# Towards Automated Termination Proofs Through "Freezing"

Hongwei Xi

Department of Mathematical Sciences
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
e-mail: hwxi@cs.cmu.edu

**Abstract.** We present a transformation technique called *freezing* to facilitate automatic termination proofs for left-linear term rewriting systems. The significant merits of this technique lie in its simplicity, its amenability to automation and its effectiveness, especially, when combined with other well-known methods such as recursive path orderings and polynomial interpretations. We prove that applying the freezing technique to a left-linear term rewriting system always terminates. We also show that many interesting TRSs in the literature can be handled with the help of *freezing* while they elude a lot of other approaches aiming for generating termination proofs automatically for term rewriting systems. We have mechanically verified all the left-linear examples presented in this paper.

## 1 Introduction

It is an ever present task to decide whether a given term rewriting system(TRS) is (strongly) terminating. While this problem is not decidable in general [14], many approaches, such as path orderings [18,5,7], Knuth-Bendix ordering [15], semantic interpretations[17,21,12], transformation orderings[4,20] and semantic labelling [22], have been developed to give termination proofs. See [6,19] for surveys.

In this paper, we are primarily concerned with automatic termination proofs for TRSs. We propose a technique called *freezing*, which transforms a given left-linear TRS into a family of left-linear TRSs such that the termination of any TRS in this family implies the termination of the original TRS. The significant merits of the freezing technique lie in its simplicity, its amenability to automation and its effectiveness. Also we prove that the transformation terminates for all left-linear TRSs.

In practice, we know that most automatic termination proving methods rely on *simplification* orderings. [1] On the other hand, there exist numerous interesting (left-linear) TRSs whose termination cannot be proven by simplification

[1] We point out that automatic techniques have been developed, for instance in [20,1], which can handle self-embedded rewrite rules.

T. Nipkow (Ed.): Rewriting Techniques and Applications, RTA-98, LNCS 1379, pp. 271–285, 1998.

orderings. With the help of the freezing technique, we are able to show that many among these TRSs can be transformed into those whose termination can be shown by some methods based on simplification orderings.

We present some preliminaries in Section 2. Then in Section 3 we illustrate the basic idea behind the freezing technique by a simple example. We formally introduce the freezing technique in Section 4, and prove the termination of the freezing technique when it is applied to a left-linear TRS. We then present some examples in Section 5 to illustrate the effectiveness of the freezing technique, compare our work with some related work in Section 6, and conclude.

## 2 Preliminaries

In general, we shall stick close to the notations in [9] though some modifications may occur. We assume that the reader is familiar with term rewriting. The following is a brief summary of the notations we shall use later.

We fix a countably infinite set of variables: $x, y, z, \ldots$, which is denoted by $\mathcal{X}$ throughout the paper. We use $\mathcal{F}$ for a (finite) set of function symbols: $f, g, h, F, G \ldots$, where each function symbol has a fixed arity $\mathcal{A}r$; $\mathcal{T}(\mathcal{F})$ for the set of (first-order) terms: $l, r, s, t, \ldots$ over $\mathcal{F}$ and $\mathcal{X}$; $\mathcal{V}ar(t)$ for the set of variables in $t$; $\boldsymbol{t}_{m,n}$ for a sequence of terms: $t_m, t_{m+1}, \ldots, t_n (m > n$ means this is an empty sequence); $\rho$, $\sigma$ for substitutions, and $\epsilon$ for the empty substitution; $\langle \mathcal{F}, \mathcal{R} \rangle$ for a TRS, where $\mathcal{R}$ consists of the rewriting rules of the form $l \to r$ such that $l, r \in \mathcal{T}(\mathcal{F})$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$; $\langle \mathcal{F}, \mathcal{R} \rangle$ is a left-linear rewrite system if for every rewrite rule $l \to r$ in $\mathcal{R}$ there exists no variable which occurs more than once in $l$. Note that we may use $f, g, h$ for some specific function symbols in certain parts of our presentation. As a consequence, we use $F, G$ ranging over $\mathcal{F}$ when this happens, avoiding possible confusion.

We feel that it is convenient to use the notion of *contexts* in reasoning, and we present a definition. See [8] for some similar use of contexts.

**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 $F$ is a function symbol with arity $\mathcal{A}r(F) = n \geq 1$ and $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$ are terms and $C$ is a context.*

*$C[t]$ is the term obtained from replacing the "hole" $[]$ in $C$ with term $t$, and $C[C']$ is the context obtained from replacing the "hole" $[]$ in $C$ with context $C'$.*

Given a TRS $\langle \mathcal{F}, \mathcal{R} \rangle$, we write $t \to_{\mathcal{R}} t'$ for $t, t' \in \mathcal{T}(\mathcal{F})$ if $t = C[l\sigma]$ and $t' = C[r\sigma]$ for some $C$ and $\sigma$, where $l \to r \in \mathcal{R}$. If $t \to_{\mathcal{R}} t'$, we say $t$ rewrites to $t'$ (in one step). We use $\to_{\mathcal{R}}^{+}$ for the transitive closure of $\to_{\mathcal{R}}$ and $\to_{\mathcal{R}}^{*}$ for the reflexive and transitive closure of $\to_{\mathcal{R}}$. A notation of the following form stands for a finite or infinite $\to_{\mathcal{R}}$-rewriting sequence (from $t_0$):

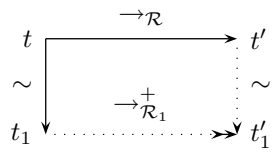
$$t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots$$

We say $\mathcal{R}$ or $\to_{\mathcal{R}}$ is terminating if there exists no infinite $\to_{\mathcal{R}}$-rewriting sequence.

## 3 The Basic Idea

It is a well-known technique to prove the termination of a TRS via "simulation", as formally shown below.

**Definition 2.** *Let $\langle \mathcal{F}, \mathcal{R} \rangle$ and $\langle \mathcal{F}_1, \mathcal{R}_1 \rangle$ be two TRSs such that $\mathcal{F} \subseteq \mathcal{F}_1$, and $\sim \subseteq \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F}_1)$ be a relation satisfying $t \sim t$ for all $t \in \mathcal{T}(\mathcal{F})$. We write $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$ if for any given $t \sim t_1$ and $t \to_{\mathcal{R}} t'$ there exists $t'_1$ such that $t_1 \to^+_{\mathcal{R}_1} t'_1$ and $t' \sim t'_1$.*



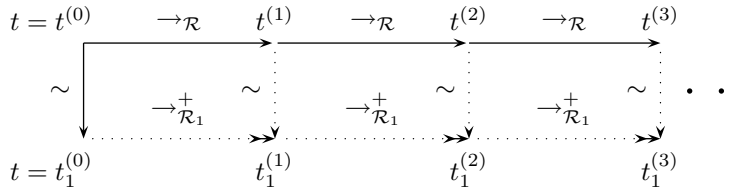

The need of $t \sim t$ for all $t \in \mathcal{T}(\mathcal{F})$ is to *start* the simulation as shown in the proof of next lemma.

**Lemma 1.** *If $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$, then $\to_{\mathcal{R}}$ is terminating if $\to_{\mathcal{R}_1}$ is terminating.*

*Proof.* Assume that there exists an infinite $\to_{\mathcal{R}}$ rewriting sequence as follows.

$$t = t^{(0)} \to_{\mathcal{R}} t^{(1)} \to_{\mathcal{R}} t^{(2)} \to_{\mathcal{R}} t^{(3)} \to_{\mathcal{R}} \cdots$$

We can then construct an infinite $\to_{\mathcal{R}_1}$ rewriting sequence as shown in the diagram below.




Therefore, there exists no infinite $\to_{\mathcal{R}}$ rewriting sequence since $\to_{\mathcal{R}_1}$ is terminating, i.e., $\to_{\mathcal{R}}$ is also terminating.

There exist many variations of the above approach. We use this formulation since it suffices for the development of our technique.

Note that Lemma 1 can yield an approach to termination proofs as follows. Suppose we want to prove that $\to_{\mathcal{R}}$ is terminating for some given TRS $\langle \mathcal{F}, \mathcal{R} \rangle$. We first construct another TRS $\langle \mathcal{F}_1, \mathcal{R}_1 \rangle$ with $\mathcal{F} \subseteq \mathcal{F}_1$ and a relation $\sim \subseteq \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F}_1)$ satisfying $t \sim t$ for all $t \in \mathcal{T}(\mathcal{F})$, and prove $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$. We then prove that $\to_{\mathcal{R}_1}$ is terminating. This yields that $\to_{\mathcal{R}}$ is terminating by Lemma 1. Clearly, we have to be able to construct $\mathcal{R}_1$ and $\sim$ in some way so that a termination proof for $\mathcal{R}_1$ can be given "more easily" than for $\mathcal{R}$, and this is the main subject of the paper. We now present a simple example before going into further details.

*Example 1.* Let $\mathcal{F} = \{f, g\}$ and $\mathcal{R}$ consist of the following rule[6].

$$f(f(x)) \to f(g(f(x)))$$

Clearly, it cannot be proven with simplification orderings that $\to_{\mathcal{R}}$ is terminating since the left-hand side of the rule is self-embedded in the right-hand side. Now we introduce the notion of *freezing*. Let $\underline{fg}$ be a new unary function symbol, and we define $\sim$ as follows.

$$\begin{array}{ll} x \sim x & \text{for all } x \in \mathcal{X}; \\ f(t) \sim f(t_1), g(t) \sim g(t_1), \text{ and } f(g(t)) \sim \underline{fg}(t_1) & \text{if } t \sim t_1; \end{array}$$

In other words, if $t_1$ is obtained from t by *freezing* some occurrences of $f(g(\cdots))$ into $\underline{fg}(\cdots)$, then $t \sim t_1$. We also extend $\sim$ to contexts as follows.

$$\begin{array}{ll} [] \sim [] & ; \\ f(C) \sim f(C_1), g(C) \sim g(C_1), \text{ and } f(g(C)) \sim \underline{fg}(C_1) & \text{if } C \sim C_1; \end{array}$$

Clearly, $C \sim C_1$ and $t \sim t_1$ implies $C[t] \sim C_1[t_1]$. Let $\mathcal{R}_1$ be the TRS consisting of the following rules, which can be generated automatically as shown in Section 4.

$$(1). \quad f(f(x)) \to \underline{fg}(f(x)) \qquad\qquad (2). \quad f(\underline{fg}(x)) \to \underline{fg}(f(g(x)))$$

Assume $t \sim t_1$ and $t \to_{\mathcal{R}} t'$. Then $t$ is of the form $C[f(f(s))]$ and $t'$ of the form $C[f(g(f(s)))]$. We do a case analysis on the form of $t_1$, showing $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$.

- $t_1 = C_1[f(f(s_1))]$, where $C \sim C_1$ and $s \sim s_1$. Let $t_1' = C_1[\underline{fg}(f(s_1))]$, and we have $t_1 \to_{\mathcal{R}_1} t_1'$ by rule (1) and $t' \sim t_1'$.
- $t_1 = C_1[f(\underline{fg}(s_1))]$, where $C \sim C_1$ and $f(s) \sim \underline{fg}(s_1)$. Obviously, $f(s) \sim f(g(s_1))$ by the definition of $\sim$. Let $t_1' = C_1[\underline{fg}(f(g(s_1)))]$, and we have $t_1 \to_{\mathcal{R}_1} t_1'$ by rule (2) and $t' \sim t_1'$.

Later, Proposition 2 will justify that the case analysis is complete. Therefore, $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$. Note that a termination proof for $\mathcal{R}_1$ can be given using rpo with the precedence: $f \succ \underline{fg} \succ g$. By Lemma 1, $\to_{\mathcal{R}}$ is terminating.

We now make an important observation. In the proof of $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$, if we replace $\mathcal{R}_1$ with any $\mathcal{R}_*$ consisting of the following rules

$$f(f(x)) \to t_1 \qquad\qquad f(\underline{fg}(x)) \to t_2,$$

where $t_1$ and $t_2$ are any terms satisfying $f(g(f(x))) \sim t_1$ and $f(g(f(g(x)))) \sim t_2$, then $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_*$ can be proven similarly. This means that we can construct a family of $\mathcal{R}_*$ such that $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_*$ can be proven uniformly, and therefore the termination of any $\mathcal{R}_*$ in this family implies the termination of $\mathcal{R}$. Now our objective is to generate this family of TRSs automatically.

## 4 The Freezing Technique

In this section, we first give a formal presentation of the freezing technique for left-linear TRSs, and prove the termination of this technique. We also show with an example that the termination of the freezing technique can no longer be guaranteed if it is extended directly to a non-left linear rewriting system.

### 4.1 Left-Linear TRSs

We fix a left-linear TRS $\langle \mathcal{F}, \mathcal{R} \rangle$ such that there exists a function symbol $f \in \mathcal{F}$ with arity $\mathcal{A}r(f) = n_f > 0$. Let $g$ be some function symbol in $\mathcal{F}$ with arity $\mathcal{A}r(g) = n_g$. We may choose $g$ to be the same as $f$. Also let $\mathcal{F}_* = \mathcal{F} \cup \{\underline{fg}\}$, where $\underline{fg}$ is a new function symbol with arity $\mathcal{A}r(\underline{fg}) = n_f + n_g - 1$.

**Definition 3.** *Let $m$ be a natural number between 1 and $n_f$. The $(f, g, m, \underline{fg})$-freezing relation $\sim \subseteq \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{F}_*)$ is defined by the following derivation rules: $t \sim t^*$ if and only if the judgement $t \sim t^*$ is derivable.*

$$\frac{}{x \sim x} \qquad \frac{t_1 \sim t_1^* \quad \cdots \quad t_{\mathcal{A}r(F)} \sim t_{\mathcal{A}r(F)}^*}{F(t_1, \ldots, t_{\mathcal{A}r(F)}) \sim F(t_1^*, \ldots, t_{\mathcal{A}r(F)}^*)}$$

$$\frac{t_1 \sim t_1^* \quad \cdots \quad t_{m-1} \sim t_{m-1}^* \quad t_m \sim g(\boldsymbol{s}_{1,n_g}^*) \quad t_{m+1} \sim t_{m+1}^* \quad \cdots \quad t_{n_f} \sim t_{n_f}^*}{f(\boldsymbol{t}_{1,m-1}, t_m, \boldsymbol{t}_{m+1,n_f}) \sim \underline{fg}(\boldsymbol{t}_{1,m-1}^*, \boldsymbol{s}_{1,n_g}^*, \boldsymbol{t}_{m+1,n_f}^*)}$$

*Similarly, $C \sim C_*$ can be defined by treating $[]$ as a variable. Note that $F$ ranges over $\mathcal{F}$ in the definition.*

**Proposition 1.** *We have the following.*

1. *If $C \sim C_*$ and $t \sim t^*$, then $C[t] \sim C_*[t^*]$.*
2. *If $t \sim t_1$ then $t\sigma \sim t_1\sigma$ for every substitution $\sigma$.*
3. *If $t \sim C[t_1]$ and $t_1 \sim t_2$, then $t \sim C[t_2]$.*

**Proposition 2.** *Suppose $t = C[s] \sim t^*$. Then one of the following two cases holds.*

- *$t^* = C_*[s^*]$ for some $C_*$ and $s^*$ such that $C \sim C^*$ and $s \sim s^*$.*
- *$C$ is of the form $C'[f(\boldsymbol{t}_{1,m-1}, [], \boldsymbol{t}_{m+1,n_f})]$ and $s$ is of the form $g(\boldsymbol{s}_{1,n_g})$. $t^* = C_*[s^*]$ for some $C_*$ and $s^*$ is of the form $\underline{fg}(\boldsymbol{t}_{1,m-1}^*, \boldsymbol{s}_{1,n_g}^*, \boldsymbol{t}_{m+1,n_f}^*)$ such that $C' \sim C_*$ and $f(\boldsymbol{t}_{1,m-1}, s, \boldsymbol{t}_{m+1,n_f}) \sim s^*$.*

*Proof.* This follows from a structural induction on the derivation of $t \sim t^*$.

The use of Proposition 2 is to do case analysis in proofs.

The following definition of $\rhd$ is crucial for the development of the freezing technique. We will list some properties of $\rhd$ to ease the understanding as well as presenting an simple example. Also a comparison of the definition with *narrowing* may be helpful.

**Definition 4.** *We define the relation $\rhd$ as follows, which is parameterized over $(f, g, m, \underline{fg})$.*

$$\frac{}{x \rhd \langle x, \epsilon \rangle}\ (var)$$

$$\frac{t_1 \rhd \langle t_1^*, \sigma_1 \rangle \quad \cdots \quad t_{\mathcal{A}r(F)} \rhd \langle t_{\mathcal{A}r(F)}^*, \sigma_{\mathcal{A}r(F)} \rangle}{F(\boldsymbol{t}_{1,\mathcal{A}r(F)}) \rhd \langle F(\boldsymbol{t}_{1,\mathcal{A}r(F)}^*), \bigcup_{i=1}^{\mathcal{A}r(F)} \sigma_i \rangle}\ (fun)$$

$$\frac{t_i \rhd \langle t_i^*, \sigma_i\rangle \text{ for } 1 \le i \ne m \le n_f}{f(\boldsymbol{t}_{1,m-1}, x, \boldsymbol{t}_{m+1,n_f}) \rhd \langle \underline{fg}(\boldsymbol{t}^*_{1,m-1}, x_1, \ldots, x_{n_g}, \boldsymbol{t}^*_{m+1,n_f}), \sigma\rangle,} \ (inner)$$

*where* $x_1, \ldots, x_{n_g}$ *are fresh variables, and*

$$\sigma = (\bigcup_{1\le i\ne m\le n} \sigma_i) \cup \{x \mapsto g(x_1, \ldots, x_{n_g})\}.$$

$$\frac{t_i \rhd \langle t_i^*, \sigma_i\rangle \text{ for } 1 \le i \ne m \le n_f \quad t_m \rhd \langle g(\boldsymbol{s}^*), \sigma_m\rangle}{f(\boldsymbol{t}_{1,m-1}, t_m, \boldsymbol{t}_{m+1,n_f}) \rhd \langle \underline{fg}(\boldsymbol{t}^*_{1,m-1}, \boldsymbol{s}^*_{1,n_g}, \boldsymbol{t}^*_{m+1,n_f}), \bigcup_{i=1}^{n_f} \sigma_i\rangle} \ (freeze)$$

$$\frac{l \rhd \langle l^*, \sigma\rangle \quad r^* = r\sigma}{l \to r \rhd \langle l^* \to r^*, \sigma\rangle}$$

$$\frac{l \rhd \langle g(\boldsymbol{l}^*_{1,n_g}), \sigma\rangle \quad r^* = f(x_1, \ldots, x_{m-1}, r\sigma, x_{m+1}, \ldots, x_{n_f})}{l \to r \rhd \langle \underline{fg}(x_1, \ldots, x_{m-1}, \boldsymbol{l}^*_{1,n_g}, x_{m+1}, \ldots, x_{n_f}) \to r^*, \sigma\rangle} \ (outer),$$

*where* $x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_{n_f}$ *are fresh variables.*

*The* master $(f, g, m, \underline{fg})$*-frozen version* $\mathcal{R}_*$ *of* $\mathcal{R}$ *is defined below.*

$$\mathcal{R}_* = \{l^* \to r^* \mid l \to r \rhd \langle l^* \to r^*, \sigma\rangle \text{ for some } l \to r \in \mathcal{R}\}$$

Some properties of $\rhd$ include (i) $t \to \langle t^*, \sigma\rangle$ implies $t\sigma = t^*$, (ii) $t \to \langle t, \epsilon\rangle$ for all every term $t$, and (iii) $l \to r \rhd \langle l \to r, \epsilon\rangle$. It can be readily verified that $\mathcal{R}_*$ is left-linear. If $\mathcal{R}$ is finite, then the finiteness of $\mathcal{R}_*$ clearly holds modulo renaming, i.e. we treat every pair of rules $l \to r$ and $l' \to r'$ in $\mathcal{R}_*$ as the same if the former can be obtained from renaming some variables in the latter and vice versa.

*Example 2.* This example is due to L. Bachmair. Let $\mathcal{R}$ consist of the following rules.

$$1.\ f(h(x)) \to f(i(x)) \quad 2.\ g(i(x)) \to g(h(x)) \quad 3.\ h(a) \to b \quad 4.\ i(a) \to b$$

Let us compute the master $(g, h, 1, \underline{gh})$-freezing version $\mathcal{R}_*$ of $\mathcal{R}$. It is straightforward to obtain the following,

$$\begin{array}{cc} f(h(x)) \rhd \langle f(h(x)), \epsilon\rangle & g(i(x)) \rhd \langle g(i(x)), \epsilon\rangle \\ h(a) \rhd \langle h(a), \epsilon\rangle & i(a) \rhd \langle i(a), \epsilon\rangle \end{array}$$

which include all the possibilities since there is no application of the *(inner)* rule in this example. Hence $\mathcal{R}_*$ consists of rule 1,2,3,4 and rule 5: $\underline{gh}(a) \to g(b)$. Let $\mathcal{R}_1$ be a TRS consisting of rule $1, 3, 4, 5$ and rule $2'$: $g(i(x)) \to \underline{gh}(x)$, then $\mathcal{R}_1$ is a $(g, h, 1, \underline{gh})$-freezing version of $\mathcal{R}$. The termination of $\mathcal{R}_1$ can be shown by recursive path ordering with precedence: $h \succ i \succ \underline{gh} \succ g, b$. Notice that $\mathcal{R}_1$ is the generated TRS $\mathcal{S}$ in Example 14 in [20].

**Lemma 2.** *Let* $l$ *be a linear term and* $\rho$ *be a substitution. If* $t = l\rho$ *and* $t \sim t^*$*, then there exists a term* $l^*$ *and a substitution* $\rho^*$ *such that* $t^* = l^*\rho^*$ *and* $l \rhd \langle l^*, \sigma\rangle$ *and* $\rho = \sigma\rho^*$.

*Proof.* We proceed by a structural induction on $t$. If $t$ is a variable then the case is trivial. Otherwise, we have the following cases.

- $t = F(\boldsymbol{t}_{1,n})$, where $F \neq f$. If $l$ is a variable $x$, then $\rho = \{x \mapsto t\}$. Let $l^* = x$, $\sigma = \epsilon$ and $\rho^* = \{x \mapsto t^*\}$, and we are done. We now assume that $l$ is not a variable. Since $l$ is linear, $l$ is of the form $F(\boldsymbol{l}_{1,n})$, and for $1 \leq i \leq n$, $t_i = l_i\rho_i$ for some $\rho_i$ and $\rho = \bigcup_{i=1}^{n} \rho_i$. Since $F \neq f$, $t^*$ is of the form $F(\boldsymbol{t}^*_{1,n})$ and $t_i \sim t^*_i$ for $1 \leq i \leq n$. By induction hypothesis, for $1 \leq i \leq n$, there exist $l^*_i$ and $\rho^*_i$ such that $t^*_i = l^*_i\rho^*_i$ and $l_i \rhd \langle l^*_i, \sigma_i\rangle$ and $\rho_i = \sigma_i\rho^*_i$. Let $l^* = F(\boldsymbol{l}^*_{1,n})$ and $\rho^* = \bigcup_{i=1}^{n} \rho^*_i$, then $l \rhd \langle l^*, \sigma\rangle$ for $\sigma = \bigcup_{i=1}^{n} \sigma_i$. It can be readily verified that $t^* = l^*\rho^*$ and $\rho = \sigma\rho^*$ since $l$ is linear.
- $t = f(\boldsymbol{t}_{1,n_f})$. If $l$ is a variable or $t^*$ is of the form $f(\boldsymbol{t}^*_{1,n_f})$, where $t_i \sim t^*_i$ for $1 \leq i \leq n_f$, then this case can be proven as in the previous one. We now assume that $l$ is of the form $f(\boldsymbol{l}_{1,n_f})$ and $t^*$ is of form $\underline{fg}(\boldsymbol{t}^*_{1,m-1}, \boldsymbol{s^*}, \boldsymbol{t}^*_{m+1,n_f})$, where $t_i \sim t^*_i$ for $1 \leq i \neq m \leq n_f$ and $t_m \sim t^*_m = g(\boldsymbol{s^*})$. If $l_m$ is not a variable, then this case can also be proven as the previous one. We now assume $l = f(\boldsymbol{l}_{1,m-1}, x, \boldsymbol{l}_{m+1,n_f})$. Then for $1 \leq i \neq m \leq n$, $t_i = l_i\rho_i$ for some $\rho_i$ and

$$\rho = (\bigcup_{1 \leq i \neq m \leq n_f} \rho_i) \cup \{x \mapsto t_m\}.$$

  By induction hypothesis, for $1 \leq i \neq m \leq n_f$, there exist $l^*_i$ and $\rho^*_i$ such that $t^*_i = l^*_i\rho^*_i$ and $l_i \rhd \langle l^*_i, \sigma_i\rangle$ and $\rho_i = \sigma_i\rho^*_i$. Then

$$l \rhd \langle \underline{fg}(\boldsymbol{l}^*_{1,m-1}, x_1, \ldots, x_{n_g}, \boldsymbol{l}^*_{m+1,n_f}), \sigma\rangle,$$

  where $\sigma = (\bigcup_{1 \leq i \neq m \leq n_f} \sigma_i) \cup \{x \mapsto g(x_1, \ldots, x_{n_g})\}$. Since $t_m \sim g(\boldsymbol{s^*})$, $t_m = g(\boldsymbol{s})$ follows for some $\boldsymbol{s}$ and $\boldsymbol{s} \sim \boldsymbol{s^*}$. Let $l^* = \underline{fg}(\boldsymbol{l}^*_{1,m-1}, x_1, \ldots, x_{n_g}, \boldsymbol{l}^*_{m+1,n_f})$ and $\rho_m = \{x_1 \mapsto s^*_1, \ldots, x_{n_g} \mapsto s^*_{n_g}\}$ and $\rho = \cup_{i=1}^{n_f} \rho_i$, then it can be readily verified that $t^* = l^*\rho^*$ and $l \rhd \langle l^*, \sigma\rangle$ and $\rho = \sigma\rho^*$.

**Lemma 3.** *Let $l \to r$ be a left-linear rewrite rule and $\rho$ be a substitution. If $t = C[l\rho] \sim t^*$, then $l \to r \rhd \langle l^* \to r^*, \sigma\rangle$ for some $l^*, r^*, \sigma$, and $t^* = C_*[l^*\rho^*]$ for some context $C_*$ and $C[r\rho] \sim C_*[r^*\rho^*]$.*

*Proof.* Let $s = l\rho$. We do a case analysis on the form of $t^*$ according to Proposition 2.

- $t^* = C_*[s^*]$, where $C \sim C_*$ and $s \sim s^*$. By Lemma 2, we have $l^*$ and $\rho^*$ such that $s^* = l^*\rho^*$ and $l \rhd \langle l^*, \sigma\rangle$ and $\rho = \sigma\rho^*$. Therefore, $l \to r \rhd \langle l^* \to r^*, \sigma\rangle$ for $r^* = r\sigma$, and this yields $r^*\rho^* = r\sigma\rho^* = r\rho$. Clearly, $C[r\rho] \sim C_*[r^*\rho^*]$.
- $t^* = C_*[s^*]$, where $C = C'[f(\boldsymbol{t}_{1,m-1}, [], \boldsymbol{t}_{m+1,n_f})]$ and $C' \sim C^*$ and

$$s = f(\boldsymbol{t}_{1,m-1}, \boldsymbol{s}_{1,n_g}, \boldsymbol{t}_{m+1,n_f}) \sim s^* = \underline{fg}(\boldsymbol{t}^*_{1,m-1}, \boldsymbol{s^*}, \boldsymbol{t}^*_{m+1,n_f}).$$

  As the previous case, we have $l^*_0 = g(\boldsymbol{l^*})$, $r^*_0$ and $\rho^*$ such that $g(\boldsymbol{s^*}) = l^*_0\rho^*_0$ and $l \rhd \langle l^*_0, \sigma\rangle$ and $r\rho = r^*_0\rho^*_0$. Note $l \to r \rhd \langle l^* \to r^*, \sigma\rangle$, where

$$l^* = \underline{fg}(x_1, \ldots, x_{m-1}, \boldsymbol{l}^*_{1,n_g}, x_{m+1}, \ldots, x_{n_f})$$

and

$$r^* = f(x_1, \ldots, x_{m-1}, r_0^*, x_{m+1}, \ldots, x_{n_f}),$$

where $x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_{n_f}$ are fresh.
Let $\rho^* = \rho_0^* \cup \{x_1 \mapsto t_1^*, \ldots, x_{m-1} \mapsto t_{m-1}^*, x_{m+1} \mapsto t_{m+1}^*, \ldots, x_{n_f} \mapsto t_{n_f}^*\}$, then

$$C[r\rho] = C'[f(\boldsymbol{t}_{1,m-1}, r\rho, \boldsymbol{t}_{m+1,n_f})] \sim C_*[r^*\rho^*].$$

**Definition 5.** *Let $\mathcal{R}_1$ be a TRS such that $l_1 \to r_1 \in \mathcal{R}_1$ if and only if there exists $l^* \to r^* \in \mathcal{R}_*$ satisfying $l_1 = l^*$ and $r^* \sim r_1$. We call $\mathcal{R}_1$ a $(f, g, m, \underline{fg})$-frozen version of $\mathcal{R}$.*

In general, there exist a (finite) family of $(f, g, m, \underline{fg})$-frozen versions of $\mathcal{R}$. For instance, the following is another $(f, g, 1, \underline{fg})$-frozen version of the $\mathcal{R}$ given in Example 1.

(1). $f(f(x)) \to \underline{fg}(f(x))$ (2). $f(\underline{fg}(x)) \to \underline{fg}(\underline{fg}(x))$

**Theorem 1.** *$\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$ holds for every $(f, g, m, \underline{fg})$-frozen version $\mathcal{R}_1$ of $\mathcal{R}$.*

*Proof.* Assume $t \sim t^*$ and $t \to_{\mathcal{R}} t'$. Then $t = C[l\rho]$ for some $C$ and $l \to r \in \mathcal{R}$ and $t' = C[r\rho]$. By Lemma 3, there exists $l \to r \rhd \langle l^* \to r^*, \sigma \rangle$ such that $t^* = C_*[l^*\rho^*]$ for some $C_*$ and $C[r\rho] \sim C_*[r^*\rho^*]$. By the definition of $\mathcal{R}_*$, $l^* \to r^* \in \mathcal{R}_*$. Since $l_1 \to r_1 \in \mathcal{R}_1$ for some $l_1, r_1$ such that $l^* = l_1$ and $r^* \sim r_1$, we have $t^* \to_{\mathcal{R}_1} C_*[r_1\rho^*]$, and therefore $C[r\rho] \sim C_*[r_1\rho^*]$ holds by (2) and (3) of Proposition 1. This yields $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$.

*Remark 1.* We emphasize that the freezing technique as presented above can only be applied to left-linear TRSs. A straightforward extension to non-left linear TRSs may end with non-termination as shown by the following examples.

*Example 3.* Let $\mathcal{R}$ consist of the non-left linear rule $g(x, f(x)) \to c$, where $c$ is a constant. Then the master $(f, f, 1, \underline{ff})$-frozen version $\mathcal{R}$, i.e., the TRS generated according to the above procedure for left-linear TRSs, consists of infinitely many rules (modulo renaming). For instance, the following rules are in $\mathcal{R}_*$:

$$g(x, f(x)) \to c, g(f(x), \underline{ff}(x)) \to c, g(\underline{ff}(x), \underline{ff}(f(x)) \to c, \ldots$$

The study on how to extend *freezing* to non-left linear TRSs is our immediate research topic.

### 4.2 Freezing More Function Symbols

Given a TRS $\langle \mathcal{F}, \mathcal{R} \rangle$ in which there exist $f$, $g$ and $h$ such that $\mathcal{A}r(f) \geq 1$ and $\mathcal{A}r(f) + \mathcal{A}r(g) - 1 \geq 1$. Note that $f, g, h$ do not have to be distinct from each other. Let $\mathcal{R}_*$ be the master $(f, g, m_1, \underline{fg})$-frozen version of $\mathcal{R}$ and $\sim_1$ the freezing relation. Also let $\mathcal{R}_{**}$ be the master $(\underline{fg}, h, m_2, \underline{fgh})$-frozen version of $\mathcal{R}_*$ and $\sim_2$ the freezing relation. Then the master $(f, g, h, m_1, m_2, \underline{fgh})$-frozen version of $\mathcal{R}$ is defined as the TRS $\mathcal{R}^-_{**}$ which consists of all the rules $l \to r^-$ such that $l \to r \in \mathcal{R}_{**}$ for some $r^- \sim_1 r$ and $\underline{fg}$ has no occurrences in $l$. The corresponding freezing relation $\sim$ is defined as $t \sim t_2$ if $t \sim_1 t_1$ and $t_1 \sim_2 t_2$ for some $t_1$ and $\underline{fg}$ has no occurrence in $t_2$. Then $\mathcal{R}_1$ is a $(f, g, h, m_1, m_2, \underline{fgh})$-frozen version of $\mathcal{R}$ if $l_1 \to r_1 \in \mathcal{R}_1$ if and only if there exists $l \to r \in \mathcal{R}^-_{**}$ such that $l = l_1$ and $r \sim r_1$.

**Theorem 2.** *If $\mathcal{R}_1$ is a $(f, g, h, m_1, m_2, \underline{fgh})$-frozen version of a left-linear $\mathcal{R}$ and $\sim$ is the freezing relation, then $\mathcal{R} \overset{\sim}{\Longrightarrow} \mathcal{R}_1$.*

*Proof.* This simply follows from Theorem 1 with the explanation above.

This idea can certainly be generalized to freezing more function symbols, and we leave out the details.

We say that $\mathcal{R}$ is a 0-level frozen version of itself and $\mathcal{R}_{n+1}$ is an $n+1$-level frozen version of $\mathcal{R}$ if there exists a TRS $\mathcal{R}_n$ such that $\mathcal{R}_{n+1}$ is a $(f, g, m, \underline{fg})$-frozen version of $\mathcal{R}_n$ for some $f, g, m, \underline{fg}$ and $\mathcal{R}_n$ is an $n$-level frozen version of $\mathcal{R}$. We say to freeze

$$f(\bullet_1, \ldots, \bullet_{m-1}, g(\bullet_m, \ldots, \bullet_{m+\mathcal{A}r(g)-1}), \bullet_{m+\mathcal{A}r(g)}, \ldots, \bullet_{\mathcal{A}r(f)+\mathcal{A}r(g)-1})$$

into

$$\underline{fg}(\bullet_1, \ldots, \bullet_{m-1}, \bullet_m, \ldots, \bullet_{m+\mathcal{A}r(g)-1}, \bullet_{m+\mathcal{A}r(g)}, \ldots, \bullet_{\mathcal{A}r(f)+\mathcal{A}r(g)-1})$$

in $\mathcal{R}$ to mean constructing a $(f, g, m, \underline{fg})$-frozen version of $\mathcal{R}$.

### 4.3 Towards Automated Termination Proofs

A straightforward combination of the freezing technique with others can be described as follows. Let $\mathcal{P}roc$ be a procedure which implements some approach to automated termination proofs. Given a TRS $\mathcal{R}$, we can enumerate all the $n$-level frozen versions of $\mathcal{R}$ for some $n$ and use $\mathcal{P}roc$ to decide if one of them is terminating. This approach is impractical when the number of $n$-level frozen versions of $\mathcal{R}$ is too large. We use the following example to demonstrate a way to cope with this problem.

*Example 4.* This example is taken from [4]. Let $\mathcal{R}$ consist of the following rules.

$$\begin{aligned}
&1. && Div2(\emptyset) \to \emptyset \\
&2. && Div2(S(\emptyset)) \to \emptyset \\
&3. && Div2(S(S(x))) \to S(Div2(x)) \\
&4. && LastBit(\emptyset) \to 0 \\
&5. && LastBit(S(\emptyset)) \to 1 \\
&6. && LastBit(S(S(x))) \to LastBit(x) \\
&7. && Conv(\emptyset) \to \epsilon \& 0 \\
&8. && Conv(S(x)) \to Conv(Div2(S(x))) \& LastBit(S(x))
\end{aligned}$$

Suppose that we want to obtain some frozen version $\mathcal{R}_*$ of $\mathcal{R}$ such that $\mathcal{R}_*$ can be proven using some rpo. Then rule $1, 2, 3, 6$ are well-oriented under any (total) precedence relation. Since none of $\epsilon$, 0, 1 and & occur in the left-hand side of any rule, we assign to them the lowest precedence, and the rule $4, 5, 7$ are then well-oriented. In order to orient rule 8, we freeze $Conv(Div2(\bullet_1))$ into $\underline{ConvDiv2}(\bullet_1)$, generating a TRS $\mathcal{R}_1$ consisting rule 1. – 7., and the following ones.

$$\begin{array}{ll} 8'. & Conv(S(x)) \to \underline{ConvDiv2}(S(x))\&LastBit(S(x)) \\ 9. & \underline{ConvDiv2}(S(S(x))) \to Conv(S(Div2(x))) \\ 10. & \underline{ConvDiv2}(S(\emptyset)) \to Conv(\emptyset) \\ 11. & \underline{ConvDiv2}(\emptyset) \to Conv(\emptyset) \end{array}$$

Now rule $8'$, 9, and 10 are well-oriented under the following precedence:

$$S \succ Conv \succ \underline{ConvDiv2}, Div2, LastBit.$$

We then freeze $Conv(\emptyset)$ into $\underline{Conv\emptyset}$, generating a TRS $\mathcal{R}_2$ consisting of rule 1—7, $8'$, 9, 10 and the following ones.

$$11'. \quad \underline{ConvDiv2}(\emptyset) \to \underline{Conv\emptyset} \qquad\qquad 12. \quad \underline{Conv\emptyset} \to \epsilon\&0$$

Then rule $11'$ is well-oriented under $\underline{ConvDiv2} \succ \underline{Conv\emptyset}$, and rule 12 is well-oriented since $\epsilon$, 0, and & have been given the lowest precedence. Hence $\mathcal{R}$ is terminating. Note that $\mathcal{R}_2$ is the $\mathcal{T} \cup \mathcal{S}$ in [4], which shows the termination of $\mathcal{R}$.

## 5 Examples

In this section we present some examples to show the effectiveness of the freezing technique. We say that $\mathcal{R}_1$ is a frozen version of $\mathcal{R}$ if $\mathcal{R}_1$ is a $n$-level frozen version of $\mathcal{R}$ for some $n$.

We will use the recursive path ordering (with status) approach (rpo(s)) *formulated in [19]* to prove the termination of TRSs. Given a function symbol $F$, the status $\tau(F)$ is either $\textcircled{m}$ for multiset status or $(i_1, \ldots, i_{\mathcal{A}r(F)})$, a permutation of $1, \ldots, \mathcal{A}r(F)$, for lexicographic status. If the status of $F$ is not presented, then it is assumed to be $\tau(F) = \textcircled{m}$ .

*Example 5.* This example is taken from [10], where it is proven terminating by dummy elimination. The TRS consists of the following rule

$$x * (y + z) \to (a(x, y) * y) + (x * a(z, x))$$

modulo associativity and commutativity of $+$.

We freeze $a(\bullet_1, \bullet_2) * \bullet_3$ into $\underline{a*}(\bullet_1, \bullet_2, \bullet_3)$ and $\bullet_1 * a(\bullet_2, \bullet_3)$ into $\underline{*a}(\bullet_1, \bullet_2, \bullet_3)$.

$$\begin{array}{c} x * (y + z) \to \underline{a*}(x, y, y) + \underline{*a}(x, z, x) \\ \underline{a*}(x_1, x_2, y + z) \to \underline{a*}(a(x_1, x_2), y, y) + \underline{*a}(a(x_1, x_2), z, a(x_1, x_2)) \end{array}$$

Let $\oplus$ and $\otimes$ be the usual addition and multiplication on positive integers. The following polynomial interpretation proves that the system is terminating modulo associativity and commutativity of $+$.

$$\begin{gathered} x + y = x \oplus y \oplus 2 \qquad a(x,y) = x \oplus y \\ \underline{a*}(x,y,z) = ((x \oplus y) \otimes z) \oplus (z \otimes z \otimes z) \\ \underline{*a}(x,y,z) = x \oplus y \oplus z \qquad x * y = x \otimes y \otimes y \otimes y \end{gathered}$$

*Example 6.* This example is taken from [1], where it is claimed that the example eludes all the techniques in [21,20,16,11], which aim for generating automatic termination proofs for TRSs.

$$\begin{gathered} minus(x,0) \to x \qquad minus(s(x),s(y)) \to minus(x,y) \\ quot(0,s(y)) \to 0 \qquad quot(s(x),s(y)) \to s(quot(minus(x,y),s(y))) \end{gathered}$$

We freeze $quot(minus(\bullet_1,\bullet_2),\bullet_3)$ into $\underline{quotminus}(\bullet_1,\bullet_2,\bullet_3)$.

$$\begin{gathered} minus(x,0) \to x \qquad minus(s(x),s(y)) \to minus(x,y) \\ quot(0,s(y)) \to 0 \qquad quot(s(x),s(y)) \to s(\underline{quotminus}(x,y,s(y))) \\ \underline{quotminus}(x,0,y) \to quot(x,y) \\ \underline{quotminus}(s(x),s(z),y) \to \underline{quotminus}(x,z,y) \end{gathered}$$

The termination of the above TRS can be proven using rpos with:

$$\begin{gathered} \text{quasi precedence: } quot \approx \underline{quotminus} \succ s \\ \text{status: } \tau(quot) = (1,2), \tau(\underline{quotminus}) = (1,3,2) \end{gathered}$$

*Example 7.* (Greatest Common Divisor)

$$\begin{gathered} x - 0 \to x \qquad s(x) - s(y) \to x - y \\ 0 < s(x) \to true \quad x < 0 \to false \quad s(x) < s(y) \to x < y \\ if(true,x,y) \to x \qquad if(false,x,y) \to y \qquad gcd(x,0) \to x \qquad gcd(0,x) \to x \\ gcd(s(x),s(y)) \to if(x < y, gcd(s(x), y - x), gcd(x - y, s(y))) \end{gathered}$$

Note that the left-hand side of the last rule becomes self-embedded in the right-hand side if $y$ gets substituted with $s(x)$. We freeze $\gcd(\bullet_1 - \bullet_2,\bullet_3)$ into $\underline{gcd{-}L}(\bullet_1,\bullet_2,\bullet_3)$ and $\gcd(\bullet_1,\bullet_2 - \bullet_3)$ into $\underline{gcd{-}R}(\bullet_1,\bullet_2,\bullet_3)$, obtaining the following TRS $\mathcal{R}_1$.

$$\begin{gathered} x - 0 \to x \qquad s(x) - s(y) \to x - y \\ 0 < s(x) \to true \quad x < 0 \to false \quad s(x) < s(y) \to x < y \\ if(true,x,y) \to x \qquad if(false,x,y) \to y \qquad gcd(x,0) \to x \qquad gcd(0,x) \to x \\ gcd(s(x),s(y)) \to if(x < y, \underline{gcd{-}R}(s(x),y,x), \underline{gcd{-}L}(x,y,s(y))) \\ \underline{gcd{-}L}(x,y,0) \to x - y \qquad \underline{gcd{-}R}(0,x,y) \to x - y \\ \underline{gcd{-}L}(s(x),s(y),z) \to \underline{gcd{-}L}(x,y,z) \\ \underline{gcd{-}R}(x,s(y),s(z)) \to \underline{gcd{-}R}(x,y,z) \\ \underline{gcd{-}L}(x,0,y) \to gcd(x,y) \qquad \underline{gcd{-}R}(x,y,0) \to gcd(x,y) \end{gathered}$$

The termination of $\mathcal{R}_1$ can be proven using the rpos with:

quasi precedence: $gcd \approx \underline{gcd{-}L} \approx \underline{gcd{-}R} \succ -, <, if \succ true, false$
status: $\tau(gcd) = (1,2), \tau(\underline{gcd{-}L}) = (1,3,2),$ and $\tau(\underline{gcd{-}R}) = (1,2,3)$

We point out that this example is much harder than Example 6 in [20], where $if$, $<$ and $-$ are simply treated as constructors.

*Example 8.* This example is taken from [2], which cannot be proven with a simplification ordering since the last rule is self-embedded:

$$app(nil,k) \to k \qquad app(l,nil) \to l \qquad app(x.l,k) \to x.app(l,k)$$
$$sum(x.nil) \to x.nil \qquad sum(x.(y.l)) \to sum((x+y).l)$$
$$sum(app(l,x.(y.k))) \to sum(app(l,sum(x.(y.k))))$$

We freeze $app(\bullet_1, sum(\bullet_2.\bullet_3))$ into $\underline{appsumdot}(\bullet_1,\bullet_2,\bullet_3)$ and $app(\bullet_1,\bullet_2.nil)$ into $\underline{appdotnil}(\bullet_1,\bullet_2)$:

$$app(nil,k) \to k \qquad app(l,nil) \to l \qquad app(x.l,k) \to x.app(l,k)$$
$$sum(x.nil) \to x.nil \qquad sum(x.(y.l)) \to sum((x+y).l)$$
$$sum(app(l,x.(y.k))) \to sum(\underline{appsumdot}(l,x,y.k))$$
$$\underline{appsumdot}(nil,x,y) \to sum(x.y)$$
$$\underline{appsumdot}(x.y,z,k) \to x.\underline{appsumdot}(y,z,k)$$
$$\underline{appsumdot}(l,x,y.k) \to \underline{appsumdot}(l,x+y,k)$$
$$\underline{appsumdot}(l,x,nil) \to \underline{appdotnil}(l,x)$$
$$\underline{appdotnil}(x.l,y) \to x.\underline{appdotnil}(l,y) \qquad \underline{appdotnil}(nil,x) \to x.nil$$

The termination of the above TRS can be proven by the rpos:

quasi precedence: $app \approx \underline{appsumdot} \succ \underline{appdotnil} \succ . \succ +$
status:
$\tau(app) = (1,2), \tau(\underline{appsumdot}) = (1,3,2), \tau(\underline{appdotnil}) = (1,2), \tau(.) = (2,1)$

The freezing technique *cannot* transform the following example of Kamin and Lévy into any TRS which can be proven terminating using rpos.

*Example 9.* Let $\mathcal{R}$ be the TRS consisting of the following rules.

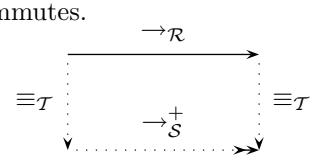
$$p(s(s(x))) \to s(p(s(x))) \quad p(s(0)) \to 0 \quad fac(s(x)) \to fac(p(s(x))) * s(x)$$

Suppose that $\mathcal{R}_1$ is some frozen version of $\mathcal{R}$. If $\mathcal{R}_1$ can be proven terminating with some rpos, then $\mathcal{R}_2 = \mathcal{R}_1 \cup \{p(s(0)) \to s(0)\}$ can also be proven terminating using the same rpos. Clearly this is impossible since $\mathcal{R}_2$ is not terminating. Therefore, none of the frozen versions of $\mathcal{R}$ can be proven using rpos.

## 6 Related Work and Conclusion

Our work closely relates to *transformation orderings* [4], with which we give some comparison.

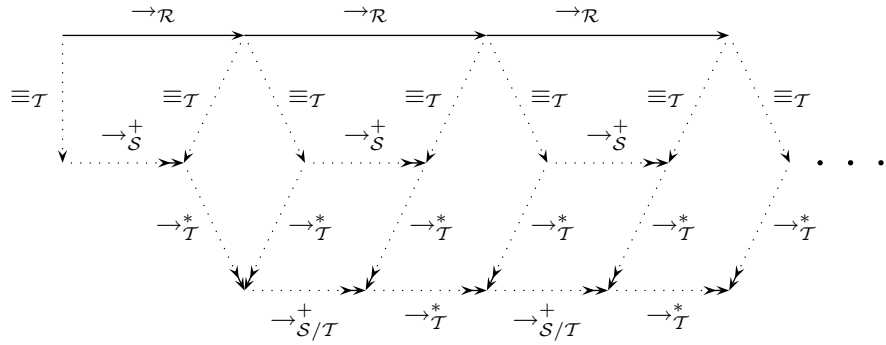
Given two relations $R_1$ and $R_2$, we write $R_1 \cdot R_2$ for the relation $R$ such that $xRz$ if and only if $xR_1y$ and $yR_2z$ for some $y$. Let $\mathcal{R}$ be a TRS for which we intend to find a termination proof. We then try to construct two TRSs $\mathcal{T}$ and $\mathcal{S}$ such that $\to_{\mathcal{R}} \subseteq \equiv_{\mathcal{T}} \cdot \to_{\mathcal{S}}^{+} \cdot \equiv_{\mathcal{T}}$,[2] where $\equiv_{\mathcal{T}} = (\to_{\mathcal{T}} \cup \leftarrow_{\mathcal{T}})^*$. In other words, the following diagram commutes.




Notice the similarity and difference between this and the definition of $\overset{\sim}{\Longrightarrow}$.[3] Also we need to establish the following:

- $\mathcal{T}$ is confluent, and
- $\mathcal{S} \cup \mathcal{T}$ is terminating, and
- $\mathcal{S}$ locally cooperates with $\mathcal{T}$, namely, $\leftarrow_{\mathcal{T}} \cdot \to_{\mathcal{S}} \subseteq \to_{\mathcal{S}/\mathcal{T}}^{*} \cdot {}^{*}\leftarrow_{\mathcal{T}}$.

By Lemma 5 in [3], ${}^{*}\leftarrow_{\mathcal{T}} \cdot \to_{\mathcal{S}}^{*} \subseteq \to_{\mathcal{S}/\mathcal{T}}^{*} \cdot {}^{*}\leftarrow_{\mathcal{T}}$ holds since $\mathcal{S}$ locally cooperates with $\mathcal{T}$, where $\to_{\mathcal{S}/\mathcal{T}}$ stands for $\to_{\mathcal{T}}^{*} \cdot \to_{S} \cdot \to_{\mathcal{T}}^{*}$. Now suppose that there exists an infinite $\to_{\mathcal{R}}$-rewriting sequence. We can then construct an infinite $\to_{\mathcal{S}/\mathcal{T}}$-rewriting sequence as shown below, yielding a contradiction since $\mathcal{S} \cup \mathcal{T}$ is terminating. Therefore, $\to_{\mathcal{R}}$ is terminating.




Note that this is very similar to Theorem 4 in [3]. The transformation ordering (TO) technique is the method which proves the termination of $\mathcal{R}$ by constructing of $\mathcal{T}$ and $\mathcal{S}$.

Although there exist many similarities between the freezing technique and the transformation ordering technique, we point out a significant difference as follows. After freezing $f(\ldots, g(\ldots), \ldots)$ into $\underline{fg}(\ldots)$, it attempting to relate freezing

[2] This condition can be weaken to $\to_{\mathcal{R}} \subseteq \to_{\mathcal{S}/\mathcal{T}}^{+} \cdot {}^{*}\leftarrow_{\mathcal{T}}$.

[3] This makes neither the transformation ordering technique nor the freezing technique encompass the other.

to transformation ordering by setting $\mathcal{T} = \{f(\boldsymbol{x}_{1,m-1}, g(\boldsymbol{y}_{1,\mathcal{A}r(g)}), \boldsymbol{x}_{m+1,\mathcal{A}r(f)}) \to \underline{fg}(\boldsymbol{x}_{1,m-1}, \boldsymbol{y}_{1,\mathcal{A}r(g)}, \boldsymbol{x}_{m+1,\mathcal{A}r(f)})\}$. However, this $\mathcal{T}$ may not be confluent(e.g., in the case where $f = g$). Moreover, if one applies freezing repeatedly, which is often the case in practice, the chance of obtaining a confluent $\mathcal{T}$ diminishes further.

In [20], Steinbach presented an algorithm to generate a transformation ordering for a given $\mathcal{R}$. The freezing technique closely resembles the projection technique in his algorithm. On the other hand, there also exist some significant differences.

1. When implementing the transformation ordering technique, one has to guarantee that the generated $\mathcal{T}$ is confluent, and this could be a rich source for the divergence of the algorithm. For a left-linear TRS, generating a frozen version of $\mathcal{R}$ always terminates.
2. Overall the algorithm in [20] seems to be quite involved and highly heuristic, while the freezing technique is conceptually simple and easily implementable.

In [3], a *quasi-commutation* method is introduced which does not require that the transformer system be confluent, either. However, there is no particular method given in [3] to facilitate the generation of such a transformer system. In contrast, the presentation of the freezing technique, an approach to transforming left-linear TRSs, is precisely the main contribution of this paper.

In summary, we have developed a technique called *freezing*, which can be applied to a left-linear TRS $\mathcal{R}$ to transform it into a family of left-linear TRSs such that the termination of any TRS in this family implies the termination of $\mathcal{R}$. We have shown that the transformation terminates for all (finite) left-linear TRSs. The effectiveness of this technique is demonstrated by many examples, all of which have been verified mechanically.

## 7 Acknowledgement

I gratefully acknowledge some electronic discussion with Joachim Steinbach regarding transformation orderings. I thank the anonymous referees for their detailed comments, which have undoubtedly enhanced the quality of the paper. Also I thank Frank Pfenning, Peter Andrews and Richard Statman for their support and for providing me with such a nice working environment.

## References

1. T. Arts and J. Giesl. Termination of constructor systems. In Proceedings of the Seventh Conference on Rewriting Techniques and Applications, vol. 1103 of Lecture Notes in Computer Science, pp. 63-77, New Brunswick, USA, 1996.
2. T. Arts and J. Giesl. Automatically Proving Termination Where Simplification Orderings Fail. *In Proc. Colloquium on Trees in Algebra and Programming*, vol. 1214 of Lecture Notes in Computer Science, pp. 261-272, Lille, France, 1997.
3. L. Bachmair and N. Dershowitz. Communication, transformation and termination. *In Proceedings of the Eighth CADE*, vol. 230 of Lecture Notes in Computer Science, pp. 52-60, Oxford, July 1986.

4. F. Bellegarde and P. Lescanne. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing*, vol. 1, pp. 79-96, 1990.
5. N. Dershowitz. Orderings for term rewriting systems. *TCS*, 17(3), pp. 279-301, 1982.
6. N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, vol. 3, pp.69-116, 1987.
7. N. Dershowitz and C. Hoot. Natural Termination, *TCS*, 142(2), pp. 179-207, 1995.
8. N. Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In I. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, vol. B, pp. 243-320.
9. N. Dershowitz and Jean-Pierre Jouannaud. Notations for rewriting. *EATCS*, vol. 43, pp. 162-172, 1991.
10. M. Ferreira. Dummy Elimination in Equational Rewriting. In Proceedings of the Seventh Conference on Rewriting Techniques and Applications, vol. 1103 of Lecture Notes in Computer Science, pp. 78-92, New Brunswick, USA, 1996.
11. M. Ferreira and H. Zantema. Dummy Elimination: making termination easier. In Proceedings of the 10th International Conference on Fundamentals of Computation Theory, LNCS 965, Dresden, 1995.
12. J. Giesl. Automated termination proofs with measure function. In *Proceedings of 19th Annual German Conference on AI*, LNAI 981, Bielefeld, 1995.
13. M. Hanus. The integration of functions into logic programming: From theory to practice. In *JLOGP*, vol. 19-20, pp. 583-628.
14. G. Huét and D. Lankford. On the uniform halting problem for term rewriting systems. Technical Report 283, INRIA, Le Chesnay, France, 1978.
15. D. Knuth and P. Bendix. Simple word problems in universal algebras. *Computational Problems in Abstract Algebra*, edited by J. Leech, Pergamon Press, pp. 263-297, 1970.
16. R. Kennaway. Complete term rewrite systems for decimal arithmetic and other total recursive functions. *Presented at the 2nd International Workshop on termination*, La Bresse, 1995
17. D. Lankford. On proving term rewriting systems are noetherian. Tech. Report Memo MTP-3, Louisiana Tech. University, 1979.
18. D. Plaisted. A recursively defined ordering for proving termination of term rewriting systems. Tech. report UIUC DCS-R-78-943, Univ. of Illinois, Urbana, 1978.
19. J. Steinbach. Simplification Orderings: History of Results, *Fundamenta Informaticae*, vol. 24, pp. 47-87, 1995.
20. J. Steinbach. Automatic termination proofs with transformation orderings. *In Proceedings of the Sixth RTA*, vol. 914 of Lecture Notes in Computer Science, pp. 11-25, 1995.
21. H. Zantema. Termination of term rewriting: interpretation and type elimination. *Journal of Symbolic Computation*, vol. 17, pp. 23-50, 1994.
22. H. Zantema. Termination of Term Rewriting by Semantic Labelling, *Fundamenta Informaticae*, vol. 24, pp 89-105, 1995.