# Termination transformation by tree lifting ordering

Takahito Aoto and Yoshihito Toyama

School of Information Science, JAIST
Tatsunokuchi, Ishikawa 923-12, Japan
{aoto, toyama}@jaist.ac.jp

**Abstract.** An extension of a modular termination result for term rewriting systems (TRSs, for short) by A. Middeldorp (1989) is presented. We intended to obtain this by adapting the dummy elimination transformation by M. C. F. Ferreira and H. Zantema (1995) under the presence of a non-collapsing non-duplicating terminating TRS whose function symbols are all to be eliminated. We propose a tree lifting ordering induced from a reduction order and a set $\mathcal{G}$ of function symbols, and use this ordering to transform a TRS $\mathcal{R}$ into $\mathcal{R}'$; termination of $\mathcal{R}'$ implies that of $\mathcal{R} \cup \mathcal{S}$ for any non-collapsing non-duplicating terminating TRS $\mathcal{S}$ whose function symbols are contained in $\mathcal{G}$, provided that for any $l \to r$ in $\mathcal{R}$ (1) the root symbol of $r$ is in $\mathcal{G}$ whenever that of $l$ is in $\mathcal{G}$; and (2) no variable appears directly below a symbol from $\mathcal{G}$ in $l$ when $\mathcal{G}$ contains a constant. Because of conditions (1) and (2), our technique covers only a part of the dummy elimination technique; however, even when $\mathcal{S}$ is empty, there are cases that our technique has an advantage over the dummy elimination technique.

## 1 Introduction

Modular aspects of properties of term rewriting systems (TRSs, for short), in particular those of termination of TRSs, have been investigated in many papers. In this paper, we present a technique that is useful to infer termination of the union of two TRSs provided that one of them does not contain collapsing rules nor duplicating rules. It is worth mentioning that our result also admits the hierarchical combination (i.e. a combination such that defined symbols in one TRS may occur in the other TRS), for which a few results have been obtained; see [2][3][8][10].

We propose a *tree lifting ordering induced from a reduction order and a set $\mathcal{G}$ of function symbols*, and use this ordering to transform a TRS $\mathcal{R}$ into $\hat{\mathcal{R}}$; termination of $\hat{\mathcal{R}}$ implies that of $\mathcal{R} \cup \mathcal{S}$ for any non-collapsing non-duplicating terminating TRS $\mathcal{S}$ whose function symbols are contained in $\mathcal{G}$, provided that for any $l \to r \in \mathcal{R}$ (1) $\mathrm{root}(l) \in \mathcal{G}$ implies $\mathrm{root}(r) \in \mathcal{G}$ and (2) no variable appears directly below a symbol from $\mathcal{G}$ in $l$ when $\mathcal{G}$ contains a constant. Here $\mathrm{root}(t)$ denotes the root symbol of a term $t$.

Our approach to infer termination of the union of two TRSs $\mathcal{R}$ and $\mathcal{S}$ is different from others in a point that we infer the termination of $\mathcal{R} \cup \mathcal{S}$ not directly

from that of $\mathcal{R}$ and $\mathcal{S}$, but instead we first transform $\mathcal{R}$ to $\hat{\mathcal{R}}$ (depending on the set of function symbols appearing in $\mathcal{S}$) and infer the termination of $\mathcal{R} \cup \mathcal{S}$ from that of $\hat{\mathcal{R}}$ and $\mathcal{S}$. In particular, when the sets of function symbols appearing in $\mathcal{R}$ and $\mathcal{S}$ are disjoint, we can take $\mathcal{R}$ as $\hat{\mathcal{R}}$ and hence our result covers a modular termination result [9] by A. Middeldorp.

We intended to obtain our result by adapting the dummy elimination transformation [6] by M. C. F. Ferreira and H. Zantema under the presence of a non-collapsing non-duplicating terminating TRS whose function symbols are all to be eliminated. Indeed, under conditions (1) and (2), the dummy elimination transformation is a particular case of the transformation from $\mathcal{R}$ to $\hat{\mathcal{R}}$ above. Although it is naturally expected that the dummy elimination transformation is sound without these conditions, but this is not true under the presence of a non-empty $\mathcal{S}$; see Example 2 in Section 3. Thus, to admit non-empty $\mathcal{S}$ we require the conditions also on $\mathcal{R}$, and consequently our technique covers only a part of the dummy elimination technique. However, even when $\mathcal{S}$ is empty, there are cases that our technique has an advantage over the dummy elimination technique; see Example 5 in Section 4.

The rest of this paper is organized as follows. In Section 2, we fix notations on term rewriting and orderings used in this paper. In Section 3, we introduce a tree lifting ordering and show its properties. Also, our main theorem (Theorem 13) is established at the end of Section 3. In Section 4, we discuss our result in the light of related works.

## 2 Preliminaries

Let $\mathcal{V}$ be a set of countably infinite variables (denoted by $x, y, z, \ldots$) and $\mathcal{F}$ a set of arity-fixed function symbols (denoted by $f, g, h, \ldots$). Then the set of *terms* (denoted by $s, t, u, \ldots$) built from $\mathcal{F}$ and $\mathcal{V}$ is written as $\mathcal{T}(\mathcal{F}, \mathcal{V})$, or as just $\mathcal{T}$ when $\mathcal{F}$ is obvious from the context. $\mathcal{V}(t)$ and $\mathrm{Var}(t)$ are the set and the multiset of variables occurring in $t$, respectively. We denote by $\equiv$ the syntactical equality.

We assume familiarity with the context formalism. A term $s$ is a *subterm* of a term $t$ (denoted by $s \trianglelefteq t$) if $t \equiv C[s]$ for some context $C$ with precisely one hole; it is a *proper* subterm (denoted by $s \triangleleft t$) if $s \not\equiv t$.

A *substitution* $\sigma$ on $\mathcal{T}$ is a mapping from $\mathcal{V}$ to $\mathcal{T}$ with finite $\mathrm{dom}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \not\equiv x\}$. A substitution $\sigma$ is extended to a homomorphism from $\mathcal{T}$ to $\mathcal{T}$. For a substitution $\sigma$ and a term $t$, we conventionally write $t\sigma$ instead of $\sigma(t)$. A substitution $\sigma$ with $\mathrm{dom}(\sigma) = \{x_1 \ldots, x_n\}$ and $\sigma(x_i) = u_i$ for $i = 1, \ldots, n$ is also denoted by $\{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$.

Given a binary relation $R$ on a set $W$, the reflexive transitive closure and transitive closure of $R$ are denoted by $R^*$ and $R^+$, respectively. A binary relation $R$ on $\mathcal{T}$ is said to be *closed under contexts* if $sRt$ implies $C[s]RC[t]$ for every terms $s, t$ and every context $C$; it is *closed under substitutions* if $sRt$ implies $(s\sigma)R(t\sigma)$ for every terms $s, t$ and every substitution $\sigma$. A *rewrite relation* is a binary relation on $\mathcal{T}$ closed under contexts and substitutions. A *strict partial order* $>$ on a set $W$ is a transitive irreflexive relation on $W$; it is *well-founded* if

there is no infinite sequence $a_1 > a_2 > \cdots$ of elements from $W$. A *rewrite order* is a rewrite relation that is a strict partial order. A rewrite order is a *reduction order* if it is well-founded. Note that, for any reduction order $>$, $s > t$ implies $\mathcal{V}(s) \supseteq \mathcal{V}(t)$.

A pair $\langle l, r \rangle$ of terms is called a *rewrite rule*. We conventionally write $l \to r$ instead of $\langle l, r \rangle$. A rewrite rule $l \to r$ is *collapsing* if $r \in \mathcal{V}$; it is *duplicating* if $\mathrm{Var}(l) \not\supseteq \mathrm{Var}(r)$. A *term rewriting system* is a set of rewrite rules[1]. A TRS is said to be *non-collapsing* (*non-duplicating*) if it contains no collapsing (duplicating, respectively) rules. Given a TRS $\mathcal{R}$, we denote by $\to_{\mathcal{R}}$ the smallest rewrite relation on $\mathcal{T}$ that includes $\mathcal{R}$. A TRS $\mathcal{R}$ is *compatible* with a rewrite order $>$ if $l > r$ for any $l \to r \in \mathcal{R}$. A TRS $\mathcal{R}$ is said to be *terminating* if it is compatible with a reduction order.

We denote by $\mathrm{Mult}(W)$ the set of finite multisets of elements taken from a set $W$. We use $\in$ for the membership relation and $[\,]$ for the empty multiset. The *multiset sum* and the *multiset minus* are denoted by $+$ and $\setminus$, respectively; e.g. $[a,b,b,c] + [b,c,c,d] = [a,b,b,b,c,c,c,d]$, and $[a,b,b,c] \setminus [b,c,c,d] = [a,b]$. For a natural number $n$ and a multiset $M$, $M * n$ stands for $\underbrace{M + \cdots + M}_{n\text{-times}}$. For $M \in \mathrm{Mult}(W)$ and $a \in W$, $\#(a, M)$ is the number of $a$ in $M$, and $|M|$ the number of elements in $M$; e.g. $\#(a, [a,b,a,b,a,c]) = 3$, and $|[a,b,a,b,a,c]| = 6$.

A *quasi-order* $\gtrsim$ on a set $W$ is a transitive reflexive relation on $W$. Given a quasi-order $\gtrsim$, the equivalence relation $\sim = (\gtrsim \cap \lesssim)$ is called the *equivalence relation of* $\gtrsim$, and the strict partial order $> = (\gtrsim \cap \not\lesssim)$ is called the *strict part of* $\gtrsim$. The quasi-order is well-founded if so is its strict part.

Given a quasi-order $\gtrsim$ on a set $W$, we denote by $\gtrsim^{mult}$ the *multiset quasi-ordering on* $\mathrm{Mult}(W)$ *induced from* $\gtrsim$; we denote by $\gtrsim^{mult}$ its strict part; see e.g. [11].

We denote by $\mathrm{Tr}(W)$ the set of finite (unordered) trees whose nodes are labeled with elements taken from a set $W$. Formally, $a\langle \Gamma \rangle \in \mathrm{Tr}(W)$ if $a \in W$ and $\Gamma \in \mathrm{Mult}(\mathrm{Tr}(W))$. We abbreviate $a\langle [T_1, \ldots, T_n] \rangle$ to $a\langle T_1, \ldots, T_n \rangle$ and $a\langle [\,] \rangle$ to $a$.

Given a strict partial order $>$ on a set $W$, we denote by $>_{rpo}$ the *recursive path ordering on* $\mathrm{Tr}(W)$ *induced from* $>$; see e.g. [11].

## 3 Tree lifting ordering

In this section, let us fix sets $\mathcal{F}$, $\mathcal{G}$ of function symbols such that $\mathcal{G} \subseteq \mathcal{F}$, a constant symbol $\Diamond$ such that $\Diamond \notin \mathcal{F}$, and a reduction order $>$ on $\mathcal{T}(\mathcal{F} \setminus \mathcal{G} \cup \{\Diamond\}, \mathcal{V})$.

**Definition 1.** 1. The *root* of a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is defined by

$$\mathrm{root}(t) = \begin{cases} t & \text{if } t \in \mathcal{V}, \\ f & \text{if } t \equiv f(t_1, \ldots, t_n). \end{cases}$$

[1] We omit the usual restrictions on a TRS $\mathcal{R}$. For, they are implied by termination of $\mathcal{R}$.

2. Let $t \equiv C[t_1, \ldots, t_n]$ $(n \geq 0)$ be a term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with $\mathrm{root}(t) \in \mathcal{G}$. We write $t \equiv C[\![t_1, \ldots, t_n]\!]$ if $C \in \mathcal{C}(\mathcal{G}, \emptyset)$ and $\mathrm{root}(t_1), \ldots, \mathrm{root}(t_n) \notin \mathcal{G}$.
3. Let $t \equiv C[t_1, \ldots, t_n]$ $(n \geq 0)$ be a term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $C \in \mathcal{C}(\mathcal{F} \backslash \mathcal{G}, \mathcal{V})$ and $\mathrm{root}(t_1), \ldots, \mathrm{root}(t_n) \in \mathcal{G}$. Suppose that $t_i \equiv C_i[\![s_{i,1}, \ldots, s_{i,m_i}]\!]$ for $i = 1, \ldots, n$. We denote by $\mathrm{suc}(t)$ the multiset $[s_{1,1}, \ldots, s_{n,m_n}]$, and by $\mathrm{cap}(t)$ the term $C[\Diamond, \ldots, \Diamond]$.
4. The *rank* of a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is defined by

$$\mathrm{rank}(t) = \begin{cases} 1 & \text{if } \mathrm{suc}(t) = [\,], \\ 1 + \max\{\mathrm{rank}(t_i) \mid t_i \in \mathrm{suc}(t)\} & \text{otherwise.} \end{cases}$$

5. For any term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, put

$$\mathrm{tree}(t) = \begin{cases} \mathrm{cap}(t) & \text{if } \mathrm{suc}(t) = [\,], \\ \mathrm{cap}(t)\langle \mathrm{tree}(t_1), \ldots, \mathrm{tree}(t_n) \rangle & \\ \qquad \text{if } \mathrm{suc}(t) = [t_1, \ldots, t_n] \text{ with } n \geq 1, & \end{cases}$$

$$\mathrm{nodes}(t) = \begin{cases} [\mathrm{cap}(t)] & \text{if } \mathrm{suc}(t) = [\,], \\ [\mathrm{cap}(t)] + \sum_{1 \leq i \leq n} \mathrm{nodes}(t_i) & \\ \qquad \text{if } \mathrm{suc}(t) = [t_1, \ldots, t_n] \text{ with } n \geq 1. & \end{cases}$$

**Definition 2.** Let $\succ = (> \cup \triangleright)^+$. We define binary relations $\sim_{tlo(\mathcal{G})}$ and $>_{tlo'(\mathcal{G})}$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ by

$$s \sim_{tlo(\mathcal{G})} t \text{ if and only if } \mathrm{tree}(s) = \mathrm{tree}(t);$$
$$s >_{tlo'(\mathcal{G})} t \text{ if and only if } \mathrm{tree}(s) \succ_{rpo} \mathrm{tree}(t).$$

Clearly, $\sim_{tlo(\mathcal{G})}$ is an equivalence relation. Also, since $\succ$ is a well-founded strict partial order, so is $>_{tlo'(\mathcal{G})}$.

We put $\gtrsim_{tlo'(\mathcal{G})} = (>_{tlo'(\mathcal{G})} \cup \sim_{tlo(\mathcal{G})})$. It follows from $s \sim_{tlo(\mathcal{G})} s' >_{tlo'(\mathcal{G})} t' \sim_{tlo(\mathcal{G})} t$ implies $s >_{tlo'(\mathcal{G})} t$ that $\gtrsim_{tlo'(\mathcal{G})}$ is a quasi-order with $\sim_{tlo(\mathcal{G})}$ as its equivalence relation and $>_{tlo'(\mathcal{G})}$ as its strict part. Thus, alternatively, we can define $>_{tlo'(\mathcal{G})}$ as follows: $s >_{tlo'(\mathcal{G})} t$ if and only if either

**TLO':1** $\mathrm{cap}(s)(> \cup \triangleright)^+ \mathrm{cap}(t)$, and $s >_{tlo'(\mathcal{G})} v$ for any $v \in \mathrm{suc}(t)$,
**TLO':2** $u \gtrsim_{tlo'(\mathcal{G})} t$ for some $u \in \mathrm{suc}(s)$, or
**TLO':3** $\mathrm{cap}(s) \equiv \mathrm{cap}(t)$, and $\mathrm{suc}(s) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$.

Here, $\gtrsim^{mult}_{tlo'(\mathcal{G})}$ is the strict part of multiset quasi-ordering induced from $\gtrsim_{tlo'(\mathcal{G})}$. We will mainly use this definition in the proofs hereafter.

The following lemma is readily checked.

**Lemma 3.** *1.* $s >_{tlo'(\mathcal{G})} u$ *for any* $u \in \mathrm{suc}(s)$*;*
*2.* $s >_{tlo'(\mathcal{G})} a$ *for any* $a \in \mathrm{nodes}(s)$ *provided that* $|\mathrm{nodes}(s)| > 1$.

At first sight, it may seem strange to include the (proper) subterm relation " $\triangleright$ " in Definition 2 (and in **TLO':1**). But without " $\triangleright$ ", we can not generally guarantee that the tree lifting ordering is closed under contexts; see Figure 1.

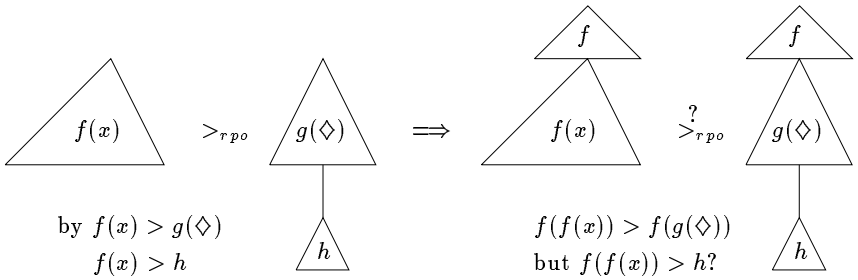


**Fig. 1.** Why is "$\triangleright$" required?

**Definition 4.** The *tree lifting ordering* $>_{tlo(\mathcal{G})}$ *on* $\mathcal{T}(\mathcal{F},\mathcal{V})$ *induced from* $>$ *and* $\mathcal{G}$ is defined as follows: $s >_{tlo(\mathcal{G})} t$ if and only if either

**TLO:1** $\mathrm{cap}(s) > \mathrm{cap}(t)$, and $s >_{tlo'(\mathcal{G})} v$ for any $v \in \mathrm{suc}(t)$, or
**TLO:2** $\mathrm{cap}(s) \equiv \mathrm{cap}(t)$, and $\mathrm{suc}(s) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$.

It is clear that $>_{tlo(\mathcal{G})} \subseteq >_{tlo'(\mathcal{G})}$.

We are going to give here an explanation why $>_{tlo'(\mathcal{G})}$ itself is not suitable for our purpose and have introduced an alternative relation $>_{tlo(\mathcal{G})}$. The first difference is that **TLO':2** is omitted in $>_{tlo(\mathcal{G})}$ at the first stage of comparison. For, otherwise the tree lifting ordering can not be closed under contexts; see Figure 2. We also can not use "$\triangleright$" at the first stage of comparison to make sure that the tree lifting ordering is closed under contexts; see Figure 3. Accordingly, we omit "$\triangleright$" in **TLO:1**.

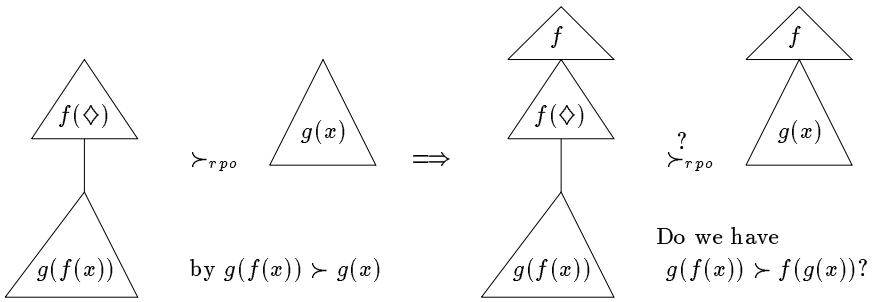


**Fig. 2.** Why is "$>_{tlo'(\mathcal{G})}$" not suitable? (1)

**Lemma 5.** *The relation* $>_{tlo(\mathcal{G})}$ *is transitive.*

*Proof.* The claim can be proved using the fact $>_{tlo(\mathcal{G})} \subseteq >_{tlo'(\mathcal{G})}$. □

**Lemma 6.** *The relation* $>_{tlo(\mathcal{G})}$ *is a well-founded strict partial order.*

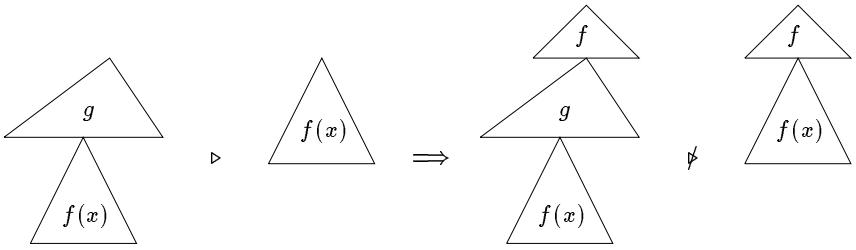


**Fig. 3.** Why is "$>_{tlo'(\mathcal{G})}$" not suitable? (2)

*Proof.* By Lemma 5, it suffices to show $>_{tlo(\mathcal{G})}$ is well-founded (because irreflexivity follows from well-foundedness). But this follows from $>_{tlo(\mathcal{G})} \subseteq >_{tlo'(\mathcal{G})}$ since $>_{tlo'(\mathcal{G})}$ is well-founded. □

Henceforth, $\sim_{tlo(\mathcal{G})} \cup >_{tlo(\mathcal{G})}$ is denoted by $\gtrsim_{tlo(\mathcal{G})}$. It is readily checked that $\gtrsim_{tlo(\mathcal{G})}$ is a quasi-order with $\sim_{tlo(\mathcal{G})}$ as its equivalence relation and $>_{tlo(\mathcal{G})}$ as its strict part.

**Lemma 7.** *For any substitution $\sigma$, if $s \sim_{tlo(\mathcal{G})} t$ then $s\sigma \sim_{tlo(\mathcal{G})} t\sigma$.*

*Proof.* The claim can be proved from the definition of $\sim_{tlo(\mathcal{G})}$ and the fact that $\mathrm{tree}(u) = x$ iff $u \equiv x$. □

**Lemma 8.** *Let $\theta$ be a substitution such that $\theta(x) \in \mathcal{T}(\mathcal{F}\backslash\mathcal{G}, \mathcal{V})$ for any $x \in \mathrm{dom}(\theta)$. Then $s >_{tlo'(\mathcal{G})} t$ implies $s\theta >_{tlo'(\mathcal{G})} t\theta$.*

*Proof.* One easily shows that $u(> \cup \triangleright)^+ v$ implies $u\theta(> \cup \triangleright)^+ v\theta$ for any $u, v \in \mathcal{T}(\mathcal{F}\backslash\mathcal{G} \cup \{\Diamond\}, \mathcal{V})$. Also, note that $\mathrm{cap}(w\theta) \equiv \mathrm{cap}(w)\theta$ and $\mathrm{suc}(w\theta) = [u\theta \mid u \in \mathrm{suc}(w)]$ for any term $w \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. Now the statement can be proved by induction on $\mathrm{rank}(s) + \mathrm{rank}(t)$. □

**Lemma 9.** *Suppose $s >_{tlo'(\mathcal{G})} t$, and let $\sigma$ be a substitution with $\mathrm{dom}(\sigma) = \{x\}$. Suppose $x \notin \mathrm{nodes}(s)$ when $\mathcal{G}$ contains a constant. Then $s\sigma \gtrsim_{tlo'(\mathcal{G})} t\sigma$; furthermore, $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ if either $\mathrm{root}(\sigma(x)) \notin \mathcal{G}$, $\mathrm{root}(s) \notin \mathcal{G}$ or $t \not\equiv x$.*

*Proof.* We only consider here the case $\sigma(x) \notin \mathcal{V}$. Also, we can assume $\mathrm{root}(s) \notin \mathcal{V}$ (because otherwise there is no such $t$). We distinguish two cases:

1. $\mathrm{root}(\sigma(x)) \in \mathcal{F}\backslash\mathcal{G}$. Then we have

$$\mathrm{cap}(w\sigma) \equiv \mathrm{cap}(w)\{x \mapsto \mathrm{cap}(\sigma(x))\} \tag{1}$$
$$\mathrm{suc}(w\sigma) = [u\sigma \mid u \in \mathrm{suc}(w)] + \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(w))) \tag{2}$$

for any term $w \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.
We show that $s >_{tlo'(\mathcal{G})} t$ implies $s\sigma >_{tlo'(\mathcal{G})} t\sigma$, by induction on $\mathrm{rank}(s) + \mathrm{rank}(t)$. Suppose that $s >_{tlo'(\mathcal{G})} t$.

For the base step, suppose that $\mathrm{rank}(s) = \mathrm{rank}(t) = 1$. Then $s >_{tlo'(\mathcal{G})} t$ is due to **TLO':1**, and hence we have $\mathrm{cap}(s)(> \cup \triangleright)^+ \mathrm{cap}(t)$. From the assumption that $>$ is a reduction order, one easily shows that $\mathrm{cap}(s)\theta(> \cup \triangleright)^+ \mathrm{cap}(t)\theta$ for any $\theta$ with $\theta(x) \in \mathcal{T}(\mathcal{F}\backslash\mathcal{G} \cup \{\Diamond\}, \mathcal{V})$ for any $x \in \mathrm{dom}(\theta)$. Thus, using equation (2), we get $\mathrm{cap}(s\sigma) \equiv \mathrm{cap}(s)\{x \mapsto \mathrm{cap}(\sigma(x))\}(> \cup \triangleright)^+ \mathrm{cap}(t)\{x \mapsto \mathrm{cap}(\sigma(x))\} \equiv \mathrm{cap}(t\sigma)$. Next, suppose $v \in \mathrm{suc}(t\sigma)$. Then, since $\mathrm{suc}(t) = [\,]$, we have $v \in \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(t)))$ by equation (2) i.e. $v \in \mathrm{suc}(\sigma(x))$ and $x \in \mathcal{V}(\mathrm{cap}(t))$. Since $>$ is a reduction order, we have $\mathcal{V}(\mathrm{cap}(t)) \subseteq \mathcal{V}(\mathrm{cap}(s))$, and so $x \in \mathcal{V}(\mathrm{cap}(s))$. Then, again using equation (2), we obtain $v \in \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(s))) = \mathrm{suc}(s\sigma)$, and so $s\sigma >_{tlo'(\mathcal{G})} v$ by Lemma 3. Thus we have shown $s\sigma >_{tlo'(\mathcal{G})} v$ for any $v \in \mathrm{suc}(t\sigma)$. Hence by **TLO':1** we conclude that $s\sigma >_{tlo'(\mathcal{G})} t\sigma$.

For the induction step, we distinguish three cases:

(a) $s >_{tlo'(\mathcal{G})} t$ by **TLO':1**. Then $\mathrm{cap}(s)(> \cup \triangleright)^+ \mathrm{cap}(t)$ and $s >_{tlo'(\mathcal{G})} u$ for any $u \in \mathrm{suc}(t)$. We have $\mathrm{cap}(s\sigma)(> \cup \triangleright)^+ \mathrm{cap}(t\sigma)$ using equation (2). Let $v \in \mathrm{suc}(t\sigma)$. To show $s\sigma >_{tlo'(\mathcal{G})} v$, we distinguish two cases according to equation (2):

i. $v \equiv u\sigma$ for some $u \in \mathrm{suc}(t)$. Since $\mathrm{root}(\sigma(x)) \notin \mathcal{G}$, we can apply the induction hypothesis to $s >_{tlo'(\mathcal{G})} u$; thus, we obtain $s\sigma >_{tlo'(\mathcal{G})} u\sigma \equiv v$.

ii. $v \in \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(t)))$. Then $x \in \mathcal{V}(\mathrm{cap}(s))$, and so $v \in \mathrm{suc}(s\sigma)$. Hence $s\sigma >_{tlo'(\mathcal{G})} v$ by Lemma 3.

Thus, $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':1**.

(b) $s >_{tlo'(\mathcal{G})} t$ by **TLO':2**. Then $u \gtrsim_{tlo'(\mathcal{G})} t$ for some $u \in \mathrm{suc}(s)$. Then, by induction hypothesis and Lemma 7, $u\sigma \gtrsim_{tlo'(\mathcal{G})} t\sigma$. Since $\mathrm{root}(\sigma(x)) \in \mathcal{F}\backslash\mathcal{G}$, we have $u\sigma \in \mathrm{suc}(s\sigma)$, and so $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':2**.

(c) $s >_{tlo'(\mathcal{G})} t$ by **TLO':3**. Then $\mathrm{cap}(s) \equiv \mathrm{cap}(t)$ and $\mathrm{suc}(s) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$. We have $\mathrm{cap}(s\sigma) \equiv \mathrm{cap}(s)\{x \mapsto \mathrm{cap}(\sigma(x))\} \equiv \mathrm{cap}(t)\{x \mapsto \mathrm{cap}(\sigma(x))\} \equiv \mathrm{cap}(t\sigma)$. Also, by induction hypothesis and Lemma 7, we get $[u\sigma \mid u \in \mathrm{suc}(s)] \gtrsim^{mult}_{tlo'(\mathcal{G})} [v\sigma \mid v \in \mathrm{suc}(t)]$. Thus,

$$\begin{aligned}\mathrm{suc}(s\sigma) &= [u\sigma \mid u \in \mathrm{suc}(s)] + \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(s))) \\ &= [u\sigma \mid u \in \mathrm{suc}(s)] + \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(t))) \\ &\gtrsim^{mult}_{tlo'(\mathcal{G})} [v\sigma \mid v \in \mathrm{suc}(t)] + \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(t))) \\ &= \mathrm{suc}(t\sigma).\end{aligned}$$

Hence, $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':3**.

2. $\mathrm{root}(\sigma(x)) \in \mathcal{G}$. Then we have

$$\mathrm{cap}(w\sigma) \equiv \mathrm{cap}(w)\{x \mapsto \Diamond\} \tag{3}$$

$$\mathrm{suc}(w\sigma) = [u\sigma \mid u \in \mathrm{suc}(w), u \not\equiv x] + \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(w)) + \mathrm{suc}(w)). \tag{4}$$

for any term $w \in \mathcal{T}(\mathcal{F}, \mathcal{V})$.

For the base step, suppose that $\mathrm{rank}(s) = \mathrm{rank}(t) = 1$. Then $s >_{tlo'(\mathcal{G})} t$ is due to **TLO':1**, and hence we have $\mathrm{cap}(s)(> \cup \triangleright)^+ \mathrm{cap}(t)$. From this follows

$\mathrm{cap}(s\sigma)(> \cup \rhd)^+ \mathrm{cap}(t\sigma)$ using equation (4). Furthermore, since $\mathrm{suc}(t) = [\,]$ and $\mathcal{V}(\mathrm{cap}(t)) \subseteq \mathcal{V}(\mathrm{cap}(s))$, $v \in \mathrm{suc}(t\sigma)$ implies $v \in \mathrm{suc}(s\sigma)$ by equation (4). Hence, $s\sigma >_{tlo'(\mathcal{G})} v$ for any $v \in \mathrm{suc}(t\sigma)$ by Lemma 3. Thus, $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':1**.

For the induction step, we distinguish three cases:

(a) $s >_{tlo'(\mathcal{G})} t$ by **TLO':1**. Then $\mathrm{cap}(s)(> \cup \rhd)^+ \mathrm{cap}(t)$ and $s >_{tlo'(\mathcal{G})} u$ for any $u \in \mathrm{suc}(t)$. We have $\mathrm{cap}(s\sigma)(> \cup \rhd)^+ \mathrm{cap}(t\sigma)$ using equation (4). Let $v \in \mathrm{suc}(t\sigma)$. To show $s\sigma >_{tlo'(\mathcal{G})} v$, we distinguish three cases according to equation (4):

i. $v \equiv u\sigma$ for some $u \in \mathrm{suc}(t)$ with $u \not\equiv x$. Since $u \not\equiv x$, we can apply the induction hypothesis to $s >_{tlo'(\mathcal{G})} u$; thus, we obtain $s\sigma >_{tlo'(\mathcal{G})} u\sigma \equiv v$.

ii. $v \in \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(t)))$. Then $x \in \mathcal{V}(\mathrm{cap}(t))$, and so $x \in \mathcal{V}(\mathrm{cap}(s))$ by $\mathrm{cap}(s)(> \cup \rhd)^+ \mathrm{cap}(t)$, since $(> \cup \rhd)^+$ is well-founded and closed under substitutions. Thus, $v \in \mathrm{suc}(s\sigma)$ by equation (4). Hence $s\sigma >_{tlo'(\mathcal{G})} v$ by Lemma 3.

iii. $v \in \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{suc}(t))$. Since $x \in \mathrm{nodes}(t)$, there exists $u \in \mathrm{nodes}(s)$ such that $x \in \mathcal{V}(u)$. Thus $v \in \mathrm{nodes}(s\sigma)$ and $|\,\mathrm{nodes}(s\sigma)| > 1$, and hence $s\sigma >_{tlo'(\mathcal{G})} v$ by Lemma 3.

Thus, $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':1**.

(b) $s >_{tlo'(\mathcal{G})} t$ by **TLO':2**. Then $v \gtrsim_{tlo'(\mathcal{G})} t$ for some $v \in \mathrm{suc}(s)$. By Lemma 7 and induction hypothesis, we get $v\sigma \gtrsim_{tlo'(\mathcal{G})} t\sigma$. If $v \not\equiv x$, then $v\sigma \in \mathrm{suc}(s\sigma)$, and so $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':2**. Suppose otherwise i.e. $v \equiv x$. Then, since $v \gtrsim_{tlo'(\mathcal{G})} t$, we have $t \equiv x$. Also, since $\mathrm{suc}(s) \neq [\,]$, $\mathrm{cap}(s) \trianglerighteq \diamondsuit$.

i. $\mathrm{root}(s) \in \mathcal{F}\backslash\mathcal{G}$. Then $\mathrm{cap}(s\sigma) \not\equiv \diamondsuit$, and hence we have $\mathrm{cap}(s\sigma) \rhd \diamondsuit$. By $\mathrm{suc}(\sigma(x)) = \mathrm{suc}(t\sigma) \subseteq \mathrm{suc}(s\sigma)$, we have $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':1** and Lemma 3.

ii. $\mathrm{root}(s) \in \mathcal{G}$. Then $\mathrm{cap}(s\sigma) \equiv \diamondsuit \equiv \mathrm{cap}(t\sigma)$. By $\mathrm{suc}(\sigma(x)) = \mathrm{suc}(t\sigma) \subseteq \mathrm{suc}(s\sigma)$, we have $s\sigma \gtrsim_{tlo'(\mathcal{G})} t\sigma$. (Note that in this case it is not necessary to have $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ because the condition of the lemma is not satisfied.)

(c) $s >_{tlo'(\mathcal{G})} t$ by **TLO':3**. Then $\mathrm{cap}(s) \equiv \mathrm{cap}(t)$ and $\mathrm{suc}(s) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$. From the former, we obtain $\mathrm{cap}(s\sigma) \equiv \mathrm{cap}(t\sigma)$ using equation (4). We are going to show $\mathrm{suc}(s\sigma) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t\sigma)$. Since $\mathrm{cap}(s) \equiv \mathrm{cap}(t)$, we have

$$\mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(s))) = \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{Var}(\mathrm{cap}(t))).$$

Thus, it remains to show

$$[v\sigma \mid v \in \mathrm{suc}(s), v \not\equiv x] + \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{suc}(s)) \gtrsim^{mult}_{tlo'(\mathcal{G})} [v\sigma \mid v \in \mathrm{suc}(t), v \not\equiv x] + \mathrm{suc}(\sigma(x)) * \#(x, \mathrm{suc}(t)).$$

Let $\|\,\mathrm{suc}(w)\|$ be $[\mathrm{tree}(u) \mid u \in \mathrm{suc}(w)]$ for any $w \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. When $\|\,\mathrm{suc}(s)\| \not\trianglerighteq \|\,\mathrm{suc}(t)\|$, it suffices to show that for any $\mathrm{tree}(w) \in \|\,\mathrm{suc}(s)\|\backslash$

$\|\,\mathrm{suc}(t)\|$ and $\mathrm{tree}(v) \in \|\,\mathrm{suc}(t)\| \setminus \|\,\mathrm{suc}(s)\|$ such that $w >_{tlo'(\mathcal{G})} v$, (i) $w\sigma >_{tlo'(\mathcal{G})} v\sigma$ when $x \not\equiv v$, and (ii) $[w\sigma] \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(v\sigma)$ when $x \equiv v$.

i. $x \not\equiv v$. Since $x \not\equiv v$, it follows from the induction hypothesis.

ii. $x \equiv v$. Since $\mathrm{root}(w) \notin \mathcal{G}$, $w\sigma >_{tlo'(\mathcal{G})} v\sigma$ by induction hypothesis; hence $[w\sigma] \gtrsim^{mult}_{tlo'(\mathcal{G})} [v\sigma] \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(v\sigma)$ using Lemma 3.

Otherwise, since $\mathrm{suc}(s) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$, we have $\|\,\mathrm{suc}(s)\| \supsetneq \|\,\mathrm{suc}(t)\|$. Then, since $\mathrm{cap}(s\sigma) \equiv \mathrm{cap}(t\sigma)$, we clearly have $\|\,\mathrm{suc}(s\sigma)\| \supseteq \|\,\mathrm{suc}(t\sigma)\|$; hence, it remains to show $\|\,\mathrm{suc}(s\sigma)\| \setminus \|\,\mathrm{suc}(t\sigma)\| \neq [\,]$. If there exists $\mathrm{tree}(v) \in \|\,\mathrm{suc}(s)\| \setminus \|\,\mathrm{suc}(t)\|$ with $v \not\equiv x$, then $\mathrm{tree}(v\sigma) \in \|\,\mathrm{suc}(s\sigma)\| \setminus \|\,\mathrm{suc}(t\sigma)\|$; otherwise, i.e. when $v \equiv x$ for any $\mathrm{tree}(v) \in \|\,\mathrm{suc}(s)\| \setminus \|\,\mathrm{suc}(t)\|$, we need to show $\mathrm{suc}(\sigma(x)) \neq [\,]$. But if $x \notin \mathrm{nodes}(s)$ then there is no such case; and if $\mathcal{G}$ contains no constant then $\sigma(x)$ contains at least one non-$\mathcal{G}$-symbol, and so $\mathrm{suc}(\sigma(x)) \neq [\,]$ holds. Thus, we conclude $\|\,\mathrm{suc}(s\sigma)\| \supsetneq \|\,\mathrm{suc}(t\sigma)\|$. Hence, $s\sigma >_{tlo'(\mathcal{G})} t\sigma$ by **TLO':3**.

□

*Remark.* The conditions in Lemma 9 can not be dropped: Suppose $\mathcal{G} = \{a, b\}$. Then $a(x) >_{tlo'(\mathcal{G})} x$ and $a(a(y)) \sim_{tlo(\mathcal{G})} a(y)$. If $\mathcal{G}$ additionally contains a constant $c$ then $f(a(x, y)) >_{tlo'(\mathcal{G})} f(b(y))$ and $f(a(c, y)) \sim_{tlo(\mathcal{G})} f(b(y))$; also note that $x \in \mathrm{nodes}(f(a(x, y)))$.

**Lemma 10.** *Let $\sigma$ be a substitution, and suppose that* $\mathrm{nodes}(s)$ *contains no variable when $\mathcal{G}$ contains a constant. Then, $s >_{tlo(\mathcal{G})} t$ implies $s\sigma >_{tlo(\mathcal{G})} t\sigma$.*

*Proof.* Similar to the proof of Lemma 9. (Replace "induction hypothesis" by "Lemma 9" and omit the cases 1(b) and 2(b).) □

**Lemma 11.** *For any $C \in \mathcal{C}(\mathcal{F}\backslash\mathcal{G}, \mathcal{V})$, if $s >_{tlo'(\mathcal{G})} t$ then $C[s] >_{tlo'(\mathcal{G})} t$.*

*Proof.* Let $C \in \mathcal{C}(\mathcal{F}\backslash\mathcal{G}, \mathcal{V})$ and suppose $s >_{tlo'(\mathcal{G})} t$. The case where $C \equiv \Box$ is trivial; so we assume $C \not\equiv \Box$. Then, $\mathrm{cap}(C[s]) \equiv C[\mathrm{cap}(s)](> \cup \triangleright)^+ \mathrm{cap}(s)$. Also, since $\mathrm{suc}(C[s]) = \mathrm{suc}(s)$, $C[s] >_{tlo'(\mathcal{G})} v$ for any $v \in \mathrm{suc}(s)$ by Lemma 3. Thus, by **TLO':1** and our assumption, we get $C[s] >_{tlo'(\mathcal{G})} s >_{tlo'(\mathcal{G})} t$. □

**Lemma 12.** *If* $\mathrm{root}(s) \in \mathcal{G}$ *implies* $\mathrm{root}(t) \in \mathcal{G}$*, then, for any $C \in \mathcal{C}(\mathcal{F}, \mathcal{V})$, $s >_{tlo(\mathcal{G})} t$ implies $C[s] >_{tlo(\mathcal{G})} C[t]$.*

*Proof.* Let $w_1, \ldots, w_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. We prove this by showing that $s >_{tlo(\mathcal{G})} t$ implies $e(w_1, \ldots, s, \ldots, w_n) >_{tlo(\mathcal{G})} e(w_1, \ldots, t, \ldots, w_n)$ for any $e \in \mathcal{F}$. Then by induction on the length of $C$, the statement of the lemma follows. Suppose $s >_{tlo(\mathcal{G})} t$, and let $\tilde{s} \equiv e(w_1, \ldots, s, \ldots, w_n)$ and $\tilde{t} \equiv e(w_1, \ldots, t, \ldots, w_n)$.

1. $e = f \in \mathcal{F}\backslash\mathcal{G}$. Then we have

$$\mathrm{cap}(f(w_1, \ldots, u, \ldots, w_n)) \equiv f(\mathrm{cap}(w_1), \ldots, \mathrm{cap}(u), \ldots, \mathrm{cap}(w_n)) \tag{5}$$

$$\mathrm{suc}(f(w_1, \ldots, u, \ldots, w_n)) = \mathrm{suc}(u) + \sum_{1 \le i \le n} \mathrm{suc}(w_i) \tag{6}$$

for any $u \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. We distinguish two cases:

(a) $s >_{tlo(\mathcal{G})} t$ by **TLO:1**. Then $\mathrm{cap}(s) > \mathrm{cap}(t)$ and $s >_{tlo'(\mathcal{G})} v$ for any $v \in \mathrm{suc}(t)$. Since $>$ is a reduction order, we have $\mathrm{cap}(\tilde{s}) > \mathrm{cap}(\tilde{t})$ using equation (6). Suppose $v \in \mathrm{suc}(\tilde{t})$.

i. $v \in \mathrm{suc}(t)$. Then we have $s >_{tlo'(\mathcal{G})} v$, and so, by Lemma 11, $f(z_1, \ldots, s, \ldots, z_n) >_{tlo'(\mathcal{G})} v$ with new variables $z_1, \ldots, z_n$. Let $\sigma_i = \{z_i \mapsto w_i\}$ $(i = 1, \ldots, n)$. Since $f \notin \mathcal{G}$ and $z_i \not\equiv v$ for all $i$, we can apply Lemma 9 to obtain $(\cdots(f(z_1, \ldots, s, \ldots, z_n)\sigma_1)\cdots)\sigma_n >_{tlo'(\mathcal{G})} (\cdots(v\sigma_1)\cdots)\sigma_n$. Thus, $\tilde{s} \equiv f(w_1, \ldots, s, \ldots, w_n) >_{tlo'(\mathcal{G})} v$.

ii. $v \in \mathrm{suc}(w_i)$ for some $i$. Then $v \in \mathrm{suc}(\tilde{s})$ by equation (6), and so $\tilde{s} >_{tlo'(\mathcal{G})} v$ by Lemma 3.

(b) $s >_{tlo(\mathcal{G})} t$ by **TLO:2**. Then $\mathrm{cap}(s) \equiv \mathrm{cap}(t)$ and $\mathrm{suc}(s) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$. From the former $\mathrm{cap}(\tilde{s}) \equiv \mathrm{cap}(\tilde{t})$ follows using equation (6), and from the latter $\mathrm{suc}(\tilde{s}) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(\tilde{t})$ using equation (6). Thus $\tilde{s} >_{tlo(\mathcal{G})} \tilde{t}$ by **TLO:2**.

2. $e = a \in \mathcal{G}$. Then we have

$$\mathrm{cap}(a(w_1, \ldots, u, \ldots, w_n)) \equiv \Diamond \tag{7}$$

$$\mathrm{suc}(a(w_1, \ldots, u, \ldots, w_n)) = [\delta \in [w_1, \ldots, u, \ldots, w_n] \mid \mathrm{root}(\delta) \notin \mathcal{G}] + \sum [\mathrm{suc}(\delta) \mid \delta \in [w_1, \ldots, u, \ldots, w_n] \text{ and } \mathrm{root}(\delta) \in \mathcal{G}] \tag{8}$$

for any $u \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. By equation (8), it remains to show $\mathrm{suc}(\tilde{s}) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(\tilde{t})$. By the condition, it suffices to distinguish three cases:

(a) $\mathrm{root}(s), \mathrm{root}(t) \notin \mathcal{G}$. Since $s >_{tlo(\mathcal{G})} t$, we have $s >_{tlo'(\mathcal{G})} t$, and so $\mathrm{suc}(\tilde{s}) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(\tilde{t})$ by equation (8).

(b) $\mathrm{root}(s), \mathrm{root}(t) \in \mathcal{G}$. Then $s >_{tlo(\mathcal{G})} t$ is due to **TLO:2**, and so we have $\mathrm{suc}(s) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$. Hence $\mathrm{suc}(\tilde{s}) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(\tilde{t})$ by equation (8).

(c) $\mathrm{root}(s) \notin \mathcal{G}$ and $\mathrm{root}(t) \in \mathcal{G}$. Since $s >_{tlo(\mathcal{G})} t$, we have $s >_{tlo'(\mathcal{G})} t$, and so $[s] \gtrsim^{mult}_{tlo'(\mathcal{G})} [t]$. Also, by Lemma 3, $t >_{tlo'(\mathcal{G})} u$ for any $u \in \mathrm{suc}(t)$, and so $[t] \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$. Thus, we obtain $[s] \gtrsim^{mult}_{tlo'(\mathcal{G})} [t] \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(t)$, and hence $\mathrm{suc}(\tilde{s}) \gtrsim^{mult}_{tlo'(\mathcal{G})} \mathrm{suc}(\tilde{t})$ by equation (8).

□

*Remark.* The condition in Lemma 12 can not be dropped: Let $\mathcal{G} = \{a\}$ and suppose $\Diamond > 1 > 0$. Then $a(0,0) >_{tlo(\mathcal{G})} 1$ and $a(1,0) >_{tlo(\mathcal{G})} a(a(0,0),0)$.

**Theorem 13.** *Let $\mathcal{G}$ be a set of function symbols and $\mathcal{R}$ a TRS such that*

1. $\mathrm{root}(l) \in \mathcal{G}$ *implies* $\mathrm{root}(r) \in \mathcal{G}$ *and*
2. *no variable appears directly below a symbol from $\mathcal{G}$ in $l$ when $\mathcal{G}$ contains a constant*

*for any $l \to r \in \mathcal{R}$. If there exists a reduction order $>$ on $\mathcal{T}(\mathcal{F}\backslash\mathcal{G} \cup \{\Diamond\}, \mathcal{V})$ such that $l >_{tlo(\mathcal{G})} r$ for any $l \to r \in \mathcal{R}$, then $\mathcal{S} \cup \mathcal{R}$ is terminating for any non-collapsing non-duplicating terminating TRS $\mathcal{S}$ whose function symbols are contained in $\mathcal{G}$.*

*Proof.* We show that $s \to_{\mathcal{S}} t$ implies $s \gtrsim_{tlo(\mathcal{G})} t$. Suppose $s \to_{\mathcal{S}} t$, then $s \equiv C[l\sigma]$ and $t \equiv C[r\sigma]$ for some $l \to r \in \mathcal{S}$. Since $\mathcal{S}$ is non-collapsing and non-duplicating, $l \to r$ must be of the form $C_l[x_1, \ldots, x_n] \to C_r[x_{i_1}, \ldots, x_{i_m}]$ with some subsequence $i_1, \ldots, i_m$ $(m \leq n)$ of a permutation of $1, \ldots, n$ and non-empty contexts $C_l, C_r$. Thus, if $m = n$ then $\mathrm{tree}(s) = \mathrm{tree}(t)$; and if $m < n$ then $s >_{tlo(\mathcal{G})} t$, since $\mathrm{tree}(s) \succ_{rpo} \mathrm{tree}(t)$ and $\mathrm{cap}(s) \equiv \mathrm{cap}(t)$.

Suppose there exists an infinite reduction sequence of $\mathcal{S} \cup \mathcal{R}$. Since $\mathcal{S}$ is terminating, the sequence has the form $t_0 \overset{*}{\to}_{\mathcal{S}} s_0 \to_{\mathcal{R}} t_1 \overset{*}{\to}_{\mathcal{S}} s_1 \to_{\mathcal{R}} t_2 \overset{*}{\to}_{\mathcal{S}} s_2 \to_{\mathcal{R}} \cdots$. Then we have $t_0 \gtrsim_{tlo(\mathcal{G})} s_0 >_{tlo(\mathcal{G})} t_1 \gtrsim_{tlo(\mathcal{G})} s_1 >_{tlo(\mathcal{G})} t_2 \gtrsim_{tlo(\mathcal{G})} s_2 >_{tlo(\mathcal{G})} \cdots$ by Lemma 10, Lemma 12 and our assumptions; thus $t_0 >_{tlo(\mathcal{G})} t_1 >_{tlo(\mathcal{G})} t_2 >_{tlo(\mathcal{G})} \cdots$. This contradicts Lemma 6 that states $>_{tlo(\mathcal{G})}$ is well-founded. □

The following example addresses how to apply our result.

*Example 1.* Let

$$\begin{cases} f(f(x)) \to f(a(b(f(x)))) & (1) \\ f(a(g(x))) \to b(x) & (2) \\ b(x) \to a(x) & (3) \end{cases}$$

be a TRS. We show this TRS is terminating via our technique. For this, let $\mathcal{G} = \{a, b\}$. Since (3) is a non-duplicating non-collapsing rule consisting of $\mathcal{G}$-symbols, we let $\mathcal{R} = \{(1), (2)\}$, $\mathcal{S} = \{(3)\}$. Now, what we want is a reduction order $>$ satisfying

$$\begin{aligned} &f(f(x)) >_{tlo(\mathcal{G})} f(a(b(f(x)))) && (4) \\ &f(a(g(x))) >_{tlo(\mathcal{G})} b(x) && (5). \end{aligned}$$

We first compare $\mathrm{cap}(f(f(x))) \equiv f(f(x))$ and $f(\Diamond) \equiv \mathrm{cap}(f(a(b(f(x)))))$. Since $f(f(x)) \not\equiv f(\Diamond)$, in order to satisfy (4), we have to impose $f(f(x)) > f(\Diamond)$ to our ordering $>$. Thus, (4) should be due to **TLO:1**; and so we require $f(f(x)) >_{tlo'(\mathcal{G})} f(x)$, which immediately follows from **TLO':1**.

To satisfy (5), we first need $\mathrm{cap}(f(a(g(x)))) \equiv f(\Diamond) > \Diamond \equiv \mathrm{cap}(b(x))$. Thus, (5) should be due to **TLO:1**, and so we require $f(a(g(x))) >_{tlo'(\mathcal{G})} x$. For this, by **TLO':2**, it suffices to show $g(x) >_{tlo'(\mathcal{G})} x$, which immediately follows from **TLO':1**.

To conclude, we need a reduction order satisfying $f(f(x)) > f(\Diamond)$ and $f(\Diamond) > \Diamond$. But this can be obtained by taking $\hat{\mathcal{R}} = \{f(f(x)) \to f(\Diamond), f(\Diamond) \to \Diamond\}$ and $>= \overset{+}{\to}_{\hat{\mathcal{R}}}$, since $\hat{\mathcal{R}}$ is terminating.

Thus, we can apply Theorem 13, and we conclude the termination of the TRS $\{(1), (2), (3)\}$.

*Remark.* As the example above, we can conclude termination of a given TRS $\mathcal{R}$ by choosing a reduction order $>= \overset{+}{\to}_{\hat{\mathcal{R}}}$ for some suitable terminating TRS $\hat{\mathcal{R}}$. Further, such $\hat{\mathcal{R}}$ can be computed effectively (when $\mathcal{R}$ is finite); below we describe this taking a single rule TRS for an example.

Suppose now that we are requested to detect termination of a TRS $\mathcal{R} = \{l \to r\}$. It is clear that the number of choices of $\mathcal{G}$ is finite. Fix an arbitrary $\mathcal{G}$;

make tree($l$) and tree($r$). Let $P$ and $Q$ be the sets of nodes of tree($l$) and tree($r$), respectively. Since $P$ and $Q$ are finite, and so is the powerset $\mathcal{P}(P \times Q)$ of the direct product of $P$ and $Q$. This set $\mathcal{P}(P \times Q)$ is the set of candidates for $\hat{\mathcal{R}}$. Indeed, if $\hat{\mathcal{R}} \notin \mathcal{P}(P \times Q)$ then we can alternatively take as $\hat{\mathcal{R}}$ the terminating TRS $\{l' \to r' \mid l' \xrightarrow{+}_{\hat{\mathcal{R}}} r', l' \in P, r' \in Q\} \in \mathcal{P}(P \times Q)$, which has the same effect as $\hat{\mathcal{R}}$ for our technique.

Further, if we are given with a decision procedure $\mathcal{D}$ which returns 'yes' for inputs of some terminating TRSs (e.g. a procedure which returns 'yes' for TRSs whose rules are all directed w.r.t. rpo induced from a given precedence), then we have a procedure which decides whether our technique is effective: For each member $\mathcal{R}'$ of $\mathcal{P}(P \times Q)$, check that whether $\mathcal{D}(\mathcal{R}')$ is 'yes.' If positive then check whether $l >_{tlo(\mathcal{G})} r$ for $>= \xrightarrow{+}_{\mathcal{R}'}$; this check can be done in a finite time when $\mathcal{R}'$ is terminating, because the set $\{t \mid s(> \cup \triangleright)^+ t\}$ is finite for each $s$ by König's Lemma. Thus, through this procedure, we can find $\hat{\mathcal{R}}$ $(= \mathcal{R}')$ if it exists.

The conditions in Theorem 13 can not be dropped:

*Example 2.* Let $\mathcal{G} = \{a, b\}$,

$$\mathcal{R} \left\{ a(x) \to f, \qquad \mathcal{S} \left\{ b(x, x) \to b(x, a(x)). \right.\right.$$

Note that $\mathcal{S}$ is a terminating non-collapsing non-duplicating TRS and that $\mathcal{R}$ satisfies condition 2 of the theorem but not condition 1. Now, let $\hat{\mathcal{R}} = \{\Diamond \to f\}$ and take $>= \xrightarrow{+}_{\hat{\mathcal{R}}}$. Clearly, $\hat{\mathcal{R}}$ is terminating, and so $>$ is a reduction order. Also, we have $a(x) >_{tlo(\mathcal{G})} f$. But $\mathcal{R} \cup \mathcal{S}$ is not terminating since $b(f, f) \to_{\mathcal{R} \cup \mathcal{S}} b(f, a(f)) \to_{\mathcal{R} \cup \mathcal{S}} b(f, f)$.

*Example 3.* Let $\mathcal{G} = \{a, b, c\}$,

$$\mathcal{R} \left\{ a(x) \to c, \qquad \mathcal{S} \left\{ b(c, x) \to b(a(x), a(c)). \right.\right.$$

Note that $\mathcal{S}$ is a terminating non-collapsing non-duplicating TRS and that $\mathcal{R}$ satisfies condition 1 of the theorem but not condition 2. Now, take $>$ as the empty order, which is a reduction order. Then, we have $a(x) >_{tlo(\mathcal{G})} c$. But $\mathcal{R} \cup \mathcal{S}$ is not terminating, since $b(c, a(c)) \to_{\mathcal{R} \cup \mathcal{S}} b(a(c), a(c)) \to_{\mathcal{R} \cup \mathcal{S}} b(c, a(c))$.

## 4 Related works

A modular termination result in [9] follows from our result.

**Corollary 14.** *Let $\mathcal{R}$, $\mathcal{S}$ be terminating TRSs, and suppose that $\mathcal{S}$ is non-collapsing non-duplicating and that the sets of function symbols that appear in $\mathcal{R}$ and $\mathcal{S}$ are disjoint. Then $\mathcal{R} \cup \mathcal{S}$ is also terminating.*

*Proof.* Let $\mathcal{G}$ be the set of function symbols that appear in $\mathcal{S}$. By assumption, no symbol from $\mathcal{G}$ appears in $\mathcal{R}$. Thus, by putting $>= \xrightarrow{+}_{\mathcal{R}}$, we obviously have $l >_{tlo(\mathcal{G})} r$ for all $l \to r \in \mathcal{R}$. Since $l$ does not contain function symbols from $\mathcal{G}$, the conditions of Theorem 13 are trivially satisfied, and hence the corollary follows. □

Now we compare our result with hierarchical modular termination results [2][3] [8][10].

*Example 4.* Let

$$\mathcal{R}\begin{cases} f(f(x)) \to f(f(x)+f(x)) \\ g(0,x,f(x)) \to f(x+s(x)), \end{cases} \qquad \mathcal{S}\begin{cases} 0+0 \to 0 \\ s(x)+0 \to s(x) \\ x+s(y) \to s(x+y). \end{cases}$$

Then let $\mathcal{G} = \{+, s, 0\}$ and we can conclude $\mathcal{R} \cup \mathcal{S}$ is terminating by Theorem 13, since $\hat{\mathcal{R}} = \{f(f(x)) \to f(\Diamond), g(\Diamond, x, f(x)) \to f(\Diamond)\}$ is terminating. Neither hierarchical modularity results in [2] (Theorems 15-17), which require $\mathcal{R}$ to be left-linear, nor those in [2] (Theorems 19-21), [3] and [10], which can not have nested occurrences of $f$ in the left-hand sides of rules in $\mathcal{R}$, apply to the combination of $\mathcal{R}$ and $\mathcal{S}$. In addition, $\mathcal{R}$ is not a proper extension of $\mathcal{S}$, and so [8] also does not apply.

Our work was inspired from the dummy elimination [6], which was further developed in [4][5]. Unfortunately, our result does not cover even the result in [6]. In our notation, the dummy elimination transformation $\mathcal{E}$ for a given a set $\mathcal{G}$ of function symbols to be eliminated can be defined as follows:

$$\mathcal{E}(\mathcal{R}) = \bigcup_{l \to r \in \mathcal{R}} \{\mathrm{cap}(l) \to s \mid s \in \mathrm{nodes}(r)\}.$$

(Note that $\mathrm{cap}(l)$ and $\mathrm{nodes}(r)$ depend on $\mathcal{G}$.)

**Corollary 15.** *Let $\mathcal{R}$ be a TRS and $\mathcal{G}$ the set of function symbols to be eliminated. Suppose that for any $l \to r \in \mathcal{R}$*

1. $\mathrm{root}(l) \in \mathcal{G}$ *implies* $\mathrm{root}(r) \in \mathcal{G}$, *and*
2. *no variable appears directly below a symbol from $\mathcal{G}$ in $l$ when $\mathcal{G}$ contains a constant.*

*Then, if $\mathcal{E}(\mathcal{R})$ is terminating then $\mathcal{R}$ is terminating.*

*Proof.* Let $>= \stackrel{+}{\to}_{\mathcal{E}(\mathcal{R})}$. It is easy to see $l >_{tlo(\mathcal{G})} r$ for all $l \to r \in \mathcal{R}$, by applying **TLO:1** and **TLO':1**. □

Because of additional restrictions 1 and 2, our result covers only a part of dummy elimination technique. But the next example shows that even when $\mathcal{S} = \emptyset$, there are cases that our technique has an advantage over the (general) dummy elimination.

*Example 5.* Let $\mathcal{G} = \{a\}$ and

$$\mathcal{R}_1\begin{cases} f(f(x)) \to f(a(f(x))) \\ f(a(g(x))) \to a(x), \end{cases} \qquad \mathcal{R}_1'\begin{cases} f(f(x)) \to f(\Diamond) \\ f(\Diamond) \to \Diamond, \end{cases}$$

$$\mathcal{R}_2\left\{ a(g(g(x))) \to a(g(a(g(x)))), \right. \qquad \mathcal{R}_2'\left\{ g(g(x)) \to g(\Diamond), \right.$$

$$\mathcal{R}_3\left\{ f(f(a(x))) \to f(a(f(a(x)))), \right. \qquad \mathcal{R}_3'\left\{ f(f(\Diamond)) \to f(\Diamond). \right.$$

Termination of $\mathcal{R}_i$ can be shown using termination of $\mathcal{R}'_i$ ($i \in \{1,2,3\}$), respectively, while the (general) dummy elimination is not effective.

Similar to the dummy elimination technique, when $\mathcal{S} = \emptyset$ eliminating more than one symbol simultaneously just weaken the power of the technique; successive elimination of symbols works better; see [4]. But in Example 1 it is witnessed that there is a case that successive elimination of $a$ and $b$ is not effective, while the simultaneous elimination of $a$ and $b$ is.

It should be remarked here that a much simpler proof for the correctness of the dummy elimination transformation appeared in [10]. But it seems that their proof technique is not directly applicable to our case.

Very recently, M. C. F. Ferreira et al. showed in [7] that the dummy elimination is sound under the AC-theory for dummy symbols, i.e. termination of $\mathcal{H}(\mathcal{R})$ implies termination of $\mathcal{R}$ modulo $\mathcal{E}$ where $\mathcal{H}(\mathcal{R}) = \bigcup_{l \to r \in \mathcal{R}} \{\mathrm{cap}(l) \to s \mid s \in \mathrm{nodes}(r)\}$ and $\mathcal{E} = \{a(x,y) \approx a(y,x) \mid a \in \mathcal{G}\} \cup \{a(x,a(y,z)) \approx a(a(x,y),z) \mid a \in \mathcal{G}\}$.

Our technique can be applied in a similar way:

**Theorem 16.** *Let $\mathcal{G}$ be a set of function symbols and $\mathcal{R}$ a TRS such that*

1. $\mathrm{root}(l) \in \mathcal{G}$ *implies* $\mathrm{root}(r) \in \mathcal{G}$ *and*
2. *no variable appears directly below a symbol from $\mathcal{G}$ in $l$ when $\mathcal{G}$ contains a constant*

*for any $l \to r \in \mathcal{R}$. If there exists a reduction order $>$ on $\mathcal{T}(\mathcal{F}\backslash\mathcal{G} \cup \{\diamondsuit\}, \mathcal{V})$ such that $l >_{tlo(\mathcal{G})} r$ for any $l \to r \in \mathcal{R}$, then $\mathcal{R}$ is terminating modulo $\mathcal{E}$ for any non-collapsing variable-preserving equational system $\mathcal{E}$ whose function symbols are contained in $\mathcal{G}$.*

## 5 Concluding remarks

We have proposed a tree lifting ordering $>_{tlo(\mathcal{G})}$ induced from a reduction order $>$ and a set $\mathcal{G}$ of function symbols. We have shown that for a given TRS $\mathcal{R}$ such that for any $l \to r \in \mathcal{R}$ (1) $\mathrm{root}(l) \in \mathcal{G}$ implies $\mathrm{root}(r) \in \mathcal{G}$ and (2) no variable appears directly below a symbol from $\mathcal{G}$ in $l$ when $\mathcal{G}$ contains a constant, if $l >_{tlo(\mathcal{G})} r$ for any $l \to r \in \mathcal{R}$ then $\mathcal{R} \cup \mathcal{S}$ is terminating for any non-collapsing non-duplicating terminating TRS $\mathcal{S}$ whose function symbols are contained in $\mathcal{G}$. Furthermore, the reduction order $>$ that works on our technique is effectively obtained from $\mathcal{R}$ when $\mathcal{R}$ is finite.

Also, we have shown that our result covers a modular termination result [9] and that there are cases that our technique has an advantage over the (general) dummy elimination [6]([4]).

## Acknowledgments

The authors are grateful to Maria C. F. Ferreira and Aart Middeldorp for detecting errors in the previous version [1] of the paper. Thanks are due to the RTA'98 referees for corrections, improvements and constructive comments.

This work is partially supported by Grants from Ministry of Education, Science and Culture of Japan, #09245212 and #07680347.

## References

1. T. Aoto and Y. Toyama. Tree lifting orderings for termination transformations of term rewriting systems. Research Report IS-RR-97-0033F, School of Information Science, JAIST, 1997.
2. N. Dershowitz. Hierarchical termination. In *Proceedings of the 4th International Workshop on Conditional (and Typed) Rewriting Systems (CTRS-94)*, *LNCS 968*, pages 89–105. Springer-Verlag, 1995.
3. M. Fernández and J.-P. Jouannaud. Modular termination of term rewriting systems revisited. In *Proceedings of the 10th Workshop on Specification of Abstract Data Types*, *LNCS 906*, pages 255–272. Springer-Verlag, 1995.
4. M. C. F. Ferreira. *Termination of term rewriting.* PhD thesis, Utrecht University, 1995.
5. M. C. F. Ferreira. Dummy elimination in equational rewriting. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications (RTA-96)*, *LNCS 1103*, pages 78–92. Springer-Verlag, 1996.
6. M. C. F. Ferreira and H. Zantema. Dummy elimination: making termination easier. In *Proceedings of the 10th International Conference on Foundamentals of Computation Theory (FCT'95)*, *LNCS 965*, pages 243–252. Springer-Verlag, 1995.
7. M. C. F. Ferreira, D. Kesner and L. Puel. Reducing AC-termination to termination. Manuscript, 1997.
8. M. R. K. Krishna Rao. Modular proofs for completeness of hierarchical term rewriting systems. *Theoretical Computer Science*, 151:487–512, 1995.
9. A. Middeldorp. A sufficient condition for the termination of the direct sum of term rewriting systems. In *Proceedings of the 4th annual IEEE Symposium on Logic in Computer Science (LICS'89)*, pages 396–401. IEEE Computer Society Press, 1989.
10. A. Middeldorp, H. Ohsaki and H. Zantema. Transforming termination by self-labelling. In *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*, *LNAI 1104*, pages 373–387. Springer-Verlag, 1996.
11. A. Middeldorp and H. Zantema. Simple termination of rewrite systems. *Theoretical Computer Science*, 175(1):127–158, 1997.

This article was processed using the LaTeX macro package with LLNCS style