s(x),s(y)) &\to le(x,y) & minus(x,s(y)) &\to pred(minus(x,y)) \end{array}$$

and $\mathcal{R}_2$ be a TRS consisting of the following rules.

$$\begin{array}{lrl} (1) & gcd(0,y) &\to 0 \\ (2) & gcd(s(x),0) &\to 0 \\ (3) & gcd(s(x),s(y)) &\to ifgcd(le(y,x),s(x),s(y)) \\ (4) & ifgcd(true,s(x),s(y)) &\to gcd(minus(x,y),s(y)) \\ (5) & ifgcd(false,s(x),s(y)) &\to gcd(minus(y,x),s(x)) \end{array}$$

Let $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$. $\mathcal{R}$ is clearly a hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$. We form a $\mathcal{S}$-erasure $\mathcal{R}'$ of $\mathcal{R}$ as follows for $\mathcal{S} = \{(\text{pred}-p-1), (\text{minus}-p-1), (ifgcd-o-1)\}$. $\mathcal{R}' = \mathcal{R}_1' \cup \mathcal{R}_2'$, where $\mathcal{R}_1'$ consists of the following rules

$$\begin{array}{rlrl} le(0,y) &\to true & s(x) &\to x \\ le(s(x),0) &\to false & x &\to x \\ le(s(x),s(y)) &\to le(x,y) & x &\to x \end{array}$$

and $\mathcal{R}_2'$ consists of the following rules.

$$\begin{array}{lrl} (1') & gcd(0,y) &\to 0 \\ (2') & gcd(s(x),0) &\to 0 \\ (3') & gcd(s(x),s(y)) &\to ifgcd(s(x),s(y)) \\ (4') & ifgcd(s(x),s(y)) &\to gcd(x,s(y)) \\ (5') & ifgcd(s(x),s(y)) &\to gcd(y,s(x)) \end{array}$$

It can be readily verified that $\mathcal{R}'$ is a $\mathcal{R}_2$-conservative erasure of $\mathcal{R}$. Under the RPO with the precedence relation $gcd \approx ifgcd$ and $le \succ true, false$, all the rules in $\mathcal{R}'_1$ and the rule $(3')$ can be ordered, and the rules $(1'), (2'), (4')$ and $(5')$ can be strictly ordered. By Theorem 2, the innermost termination of $\mathcal{R}_1 \cup \{(3)\}$ implies that of $\mathcal{R}$. Since $\mathcal{R}_1 \cup \{(3)\}$ can be easily proven terminating with a RPOS, $\mathcal{R}$ is innermost termination. Note that $\mathcal{R}$ is a non-overlapping TRS and thus $\mathcal{R}$ is terminating.

In practice, we may encounter the case where $\mathcal{R}_1 = \emptyset$ when we apply Theorem 1, or $\mathcal{R}_{21} = \emptyset$ when we apply Theorem 2. Let use consider a concrete example. The TRS $\mathcal{R}$ consist of the following single rule.

$$f(g(x)) \rightarrow g(f(f(x)))$$

If we form the $\mathcal{S}$-erasure of $\mathcal{R}$ for $\mathcal{S} = \{(f\text{-p-}1)\}$, we obtain the following TRS $|\mathcal{R}|$,

$$g(x) \rightarrow g(x)$$

which cannot be strictly ordered under any reduction ordering. Therefore, if we apply Theorem 1, we make no progress. However, we can argue that $\mathcal{R}$ is terminating as follows. Suppose that there is an infinite $\mathcal{R}$-rewriting sequence:

$$t_1 \rightarrow_{\mathcal{R}} t_2 \cdots \rightarrow_{\mathcal{R}} t_n \rightarrow_{\mathcal{R}} \cdots$$

We can choose $t_1$ such that all proper subterms of $t_1$ are $\mathcal{R}$-terminating and $t$ is $\mathcal{R}$-terminating for every $t$ if $|t|$ is a subterm of $t_1$. Then $t_1$ must be of form $f(s)$. Since $s$ is $\mathcal{R}$-terminating, there is some $t_n = f(g(s'))$ such that $s \rightarrow^*_{\mathcal{R}} g(s')$ and $t_{n+1} = g(f(f(s')))$. It is clear that $|t_1| = |t_{n+1}| = g(|s'|)$. Given the property of $t_1$, we know that $s'$ is $\mathcal{R}$-terminating. This implies that $t_{n+1}$ is $\mathcal{R}$-terminating, contradicting that the above $\mathcal{R}$-rewriting sequence is infinite. Therefore, there exists no infinite $\mathcal{R}$-rewriting sequence, that is, $\mathcal{R}$ is terminating. We present a formalization of this idea as follows.

**Definition 7.** *Let $\mathcal{S}$ be an erasure TRS. Given a simplification ordering $\succeq_1$ on terms and a quasi precedence relation $\succeq$ on a finite set of function symbols, we can define a (strict) ordering $\succ_2$ as follows. Given $s$ and $t$, $s \succ_2 t$ if either $|s| \succ_1 |t|$, or $|s| \succeq_1 |t|$ and $s$ and $t$ are of form $f(s_1, \ldots, s_m)$ and $t = g(t_1, \ldots, t_n)$, respectively, and $f \succ g$ and*

- *there is no erasure rule in $\mathcal{S}$ is associated with $g$, or*
- *the erasure rule in $\mathcal{S}$ associated with $g$ is an omitting rule, or*
- *$(g\text{-p-}i) \in \mathcal{S}$ and $s \succ_2 t_i$.*

**Lemma 3.** *The ordering $\succ_2$ defined in Definition 7 is well-founded and stable under substitutions.*

*Proof* This is straightforward since $\succeq_1$ is well-founded and stable under substitutions and $\succeq$ is well-founded.

■

**Theorem 3.** *Let $\mathcal{S}$ be an erasure TRS and $\mathcal{R}$ be the hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $|\mathcal{R}_1|$ and $|\mathcal{R}_2|$ are $\mathcal{R}_2$-conservative, and the erasure of every rule in $\mathcal{R}_1$ and $\mathcal{R}_2$ can be ordered under some simplification ordering $\succeq_1$. Let $\succeq$ be a quasi precedence relation on a finite set of function symbols, and we form an ordering $\succ_2$ as described in Definition 7. Assume that for every rule $l \rightarrow r \in \mathcal{R}_2$, either $r$ is skeleton $\mathcal{R}_2$-normal or $l \succ_2 r$. Then the innermost termination of $\mathcal{R}_1$ implies that of $\mathcal{R}_2$. If all the erasure rules in $\mathcal{S}$ are not argument-dropping and for every rule $l \rightarrow r \in \mathcal{R}_2$, either $r$ is skeleton $\mathcal{R}_2$-terminating or $l \succ_2 r$, then the termination of $\mathcal{R}_1$ implies that of $\mathcal{R}_2$.*

*Proof* Assume that $\mathcal{R}_1$ is innermost terminating but $\mathcal{R}$ is not. Let $P(s)$ be a property on terms stating that $s$ is not $\mathcal{R}$-terminating but all proper subterms of $t$ are $\mathcal{R}$-terminating. Since $\succ_2$ is well-founded by Lemma 3, we can choose a term $s$ such that $P(s)$ holds but $P(t)$ fails for every $t$ satisfying $s \succ_2 t$. We can prove by a structural induction the claim that $t$ is $\mathcal{R}$-terminating for every $t$ such that all terms in

$\epsilon(t)$ are in $\mathcal{R}_2$ normal form and $s \succ_2 t$. Please see the proof of Theorem 4 for details. Since $P(s)$ holds, there exists an infinite innermost $\mathcal{R}$-rewriting sequence of the following form,

$$s = f(s_1, \ldots, s_m) \rightarrow^*_{\mathcal{R}} f(s'_1, \ldots, s'_m) = s' \rightarrow_{\mathcal{R}} s'' \rightarrow_{\mathcal{R}} \cdots$$

where $s_i \rightarrow^*_{\mathcal{R}} s'_i$ and $s'_i$ are in $\mathcal{R}$-normal form for $i = 1, \ldots, m$ and $s' = l\sigma$ and $s'' = r\sigma$ and $l \rightarrow r \in \mathcal{R}$. $l \rightarrow r$ must be a rule in $\mathcal{R}_2$ by Lemma 1 (2) and $r$ clearly cannot be skeleton $\mathcal{R}_2$-normal. Therefore, $s' \succ_2 s''$. It can be readily proven that $s \succ_2 s''$ since all rules in $\mathcal{R}$ are ordered under $\succeq_1$. Now let us assume that $s''$ is of form $g(t_1, \ldots, t_n)$. We do a case analysis on the form of $|s''|$.

- There exists no rule in $\mathcal{S}$ associated with $g$. This case is the same as the next one.
- ($g$-o-$(i_1, \ldots, i_k)$) $\in \mathcal{S}$. Then we have

$$|s''| = g(|t_1|, \ldots, |t_{i_1-1}|, |t_{i_1+1}|, \ldots, |t_{i_k-1}|, |t_{i_k+1}|, \ldots, |t_n|).$$

  Note that all terms in $\epsilon(s'')$ are in $\mathcal{R}_2$-normal form since $|\mathcal{R}|$ are $\mathcal{R}_2$-conservative. We have $|s| \succeq_1 |s''| \succ_1 |t_j|$ for $j \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_k\}$ since $\succeq_1$ is a simplification ordering. Hence, $s \succ_2 t_j$. With the above claim, these $t_j$ are $\mathcal{R}$-terminating since $\epsilon(t_j) \subseteq \epsilon(s'')$. Clearly, $t_{i_1}, \ldots, t_{i_k}$ are $\mathcal{R}$-terminating and this implies that all proper subterms of $s''$ are $\mathcal{R}$-terminating. Hence $s''$ is $\mathcal{R}$-terminating since $P(s'')$ holds and $s \succ_2 s''$. We have thus reached a contradiction.
- ($g$-p-$i$) $\in \mathcal{S}$. Then $|s''| = |t_i|$. We have $s' \succ_2 t_i$ by the definition of $\succ_2$, and this can lead to $s \succ_2 t_i$. With the above claim, $t_i$ is $\mathcal{R}$-terminating since $\epsilon(t_i) \subset \epsilon(s'')$. Clearly, $t_j$ are $\mathcal{R}$-terminating for all $j \in \{1, \ldots, i-1, i+1, n\}$. Again, this implies that $s''$ is $\mathcal{R}$ terminating since $P(s'')$ holds and $s \succ_2 s''$. This is a contradiction. terminating.

Therefore, $\mathcal{R}$ must be terminating. It should be straightforward to prove the second part of the theorem. ∎

We present an application of Theorem 3. The following TRS $\mathcal{R}$ is due to Dershowitz.

$$\begin{array}{ll}
(1) & \neg(\neg(x)) \rightarrow x \\
(2) & \neg(x \wedge y) \rightarrow \neg(\neg(\neg(x))) \vee \neg(\neg(\neg(y))) \\
(3) & \neg(x \vee y) \rightarrow \neg(\neg(\neg(x))) \wedge \neg(\neg(\neg(y)))
\end{array}$$

The following is the $\mathcal{S}$-erasure $|\mathcal{R}|$ of $\mathcal{R}$ for $\mathcal{S} = \{(\neg\text{-p-}1)\}$.

$$\begin{array}{c}
x \rightarrow x \\
x \wedge y \rightarrow x \vee y \\
x \vee y \rightarrow x \wedge y
\end{array}$$

Clearly, all rules in $|\mathcal{R}|$ are ordered in the RPO with the precedence $\wedge \approx \vee$. Let $\succeq_1$ denote this RPO. We can form an ordering $\succ_2$ with the precedence relation $\neg \succ \wedge, \vee$ as described in Definition 7. Notice that the right side of (1) is $\mathcal{R}$-skeleton terminating and both rules (2) and (3) can be ordered under $\succ_2$. By Theorem 3, $\mathcal{R}$ is terminating since the rule in $\mathcal{S}$ is not argument-dropping.

Please see Example 6 for a more sophisticated application of Theorem 3

### 3.2 Nondeterministic Erasure Rules

For those who are familiar with the dependency pair approach (DPA) [AG97,AG98], it should be clear that the erasure technique presented so far can be regarded as a closely related idea recast into the framework of *termination through transformation*. However, the following development significantly separates ET from DPA.

Let us now take a look at a limitation of the erasure technique developed so far before proceeding to formulate more sophisticated versions of ET. The rules associated with *if* are the following.

$$\mathit{if}(\mathit{true}, x, y) \rightarrow x \qquad \mathit{if}(\mathit{false}, x, y) \rightarrow y$$

For the example $\mathcal{R}_{\text{pg}}$, we would like to use the erasure TRS $\mathcal{S} = \{(\textit{remove}\text{-p-}2), (\textit{if}\text{-p-}3)\}$ so that we can erase the following rule into $\textit{cons}(y, ys) \to \textit{cons}(y, ys)$.

$$\textit{remove}(x, \textit{cons}(y, ys)) \to \textit{if}(x = y, \textit{remove}(x, ys), \textit{cons}(x, \textit{remove}(y, ys))$$

Unfortunately, we also obtain the erasure $y \to x$ for the rule $\textit{if}(\textit{true}, x, y) \to x$, which is not a legal rewrite rule. This is a severe limitation in practice since *if* is widely used in defining TRSs. We extend the definition of erasure to resolve this problem.

**Definition 8.** *Given a function symbol $f$ with arity $n$ and $1 \le i_i < \cdots < i_k \le n$, the following nondeterministic rule is also an erasure rule.*

$$(f\text{-}p\text{-}(i_1, \ldots, i_k))\ \ f(x_1, \ldots, x_n) \to \{x_{i_1}, \ldots, x_{i_k}\}$$

*This means that $f(x_1, \ldots, x_n)$ can rewrite to $x_{i_j}$ for each $1 \le j \le k$. This rule is not argument-dropping if $\{i_1, \ldots, i_k\} = \{1, \ldots, n\}$.*

With this extension, the erasure $|t|$ of a term $t$ is a multiset of terms, which can be defined as follows.

$$|t| = \begin{cases} \{t\} & \text{if } t \text{ is a variable;} \\ f(|t_1|, \ldots, |t_{i_i - 1}|, |t_{i_i+1}|, \ldots, |t_{i_k-1}|, |t_{i_k+1}|, \ldots, |t_n|) & \\ & \text{if } t = f(t_1, \ldots, t_n) \text{ and } (f\text{-o-}(i_1, \ldots, i_k)) \in \mathcal{S}; \\ |t_{i_1}| \cup \cdots \cup |t_{i_k}| & \text{if } t = f(t_1, \ldots, t_n) \text{ and } (f\text{-p-}(i_1, \ldots, i_k)) \in \mathcal{S}; \\ f(|t_1|, \ldots, |t_n|) & \text{if } t = f(t_1, \ldots, t_n) \text{ and otherwise.} \end{cases}$$

We use the notation $f(|t_1|, \ldots, |t_n|)$ for the multiset

$$\{f(s_1, \ldots, s_n) \mid s_i \in |t_i| \text{ for } 1 \le i \le n\},$$

that is, the multiset of terms $f(s_1, \ldots, s_n)$, where $s_i$ range over $|t_i|$ for $1 \le i \le n$. We also present the definition for $\epsilon(t)$, which is the set of terms erased from $t$.

$$\epsilon(t) = \begin{cases} \emptyset & \text{if } t \text{ is a variable;} \\ \{t_{i_1}, \ldots, t_{i_k}\} \cup \bigcup_{j \in \{1,\ldots,n\} \setminus \{i_1,\ldots,i_k\}} \epsilon(t_j) & \\ & \text{if } t = f(t_1, \ldots, t_n) \text{ and } (f\text{-o-}(i_1, \ldots, i_k)) \in \mathcal{S}; \\ \{t_1, \ldots, t_{i_1-1}, t_{i_1+1}, \ldots, t_{i_k-1}, t_{i_k+1}, \ldots, t_n\} \cup \epsilon(t_{i_1}) \cup \cdots \cup \epsilon(t_{i_k}) & \\ & \text{if } t = f(t_1, \ldots, t_n) \text{ and } (f\text{-p-}(i_1, \ldots, i_k)) \in \mathcal{S}; \\ \bigcup_{j \in \{1,\ldots,n\}} \epsilon(t_j) & \text{if } t = f(t_1, \ldots, t_n) \text{ and otherwise.} \end{cases}$$

In addition, the erasure $|C|$ of context $C$ is a multiset, in which every element is either a context or a term. The erasure $|\sigma|$ of substitution $\sigma$ with a finite domain is defined below.

$$|\sigma| = \{\tau \mid \text{dom}(\tau) = \text{dom}(\sigma) \text{ and } \tau(x) \in |\sigma(x)| \text{ for every } x \in \text{dom}(\tau)\}$$

**Definition 9.** *Let $\succeq$ be an ordering on terms. We extend this ordering to the (nonempty) multisets of terms as follows: $S \succeq^{max}$ ($\succ^{max}$) $T$ if and only if for every $t \in T$ there is an $s \in S$ such that $s \succeq (\succ)\ t$, where $S$ and $T$ stand for the multisets of terms.*

Please notice the difference between $\succeq^{max}$ and $\succeq^{ⓜ}$. For instance, we have $\{c(x)\} \succeq^{max} \{x, c(x)\}$ but $\{c(x)\} \not\succeq^{ⓜ} \{x, c(x)\}$. Also we observe that $\succ^{max}$ is well-founded on the multisets of terms if $\succ$ is well-founded on terms.

Given a rule $l \to r$, the erasure of this rule is $|l| \to |r|$. The erasure of a TRS is defined similarly. Note that we no longer consider the erasure of a rule (TRS) as a rule (TRS), but refer it as a *rule (TRS) erasure.* Given a reduction ordering $\succeq$ on terms, we say that the rule erasure $|l| \to |r|$ is strictly ordered under $\succeq$ if $|l| \succ^{max} |r|$, and it is ordered if $|l| \succeq^{max} |r|$.

For instance, for $\mathcal{S} = \{(\mathit{remove}\text{-p-}2), (\mathit{if}\text{-p-}(2,3))\}$, the $\mathcal{S}$-erasure of $\mathcal{R}_{\mathrm{pg}}$ is the following. We write a term for the singleton set consisting of the term to support transparent syntax.

$$\begin{array}{lrlr} (1') & \mathit{nil} \to \mathit{nil} & (2') & \mathit{cons}(y, ys) \to \{ys, \mathit{cons}(y, ys)\} \\ (3') & \mathit{purge}(\mathit{nil}) \to \mathit{nil} & (4') & \mathit{purge}(\mathit{cons}(x, xs)) \to \mathit{cons}(x, \mathit{purge}(xs)) \end{array}$$

Under the RPO with the precedence $\mathit{purge} \succ \mathit{cons}$, the top two rule erasures are ordered and the rest are strictly ordered.

**Definition 10.** *An ordering $\succeq$ is a weak reduction ordering if its strict part $\succ$ is well-founded and stable under substitutions and $\succeq$ is compatible wrt. term structure and stable under substitutions. Notice that a weak reduction ordering $\succeq$ may not be a reduction ordering since it is not required that $\succ$ be also compatible wrt. term structure.*

**Lemma 4.** *Let $\succeq$ be a weak reduction ordering which is total on ground terms. Given a ground substitution $\sigma$, that is, $\sigma(x)$ is a ground term for every $x \in \mathrm{dom}(\sigma)$, we have the following for every erasure TRS $\mathcal{S}$.*

1. *There exists a substitution $\sigma_{max} \in |\sigma|_{\mathcal{S}}$ such that for every $\tau \in |\sigma|_{\mathcal{S}}$, $\sigma_{max}(x) \succeq \tau(x)$ hold for all $x \in \mathrm{dom}(\sigma)$.*
2. *If $t$ is a term such that $\mathcal{V}ar(t) \subset \mathrm{dom}(\sigma)$, then for every $s_2 \in |t\sigma|_{\mathcal{S}}$ there exists $s_1 \in |t|_{\mathcal{S}}$ such that $s_1\sigma_{max} \succeq s_2$.*

*Proof* For every $x \in \mathrm{dom}(\sigma)$, we can choose a term $t_x \in |\sigma(x)|$ such that $t_x \succeq t$ for all $t \in |\sigma(x)|$ since $\succeq$ is total on ground terms. Let $\sigma_{max}$ be the substitution with domain $\mathrm{dom}(\sigma)$ and $\sigma_{max}(x) = t_x$ for all $x \in \mathrm{dom}(\sigma)$. By the definition of $|\sigma|$, we obtain (1). (2) can be readily proven by a structural induction on $t$. ■

Notice that we actually only require that the weak reduction ordering $\succeq$ be extendable to a total ordering on ground terms. For example, reduction orderings based on RPOS or polynomial interpretations satisfy the requirement.

**Definition 11.** *Let $\mathcal{S}$ be an erasure TRS. For every weak reduction ordering $\succeq$ which is total on ground terms, we can define an ordering $\succeq_{\mathcal{S}}^{max}$ as follows.*

$$t_1 \succeq_{\mathcal{S}}^{max} t_2 \textit{ if and only if } |t_1|_{\mathcal{S}} \succeq^{max} |t_2|_{\mathcal{S}}$$

The next proposition states a crucial property of $\succeq_{\mathcal{S}}^{max}$.

**Proposition 3.** *Given a weak reduction ordering $\succeq$ and an erasure TRS $\mathcal{S}$, the ordering $\succeq_{\mathcal{S}}^{max}$ on terms is also a weak reduction ordering.*

*Proof* By Lemma 4, it is straightforward to prove that both $\succ^{max}$ and $\succeq_{\mathcal{S}}^{max}$ are stable under substitutions. The compatibility of $\succeq_{\mathcal{S}}^{max}$ with term structure follows from the definition of the erasure function $|\cdot|_{\mathcal{S}}$. ■

In general, it does not hold that $t_1 \succ_{\mathcal{S}}^{max} t_2$ implies $C[t_1] \succ_{\mathcal{S}}^{max} C[t_2]$ for every context $C$ even if $\succeq$ is a reduction ordering. Therefore, we cannot infer that $\succeq_{\mathcal{S}}^{max}$ is a reduction ordering under the assumption that $\succeq$ is.

**Theorem 4.** *Let TRS $\mathcal{R}$ be the hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$ and $|\mathcal{R}_1|_{\mathcal{S}}$ and $|\mathcal{R}_2|_{\mathcal{S}}$ are $\mathcal{R}_2$-conservative TRS erasures for some erasure TRS $\mathcal{S}$. Assume that $\succeq$ is a weak reduction ordering which is total on ground terms and $l \succeq_{\mathcal{S}}^{max} r$ for every rule $l \to r \in \mathcal{R}_1$ and $l \succ_{\mathcal{S}}^{max} r$ for every rule $l \to r \in \mathcal{R}_2$. Then the innermost termination of $\mathcal{R}_1$ implies that of $\mathcal{R}$.*

*Proof* Assume that $\mathcal{R}_1$ is innermost terminating but $\mathcal{R}$ is not. Let $P(t)$ be a property on terms stating that $t$ is not innermost $\mathcal{R}$-terminating and every proper subterm of $t$ is innermost terminating. We can choose a ground term $t$ such that $P(t)$ holds and $P(s)$ fails for every term $s$ satisfying $t \succ_{\mathcal{S}}^{max} s$ since $\succ_{\mathcal{S}}^{max}$ is well-founded. We now prove by a structural induction on $s$ that $s$ is innermost terminating if $t \succ_{\mathcal{S}}^{max} s$ and all terms in $\epsilon(s)_{\mathcal{S}}$ are innermost terminating. Assume that $s$ is of form $f(s_1, \ldots, s_n)$. We do a case analysis on the form of $|s|$.

- No erasure rule in $\mathcal{S}$ is associated with $f$. This case is the same as the next one.
- The erasure rule ($f$-o-$(i_1, \ldots, i_k)$) is in $\mathcal{S}$. Then

$$|s| = f(|s_1|, \ldots, |s_{i_1-1}|, |s_{i_1+1}|, \ldots, |s_{i_k-1}|, |s_{i_k+1}|, \ldots, |s_n|).$$

  $\succeq$ must be a simplification ordering on ground terms since it is total on them. Therefore, for every $j \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_k\}$, $t \succ_{\mathcal{S}}^{max} s \succeq_{\mathcal{S}}^{max} s_j$, and thus $s_j$ is innermost terminating since $\epsilon(s_j) \subset \epsilon(s)$ implies that all terms in $\epsilon(s_j)$ are innermost terminating. Also $s_{i_j}$ are innermost terminating for $1 \leq j \leq k$ since they are in $\epsilon(s)$. Therefore, all proper subterms of $s$ are innermost terminating. Given the property of $t$, $s$ is innermost $\mathcal{R}$-terminating.
- The erasure rule ($f$-p-$(i_1, \ldots, i_k)$) is in $\mathcal{S}$. This is similar to the previous case.

Thus we have proven the claim that $s$ is innermost $\mathcal{R}$-terminating if $t \succ_{\mathcal{S}}^{max} s$ and all terms in $\epsilon(s)$ are innermost terminating.

Assume that $t$ is of form $f(t_1, \ldots, t_n)$. Since $t$ is not innermost $\mathcal{R}$-terminating and all proper terms of $t$ are innermost $\mathcal{R}$-terminating, there is an infinite innermost rewriting sequence beginning with the following form,

$$t = f(t_1, \ldots, t_n) \overset{i}{\rightarrow}^*_{\mathcal{R}} f(t'_1, \ldots, t'_n) = t' \overset{i}{\rightarrow}_{\mathcal{R}} t''$$

where $t_i \overset{i}{\rightarrow}^*_{\mathcal{R}} t'_i$ and $t'_i$ are in $\mathcal{R}$-normal form for $1 \leq i \leq n$, and $t' = l\sigma$ and $t'' = r\sigma$ for some $l \rightarrow r \in \mathcal{R}$. Note that $f$ cannot be a defined function symbol in $\mathcal{R}_1$ by Lemma 1 (2). Hence, $l \rightarrow r \in \mathcal{R}_2$, and this implies $l \succ_{\mathcal{S}}^{max} r$. Therefore, we have $t \succeq_{\mathcal{S}}^{max} t' \succ_{\mathcal{S}}^{max} t''$. Note that all terms in $\epsilon(t'')$ are $\mathcal{R}_2$-normal since all terms in $\epsilon(r)$ are skeleton $\mathcal{R}_2$-normal. Therefore, all terms in $\epsilon(t'')$ are innermost $\mathcal{R}$-terminating by Lemma 1 (2). This implies that $t''$ is $\mathcal{R}$-innermost terminating by the above proven claim, contradicting the assumption that $t$ is not innermost $\mathcal{R}$-terminating. Therefore $\mathcal{R}$ is innermost terminating. ■

For instance, $\mathcal{R}_{\text{pg}}$ can be readily proven innermost terminating with Theorem 4. We will present in Appendix A more realistic examples which can be proven (innermost) terminating with the applications of Theorem 1, 2 and 4. We regard these theorems as the major contribution of this paper.

We now present a theorem to demonstrate that the erasure technique can also be directly applied to termination proofs. We first establish a lemma needed later.

**Lemma 5.** *Let $\succeq$ be an ordering on terms. For multisets $S$, $T_1$ and $T_2$ of terms, if $T_1 \succ^{max} T_2$, then $S \cup T_1 \succ^{ⓜ} S \cup T_2$.*

*Proof* The lemma immediately follows from the definition of $\succ^{max}$ and $\succ^{ⓜ}$. ■

**Theorem 5.** *Let $\mathcal{S}$ be an erasure TRS in which all the rules are not argument-dropping and $\succeq$ be a reduction ordering which is total on ground terms. Assume $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ such that*

- *$|l| = |r|$ for all rules $l \rightarrow r \in \mathcal{R}_1$, and*
- *$|l| \setminus |r| \succ^{max} |r| \setminus |l|$ for all rules $l \rightarrow r \in \mathcal{R}_2$.*

*Then the termination of $\mathcal{R}_1$ implies that of $\mathcal{R}_2$.*

*Proof* (sketch) Assume that $t_1 \to_{\mathcal{R}} t_2/\langle C, l \to r, \sigma\rangle$. It suffices to prove that $|t_1| \succeq^{ⓜ} |t_2|$ if $l \to r \in \mathcal{R}_1$ and $|t_1| \succ^{ⓜ} |t_2|$ if $l \to r \in \mathcal{R}_2$.

If $l \to r \in \mathcal{R}_1$, then $|l| = |r|$, which implies that $|t_1| = |t_2|$. We now assume that $l \to r \in \mathcal{R}_2$. Let $|C| = \{C_1, \ldots, C_k\}$. Since all rules in $\mathcal{S}$ are not argument-dropping, every $C_i$ is a context for $1 \le i \le k$. Let us define multisets $S, T_1, T_2$ of terms as follows.

$$\begin{aligned} S &= \textstyle\bigcup_{1\le i\le k}\{C_i[s] \mid s \in |t\sigma| \text{ and } t \in |l| \cap |r|\} \\ T_1 &= \textstyle\bigcup_{1\le i\le k}\{C_i[s] \mid s \in |t\sigma| \text{ and } t \in |l| \setminus |r|\} \\ T_2 &= \textstyle\bigcup_{1\le i\le k}\{C_i[s] \mid s \in |t\sigma| \text{ and } t \in |r| \setminus |l|\} \end{aligned}$$

It can be readily proven with Lemma 4 that $T_1 \succ^{max} T_2$ holds. Therefore, Also we can show $|t_1| = S \cup T_1$ and $|t_2| = S \cup T_2$. By Lemma 5, we have $|t_1| \succ^{ⓜ} |t_2|$. ■

Theorem 5 immediately strengthens Theorem 12 (1) in [Zan94], where it is required that $f$ does not occur in $l$ for every $l \to r \in \mathcal{R}$ if the erasure rule ($f$-p-$(1, \ldots, n)$) is included in $\mathcal{S}$ [3]. Applications of Theorem 5 can be found in Appendix A.

## 4 Related Work

There is a large number of results in the literature concerning termination proofs for various modular combinations of TRSs. We refer the reader to [Der95] for some clean explanation on many significant results in this area. The general scenario is to prove the termination of $\mathcal{R}_1 \cup \mathcal{R}_2$ for terminating TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ under some assumption on the relation between $\mathcal{R}_1$ and $\mathcal{R}_2$. We have found that most of the results such as the ones mentioned in [Der95], though interesting, make assumptions about $\mathcal{R}_1$ and $\mathcal{R}_2$ which are too strong for the purpose of verifying the termination of hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$, sometimes.

We are most interested in the case of hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$ where the defined function symbols in $\mathcal{R}_1$ are used in $\mathcal{R}_2$ in an essential way since this closely resembles the structure of a functional or logic program. This almost forces us to know the semantics of $\mathcal{R}_1$ to certain extent in order to prove the termination of the combined system. ET is proposed to address the issue in a (very) restricted manner. For instance, the use of the projection rule (*remove*-p-2) in the $\mathcal{R}_{\text{pg}}$ example is simply to test that $remove(x, ys)$ can never return a list of length greater than that of $ys$. This test succeeds because the generated erasure of $\mathcal{R}_{\text{pg}}$ can be ordered. Let $\mathcal{R}'_{\text{pg}}$ be $\mathcal{R}_{\text{pg}}$ in which the rule $remove(x, nil) \to nil$ is replaced with another rule $remove(x, nil) \to cons(x, nil)$, then the test will fail on $\mathcal{R}'_{\text{pg}}$ since we cannot order $nil \to cons(x, nil)$. Notice that $\mathcal{R}'_{\text{pg}}$ is *not* terminating. This immediately implies that none of the results mentioned in [Der95] can give modular termination proofs for $\mathcal{R}_{\text{pg}}$. If they could, they would also prove this for $\mathcal{R}'_{\text{pg}}$ since $\mathcal{R}_{\text{pg}}$ and $\mathcal{R}'_{\text{pg}}$ exhibit the very same characteristics to them.

The dependency pair approach (DPA) [AG97,AG98], which inspired our work on erasure, deserves special mentioning. We regard ET as a similar idea cast into the general framework of *termination through transformation*. The technical explanation is that, to a large extent, erasure amounts to the use of weak reduction orderings, which are referred as weakly monotonic orderings stable under substitutions in papers on DPA. In general, DPA seems more powerful than ET but it is also (in our opinion) more involved. For instance, DPA uses unification to detect circles of dependency pairs and the set of usable rules, but this is currently unavailable in ET. However, this seems to be a less significant issue so far in our experiment, especially, after we combine ET with the freezing technique [Xi98]. We also plan to incorporate similar ideas into ET if the needs appear. We feel that the most significant advantage of ET over DPA is the availability of nondeterministic erasure rules. Because of the lack of a similar feature,

[3] A strengthened version of this theorem is proven in [MOZ96] which does allow the occurrences of $f$ on the left-hand sides of the rules, but it is nonetheless essentially different from Theorem 5. Please see Example 4 in Appendix A.

DPA is often awkward in handling conditional *if*. For instance, we must order the following rule

$$remove(x, cons(y, ys)) \to if(x = y, remove(x, ys), cons(x, remove(y, ys)))$$

with a weakly monotone ordering if $\mathcal{R}_{\rm pg}$ is to be proven terminating using DPA. Suppose that we use RPOS as the underlying approach to ordering the rule. We cannot assume $remove \succ cons$ in the precedence relation since this prevents us from strictly ordering the following generated dependency pair $PURGE(cons(x, xs)) > PURGE(remove(x, xs))$. If we map $if(b, x, y)$ to $x$ $(y)$, then the rule $if(false, x, y) \to y$ $(if(true, x, y) \to x)$ cannot be ordered. As a consequence, the *if* function often needs to be "preprocessed" away when DPA is applied because it is difficult to synthesize a weakly monotone ordering based on RPOS or polynomial interpretations in the presence of *if* to order the generated dependency pairs. For instance, the following rules are introduced in the technical report version of [AG97] for handling the purge function example.

$$\begin{aligned} remove(x, cons(y, ys)) &\to ifremove(x = y, x, cons(y, ys)) \\ ifremove(true, x, cons(y, ys)) &\to remove(x, ys) \\ ifremove(false, x, cons(y, ys)) &\to cons(y, remove(x, ys)) \end{aligned}$$

Though the argument is that the introduction of these rules is to forbid rewriting terms under if-branches until the condition is resolved, we feel that this is also a bit unnatural at least since realistic TRSs are seldom formed in such a manner. Notice that the termination of $\mathcal{R}_{\rm pg}$ is independent of whether we rewrite terms under if-branches or not. It seems straightforward to make use of the weak reduction ordering $\succeq^{max}$ in DPA for handling *if*, and this can elegantly resolve the above issue. We will use some concrete examples to further compare ET or ET plus the freezing technique with DPA in Appendix A.

The use of projection erasure rules bears some resemblance to *distribution elimination* [Zan94], but there are also many significant differences. Although it is clearly possible, there seems no attempt in [Zan94] to construct the ordering $\succeq^{max}$ from a given weak reduction ordering $\succeq$, which we regard as a significant contribution of the paper. Also we mention that the use of an omitting rule ($f$-o-$(1, \ldots, n)$) in case $\mathcal{A}r(f) = n$ casually relates to *dummy elimination* [Fer96].

## 5 Conclusion

We have presented a technique named erasure to facilitate the termination and innermost termination proofs, and this technique is inspired by the dependency pair approach in the literature. The erasure technique (ET) is simple to apply and effective in practice, and therefore is reasonable to expect that ET can be combined with other automated approaches to termination proofs for TRS such as freezing [Xi98]. However, we observe in practice that it is even difficult to scale an approach as simple as RPOS. This makes us believe that a more promising direction is to apply ET interactively. In this respect, we have tried ET extensively on various TRSs and the results are encouraging. We present some examples in Appendix A to substantiate this claim.

In general, we are highly motivated to look for approaches to termination proofs for TRSs which are simple and effective. We intend to integrate these approaches into an interactive termination prover for TRSs. The user may be required to interact when applying these approaches but the needed interaction should not be overwhelming. We view this as promising direction to pursue so as to address the following dilemma: too much automation can severely hinder the scalability of a termination proof procedure for TRSs while too little can easily lead to an amount of required interaction which is simply overwhelming for the user. This should be especially clear to those who have used interactive theorem provers such as PVS [ORR+96] or Isabelle [Law94] for proving the termination of recursively defined functions.

## References

[AG97] Thomas Arts and Jürgen Giesl. Proving innermost normalisation automatically. In Hubert Comon, editor, *Proceedings of the 8th Conference on Rewriting Techniques and Applications*, pages 157–171.

Springer-Verlag LNCS 1232, 1997. An extended version is available as Technical Report IBN 96/39. Technische Hochschule Darmstadt.

[AG98] Thomas Arts and Jürgen Giesl. Modularity of termination using dependency pairs. In Tobias Nipkow, editor, *Proceedings of the 9th Conference on Rewriting Techniques and Applications*, pages 226–240. Springer-Verlag LNCS 1379, 1998.

[AZ95] Thomas Arts and Hans Zantema. Termination of logical programs using semantic unification. In *Proceedings of the 5th International Workshop on Logical Program Synthesis and Transformation*, pages 219–233, Utrecht, 1995. Springer-Verlag LNCS 1048.

[BL87] Ahkem BenCherifa and Pierre Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *SCP*, 9(2):137–160, 1987.

[BL90] Francoise Bellegarde and Pierre Lescanne. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing*, 1:79–96, 1990.

[CHR92] P.-L. Curien, T. Hardin, and A. Ríos. Strong normalization of substitutions. In I. M. Havel and V. Koubek, editors, *Proceedings of Mathematical Foundations of Computer Science*, pages 209–217. Springer-Verlag LNCS 629, 1992.

[Der82] Nachum Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[Der87] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

[Der95] Nachum Dershowitz. Hierarchical termination. In *Proceedings of the 4th International Workshop on Conditional and Typed Rewriting Systems*, pages 89–105. Springer-Verlag LNCS 968, 1995.

[DJ91] Nachum Dershowitz and Jean-Pierre Jouannaud. Notations for rewriting. *EATCS*, 43:162–172, 1991.

[Fer96] Maria Ferreira. Dummy elimination in equational rewriting. In Harald Ganzinger, editor, *Proceedings of the 7th Conference on Rewriting Techniques and Applications*, pages 78–92. Springer-Verlag LNCS 1103, 1996.

[Gra95] Bernhard Gramlich. Abstract relations between restricted termination and confluence properties of rewrite systems. *Fundamenta Informaticae*, 24(1/2):2–23, 1995.

[KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[KL80] Sam Kamin and Jean-Jacques Lévy. Attempts for generalizing the recursive path orderings. Unpublished manuscript, February 1980.

[Lan79] Dallas Lankford. On proving term rewriting systems are noetherian. Technical Report Memo MTP-3, Louisiana Tech. University, 1979.

[Law94] Paul Lawrence. *Isabelle: A Generic Theorem Prover*. Springer-Verlag LNCS 828, 1994.

[MOZ96] Aart Middeldorp, Hitoshi Ohsaki, and Hans Zantema. Transforming termination by self-labelling. In *Proceedings of 13th International Conference on Automated Deduction*, pages 373–387, New Brunswick, July/August 1996. Springer-Verlag LNCS 1104.

[MTHM97] Robin Milner, Mads Tofte, Robert W. Harper, and D. MacQueen. *The Definition of Standard ML*. MIT Press, Cambridge, Massachusetts, 1997.

[ORR+96] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. PVS: Combining specification, proof checking, and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer-Aided Verification, CAV '96*, pages 411–414, New Brunswick, NJ, July/August 1996. Springer-Verlag LNCS 1102.

[Pla78] David Plaisted. A recursively defined ordering for proving termination of term rewriting systems. Technical Report UIUC DCS-R-78-943, Univ. of Illinois at Urbana-Champaign, 1978.

[Ste95a] Joachim Steinbach. Automatic termination proofs with transformation orderings. In *Proceedings of the 6th Conference on Rewriting Techniques and Applications*, pages 11–25. Springer-Verlag LNCS 914, 1995.

[Ste95b] Joachim Steinbach. Simplification orderings: History of results. *Fundamenta Informaticae*, 24:47–87, 1995.

[SX98] Joachim Steinbach and Hongwei Xi. Freezing – termination proofs for classical, context sensitive and innermost rewriting. Technical report, Technische Universität München, 1998.

[Xi98] Hongwei Xi. Towards automated termination proofs through "freezing". In *9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 271–285, March–April 1998.

[Zan94] Hans Zantema. Termination of term rewriting: interpretation and type elimination. *Journal of Symbolic Computation*, 17:23–50, 1994.

[Zan95] Hans Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.

## A Examples

*Example 1.* We often combine ET with the freezing technique [Xi98] in practice. Let $\mathcal{R}_{\text{fact}}$ be the following TRS [KL80].

$$\begin{array}{rclcrcl} p(s(0)) & \to & 0 & \quad & fact(0) & \to & s(0) \\ p(s(s(x))) & \to & s(p(s(x))) & & fact(s(x)) & \to & s(x) * fact(p(s(x))) \end{array}$$

The following $\mathcal{R}^1_{\text{fact}}$ is a $(p, s, \underline{ps}, 1)$-frozen version of $\mathcal{R}_{\text{fact}}$, and therefore the termination of $\mathcal{R}^1_{\text{fact}}$ implies that of $\mathcal{R}_{\text{fact}}$.

$$\begin{array}{lrcllrcl} (1) & p(s(0)) & \to & 0 & (4) & \underline{ps}(s(x)) & \to & s(\underline{ps}(x)) \\ (2) & \underline{ps}(0) & \to & 0 & (5) & fact(0) & \to & s(0) \\ (3) & p(s(s(x))) & \to & s(\underline{ps}(x)) & (6) & fact(s(x)) & \to & s(x) * fact(\underline{ps}(x)) \end{array}$$

The following $\mathcal{R}^2_{\text{fact}}$ is the $\mathcal{S}$-erasure of $\mathcal{R}^1_{\text{fact}}$ for $\mathcal{S} = \{\underline{ps}\text{-p-1}\}$.

$$\begin{array}{lrcllrcl} (1') & p(s(0)) & \to & 0 & (4') & s(x) & \to & s(x) \\ (2') & 0 & \to & 0 & (5') & fact(0) & \to & s(0) \\ (3') & p(s(s(x))) & \to & s(x) & (6') & fact(s(x)) & \to & s(x) * fact(x) \end{array}$$

Under the RPO with the precedence $fact \succ *$, rules $(2')$ and $(4')$ can be ordered and the rest of the rules can be strictly ordered. Since the TRS consisting of rules (2) and (4) is obviously terminating, the termination of $\mathcal{R}^1_{\text{fact}}$ follows from Theorem 1. Therefore, $\mathcal{R}_{\text{fact}}$ is terminating by a theorem on the freezing technique.

If we apply DPA to $\mathcal{R}_{\text{fact}}$, the following dependency pair is generated.

$$FACT(s(x)) > FACT(p(s(x)))$$

It is unclear how this can be strictly ordered since we cannot project away the argument of $p$ because of the existence of the rule $p(s(s(x))) \to s(p(s(x)))$. If one argues that this example is too contrived, then the following example exhibits the same characteristics.

*Example 2.* In the following TRS $\mathcal{R}_{\log}$, $h(n) = \lfloor n/2 \rfloor$ for every natural number $n$, and $\log(n) = 1 + \log_2(n)$ for $n > 0$.

$$\begin{array}{rclcrcl} h(0) & \to & 0 & \quad & \log(0) & \to & 0 \\ h(s(0)) & \to & 0 & & \log(s(x)) & \to & s(\log(h(s(x)))) \\ h(s(s(x))) & \to & s(h(x)) & & & & \end{array}$$

The last rule is self-embedding, and therefore the termination of this TRS cannot be proven with a simplification ordering. We form an $(h, s, 1, \underline{hs})$-frozen version $\mathcal{R}^1_{\log}$ of $\mathcal{R}_{\log}$ as follows.

$$\begin{array}{lrcllrcl} (1) & h(0) & \to & 0 & (5) & \underline{hs}(s(x)) & \to & s(h(x)) \\ (2) & h(s(0)) & \to & 0 & (6) & \log(0) & \to & 0 \\ (3) & \underline{hs}(0) & \to & 0 & (7) & \log(s(x)) & \to & s(\log(\underline{hs}(x))) \\ (4) & h(s(s(x))) & \to & s(h(x)) & & & & \end{array}$$

We can prove the termination of $\mathcal{R}^1_{\log}$ by forming its $\mathcal{S}$-erasure for $\mathcal{S} = \{h\text{-p-1}, \underline{hs}\text{-p-1}\}$. This then implies the termination of $\mathcal{R}_{\log}$. We omit the details that are straightforward to fill in. Notice that this example can not handled by DPA for the same reason as explained in the previous example.

In general, we intend to apply various transformations for proving the (innermost) termination of a TRS $\mathcal{R}$. We generate a chain of TRSs $\mathcal{R} = \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n$ such that the (innermost) termination of $\mathcal{R}_{i+1}$ implies that of $\mathcal{R}_i$ for $1 \le i < n$ and the (innermost) termination of $\mathcal{R}_n$ can be proven with some basic approach such as RPOS or polynomial interpretations. The problem with DPA is that it generates a set of dependency pairs rather than a TRS, and therefore it is difficult to be combined with other transformational approaches.

*Example 3.* The following TRS $\mathcal{R}_{\text{qs}}$ defines a quicksort function on lists. Let $\mathcal{R}_1$ consist of all these rules except the last 2, and $\mathcal{R}_2$ consist of the last 2 rules. Then $\mathcal{R}_{\text{qs}}$ is the hierarchical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$.

$$
\begin{array}{rr}
(1) & \mathit{if}(\mathit{true}, x, y) \to x \\
(2) & \mathit{if}(\mathit{false}, x, y) \to y \\
(3) & 0 \leq x \to \mathit{true} \\
(4) & s(x) \leq 0 \to \mathit{false} \\
(5) & s(x) \leq s(y) \to x \leq y \\
(6) & \mathit{gte}(x, []) \to [] \\
(7) & \mathit{gte}(x, y :: ys) \to \mathit{if}(x \leq y, y :: \mathit{gte}(x, ys), \mathit{gte}(x, ys)) \\
(8) & \mathit{lt}(x, []) \to [] \\
(9) & \mathit{lt}(x, y :: ys) \to \mathit{if}(x \leq y, \mathit{lt}(x, ys), y :: \mathit{lt}(x, ys)) \\
(10) & [] \mathbin{@} ys \to ys \\
(11) & (x :: xs) \mathbin{@} ys \to x :: (xs \mathbin{@} ys) \\
(12) & \mathit{quicksort}([]) \to [] \\
(13) & \mathit{quicksort}(x :: xs) \to \mathit{quicksort}(\mathit{lt}(x, xs)) \mathbin{@} [x] \mathbin{@} \mathit{quicksort}(\mathit{gte}(x, xs))
\end{array}
$$

The following is the $\mathcal{S}$-erasure of $\mathcal{R}$ for $\mathcal{S} = \{(\mathit{if}\text{-p-}(2,3)), (\mathit{gte}\text{-p-}2), (\mathit{lt}\text{-p-}2)\}$.

$$
\begin{array}{rr}
(1') & \{x, y\} \to x \\
(2') & \{x, y\} \to y \\
(3') & 0 \leq x \to \mathit{true} \\
(4') & s(x) \leq 0 \to \mathit{false} \\
(5') & s(x) \leq s(y) \to x \leq y \\
(6') & [] \to [] \\
(7') & y :: ys \to \{y :: ys, ys\} \\
(8') & [] \to [] \\
(9') & y :: ys \to \{ys, y :: ys\} \\
(10') & [] \mathbin{@} ys \to ys \\
(11') & (x :: xs) \mathbin{@} ys \to x :: (xs \mathbin{@} ys) \\
(12') & \mathit{quicksort}([]) \to [] \\
(13') & \mathit{quicksort}(x :: xs) \to \mathit{quicksort}(xs) \mathbin{@} [x] \mathbin{@} \mathit{quicksort}(xs)
\end{array}
$$

Under the RPO with precedence $\mathit{quicksort} \succ @, \leq \succ \mathit{true}, \mathit{false}$, all rule erasures in $\mathcal{R}_1$ can be ordered and all rule erasures in $\mathcal{R}_2$ can be strictly ordered. By Theorem 4, the innermost termination of $\mathcal{R}_{\text{qs}}$ follows from that of $\mathcal{R}_1$. It can be readily proven with a RPO that $\mathcal{R}_1$ is (innermost) terminating, and therefore $\mathcal{R}_{\text{qs}}$ is innermost terminating. This implies that $\mathcal{R}_{\text{qs}}$ is terminating since it is non-overlapping.

A similar example also appears in the technical report version of [AG97], but *if* is "preprocessed" away. The termination of that example can be readily proven with Theorem 2.

*Example 4.* Let $\mathcal{R}_1$ be the following TRS.

$$
\begin{array}{rr}
(1) & f(\emptyset) \to \emptyset \\
(2) & f(\mathit{branch}(\emptyset, x)) \to \mathit{branch}(\emptyset, f(x)) \\
(3) & f(\mathit{branch}(\mathit{branch}(x, y), z)) \to f(\mathit{branch}(x, \mathit{branch}(y, z))) \\
(4) & g(\emptyset) \to \emptyset \\
(5) & g(\mathit{branch}(x, \emptyset)) \to \mathit{branch}(\emptyset, g(x)) \\
(6) & g(\mathit{branch}(x, \mathit{branch}(y, z))) \to g(\mathit{branch}(\mathit{branch}(x, y), z))
\end{array}
$$

The following is the $\mathcal{S}$-erasure of $\mathcal{R}_1$ for $\mathcal{S} = \{branch\text{-p-}(1,2)\}$.

$$\begin{array}{lr}
(1') & f(\emptyset) \to \emptyset \\
(2') & \{f(\emptyset), f(x)\} \to \{\emptyset, f(x)\} \\
(3') & \{f(x), f(y), f(z)\} \to \{f(x), f(y), f(z)\} \\
(4') & g(\emptyset) \to \emptyset \\
(5') & \{g(x), g(\emptyset)\} \to \{\emptyset, g(x)\} \\
(6') & \{g(x), g(y), g(z)\} \to \{g(x), g(y), g(z)\}
\end{array}$$

By Theorem 5, the termination of $\mathcal{R}_1$ follows from the termination of $\mathcal{R}_2 = \{(3), (6)\}$. We now construct a TRS $\mathcal{R}_3$ below, which is an $(f, branch, 1, \underline{fbranch})$-frozen version of $\mathcal{R}_2$.

$$\begin{array}{r}
\underline{fbranch}(branch(x, y), z) \to \underline{fbranch}(x, branch(y, z)) \\
g(branch(x, branch(y, z))) \to g(branch(branch(x, y), z))
\end{array}$$

The termination of $\mathcal{R}_3$ is easily proven with a $\mathsf{RPOS}$, and therefore, $\mathcal{R}_1$ is terminating. We point out that it would be greatly involved (though possible) if we applied the freezing technique to $\mathcal{R}_1$ directly.

Notice that Theorem 12 [Zan94] cannot be applied to this example since $f$ has occurrences on the left-hand sides of the rules. The strengthened version of this theorem in [MOZ96] cannot handle this example, either.

*Example 5.* The termination of the following TRS $\sigma_0$ describes the process of substitution in combinatory logic, and the proof for the termination of $\sigma_0$ in [CHR92] is involved. Some simplified proofs have been given in [Zan94,Zan95].

$$\begin{array}{llll}
(1) & \lambda(x) \circ y \to \lambda(x \circ (1 \cdot (y \circ \uparrow))) & (5) & 1 \circ id \to 1 \\
(2) & (x \cdot y) \circ z \to (x \circ z) \cdot (y \circ z) & (6) & 1 \circ (x \cdot y) \to x \\
(3) & (x \circ y) \circ z \to x \circ (y \circ z) & (7) & \uparrow \circ (x \cdot y) \to y \\
(4) & id \circ x \to x & &
\end{array}$$

The following is the $\mathcal{S}$-erasure of $\sigma_0$ for $\mathcal{S} = \{(\cdot\text{-p-}(1,2))\}$.

$$\begin{array}{llll}
(1') & \lambda(x) \circ y \to \{\lambda(x \circ 1), \lambda(x \circ (y \circ \uparrow))\} & (5') & 1 \circ id \to 1 \\
(2') & \{x \circ z, y \circ z\} \to \{x \circ z, y \circ z\} & (6') & \{1 \circ x, 1 \circ y\} \to x \\
(3') & (x \circ y) \circ z \to x \circ (y \circ z) & (7') & \{\uparrow \circ x, \uparrow \circ y\} \to y \\
(4') & id \circ x \to x & &
\end{array}$$

As shown in [Zan94], all the rule erasures except the second one can be strictly ordered under a total ordering. By Theorem 5, the termination of $\sigma_0$ follows from the termination of the TRS consisting of the rule $(x \cdot y) \circ z \to (x \circ z) \cdot (y \circ z)$, which is obvious. Notice that the distribution elimination technique [Zan94] cannot be directly applied to this example because of the occurrences of $\cdot$ on the left-hand sides of some rules. If we replace the last rule in $\sigma_0$ with $\uparrow \circ (x \cdot y) \to y \cdot x$, then the strategy used in [Zan94] would no longer work but Theorem 5 could still be applied.

*Example 6.* The following example is adopted from the technical report version of [AG97], where it is formed as a variation of an algorithm in [**?**]. The purpose of the function $rename(x, y, t)$ is to replace every free occurrence of the variable $x$ in the term $t$ with the variable $y$.

$$
\begin{array}{lr}
(1) & \mathit{true} \wedge y \rightarrow y \\
(2) & \mathit{false} \wedge y \rightarrow \mathit{false} \\
(3) & [] = [] \rightarrow \mathit{true} \\
(4) & (x :: xs) = [] \rightarrow \mathit{false} \\
(5) & [] = (y :: ys) \rightarrow \mathit{false} \\
(6) & (x :: xs) = (y :: ys) \rightarrow (x = y) \wedge (xs = ys) \\
(7) & var(xs) = var(ys) \rightarrow xs = ys \\
(8) & var(xs) = \mathit{apply}(s,t) \rightarrow \mathit{false} \\
(9) & var(xs) = lambda(x,s) \rightarrow \mathit{false} \\
(10) & \mathit{apply}(s,t) = var(ys) \rightarrow \mathit{false} \\
(11) & \mathit{apply}(s,t) = \mathit{apply}(u,v) \rightarrow (s = u) \wedge (t = v) \\
(12) & \mathit{apply}(s,t) = lambda(x,u) \rightarrow \mathit{false} \\
(13) & lambda(x,s) = var(ys) \rightarrow \mathit{false} \\
(14) & lambda(x,s) = \mathit{apply}(u,v) \rightarrow \mathit{false} \\
(15) & lambda(x,s) = lambda(y,t) \rightarrow (x = y) \wedge (s = t) \\
(16) & \mathit{if}(\mathit{true}, var(xs), var(ys)) \rightarrow var(xs) \\
(17) & \mathit{if}(\mathit{false}, var(xs), var(ys)) \rightarrow var(ys) \\
(18) & rename(var(xs), var(ys), var(zs)) \rightarrow \mathit{if}(xs = zs, var(ys), var(zs)) \\
(19) & rename(x,y,\mathit{apply}(s,t)) \rightarrow \mathit{apply}(rename(x,y,s), rename(x,y,t)) \\
(20) & rename(x,y,lambda(z,t)) \rightarrow lambda(\bullet, rename(x,y,rename(z,\bullet,t)))
\end{array}
$$

Note that $\bullet$ in rule (20) stands for $var([x,y,lambda(z,t)])$. Let $\mathcal{R}_1$ consist of the first 17 rules and $\mathcal{R}_2$ consist of the rest of rules. Then $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ is a hierachical combination of $\mathcal{R}_1$ and $\mathcal{R}_2$. Clearly, $\mathcal{R}_1$ can be proven terminating with some RPO. We form the following $\mathcal{S}$-erasure $|\mathcal{R}|$ of $\mathcal{R}$ for $\mathcal{S} = \{(\wedge\text{-p-}2), (=\text{-o-}(1,2)), (var\text{-o-}1), (\mathit{if}\text{-o-}(1,2,3)), (rename\text{-p-}3), (lambda\text{-o-}1)\}$.

$$
\begin{array}{lr}
(1') & y \rightarrow y \\
(2') & y \rightarrow \mathit{false} \\
(3') & = \rightarrow \mathit{true} \\
(4') & = \rightarrow \mathit{false} \\
(5') & = \rightarrow \mathit{false} \\
(6') & = \rightarrow = \\
(7') & = \rightarrow = \\
(8') & = \rightarrow \mathit{false} \\
(9') & = \rightarrow \mathit{false} \\
(10') & = \rightarrow \mathit{false} \\
(11') & = \rightarrow = \\
(12') & = \rightarrow \mathit{false} \\
(13') & = \rightarrow \mathit{false} \\
(14') & = \rightarrow \mathit{false} \\
(15') & = \rightarrow = \\
(16') & \mathit{if} \rightarrow var \\
(17') & \mathit{if} \rightarrow var \\
(18') & var \rightarrow \mathit{if} \\
(19') & \mathit{apply}(s,t) \rightarrow \mathit{apply}(s,t) \\
(20') & lambda(t) \rightarrow lambda(t)
\end{array}
$$

It can be readily verified that $|\mathcal{R}|$ is $\mathcal{R}_2$-conservative. Under the RPO with the precedence relation $\mathit{true} \approx \mathit{false} \approx = \approx var = \mathit{if}$, all the rules can be ordered. Note that rule $(2')$ is ordered because we can require that *false* be a constant with the lowest precedence. Let $\succeq_1$ denote this RPO. We can then form an ordering $\succ_2$ with the precedence $rename \succ \mathit{apply}, lambda$ as described in Definition 7. Then the right side of rule (18) is $\mathcal{R}_2$-skeleton normal, and both rules (19) and (20) can be ordered under $\succ_2$. Therefore, $\mathcal{R}$ is terminating by Theorem 3.