Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 237 (2009) 3–21

www.elsevier.com/locate/entcs

A Transformational Approach to Prove Outermost Termination Automatically

Matthias Raffelsieper[1]

Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Hans Zantema[2]

Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

Abstract

We present transformations from a generalized form of left-linear TRSs, called *quasi left-linear* TRSs, to TRSs such that outermost termination of the original TRS can be concluded from termination of the transformed TRS. In this way we can apply state-of-the-art termination tools for automatically proving outermost termination of any given quasi left-linear TRS. Experiments show that this works well for non-trivial examples, some of which could not be automatically proven outermost terminating before. Therefore, our approach substantially increases the class of systems that can be shown outermost terminating automatically.

1 Introduction

A lot of work has been done on automatically proving termination and innermost termination. However, also termination with respect to the outermost strategy makes sense. For instance, this is the standard strategy in the functional programming language Haskell [11], and it can be specified in CafeOBJ [4] and Maude [2]. We will focus on the most general variant of the outermost strategy: reducing a redex is always allowed if it is not a proper subterm of another redex. Termination with respect to this strategy we shortly call *outermost termination*. This is different from the approaches for proving termination of Haskell presented in [7,16], which feature proving termination for a specific set of terms (ground instantiations of a

[1] Email: M.Raffelsieper@tue.nl

[2] Email: H.Zantema@tue.nl

doi:10.1016/j.entcs.2009.03.032

so-called *start term*), while we only do it for all terms. Another difference is that the language Haskell does not allow overlapping rules, i.e., there is at most one rule applicable for every term.

Let's consider the following example. The infinite list $0 : 1 : 2 : 3 : \cdots$ can be generated by applying the non-terminating rule

$$\mathsf{from}(x) \rightarrow x : \mathsf{from}(\mathsf{s}(x))$$

to the start term $\mathsf{from}(0)$. If we want to check for overflow, e.g., numbers should not be n or higher, we could add the rule

$$\mathsf{s}^n(x) : xs \rightarrow \mathsf{overflow}.$$

Now for sure the TRS remains non-terminating since it still contains the first non-terminating rule. But we expect the combined system to be outermost terminating. This is the kind of examples for which we want to prove outermost termination automatically.

Until now Cariboo [9] was the only tool having facilities for proving outermost termination. Its approach is a stand alone one, while our goal is to make use of the huge effort of the last years to improve the power of termination tools. For making use of the impressive power of termination tools, the natural approach is to make a transformation from TRSs to TRSs such that the modified termination property (in our case outermost termination) of the original TRS can be concluded from termination of the transformed TRS. In the past, a similar approach was successfully applied to context-sensitive termination [5] and liveness problems [8].

The Termination Problem DataBase (TPDB) [15] already contains 6 outermost examples that really require a consideration of the outermost strategy. If no strategy is regarded, then all of these examples are non-terminating. For these examples, both Cariboo and our presented transformation together with a termination prover for term rewrite systems without a strategy can prove outermost termination. As will be shown later, using the presented transformation we can prove outermost termination for systems where Cariboo fails to do so. Therefore, this approach increases the number of term rewrite systems that can be shown outermost terminating automatically. However, it is not the case that it supersedes Cariboo: There are also examples where Cariboo succeeds while the transformed TRS cannot be shown terminating by any of the termination provers that we tried.

The presented approach deals with ground outermost termination. We will see that when fixing the signature there may be a difference between outermost termination and ground outermost termination, but by adding fresh constants there is no difference any more. Therefore it is not a restriction to focus on ground outermost termination.

The crucial ingredient of our transformation is *anti-matching*: for L being the left-hand sides of the given TRS, we need a set S_L such that any term matches with a term in S_L if and only if it does not match with a term in L. It turns out that if all terms in L are linear, then a finite set S_L satisfying this requirement can easily be constructed, while there are sets L containing non-linear terms such that every set S_L satisfying this property is infinite. That's why we restrict to the class of *quasi left-linear* TRSs, which are all TRSs where every left-hand side is an instance of a linear left-hand side. Clearly, this class also includes all left-linear TRSs.

Based on this anti-matching we give a straightforward transformation T_1 such that every infinite outermost reduction with respect to any quasi left-linear TRS R gives rise to an infinite $T_1(R)$-reduction. We experimented with several variants of the basic transformation and chose the most powerful one in our final definition of the transformation.

Using anti-matching we present two other transformations T_2 and T_3, based on irreducible contexts. Since these contexts can only capture the first level of nesting, this approach is inherently incomplete. However, experiments show that for some examples these transformations are successful, while the transformation T_1 gives rise to a problem that is too hard for existing termination tools.

This paper is structured as follows: After introducing the used notations in Section 2, we present our basic transformation T_1 and prove soundness in Section 3. Thereafter, we present our alternative transformations T_2 and T_3 using contexts in Section 4. In these two sections, we assume that we can construct a set of terms that match those terms not being matched by a left-hand side. This problem of anti-matching is treated in Section 5, which proves that our transformations can be applied automatically for quasi left-linear TRSs. In Section 6 we give a short description of our implementation of the transformations and present a number of examples. We conclude in Section 7.

2 Preliminaries

This section shall briefly introduce the notations used in this paper. For an introduction of term rewriting see for example [1,17].

We consider the set $\mathcal{T}(\Sigma, \mathcal{V})$ of all *terms* over a set $\mathcal{V}$ of *variables* and a finite, non-empty set Σ of *function symbols*, called *signature*, where each $f \in \Sigma$ is associated with a natural number called its *arity*. If the arity of $f \in \Sigma$ is $n \in \mathbb{N}$, then we denote this by $\text{arity}(f) = n$. Instead of $\mathcal{T}(\Sigma, \emptyset)$ we also write $\mathcal{T}(\Sigma)$, which we call the set of *ground terms*. For a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ we write $\mathcal{V}(t)$ to denote the set of variables occurring in t. For a non-variable term $t = f(t_1, \ldots, t_n)$ we say that f is the *root* of t, denoted by $\mathsf{root}(t)$. A term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is called *linear* if every variable occurs at most once in t; we write $\mathcal{T}_{\text{lin}}(\Sigma, \mathcal{V})$ for the set of all linear terms in $\mathcal{T}(\Sigma, \mathcal{V})$.

A *position* of a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is a sequence of natural numbers, the set of all positions of a term t is denoted $Pos(t)$. The empty sequence is denoted as ϵ. Such a position $\pi \in Pos(t)$ identifies a *subterm* of t, which is written $t|_\pi$. The term that we get from replacing the subterm $t|_\pi$ by another term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ is denoted $t[s]_\pi$.

A *substitution* $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ is written as $\sigma = \{x_1 := t_1, \ldots, x_m := t_m\}$, which denotes the mapping $\sigma(x_i) = t_i$ and $\sigma(x) = x$ for all $x \neq x_i$ and $1 \leq i \leq m$. The set of all substitutions over Σ and $\mathcal{V}$ is denoted as $\text{SUB}(\Sigma, \mathcal{V})$. The application of a substitution $\sigma \in \text{SUB}(\Sigma, \mathcal{V})$ to a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is denoted $t\sigma$ and replaces all variables by their corresponding terms. Such a term $t\sigma$ is called an *instance* of t. Two terms $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$ are said to *unify*, if a *unifier* $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ exists such that $s\sigma = t\sigma$. A term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ is said to *match* a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$, if a substitution $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ exists such that $s\sigma = t$.

A standard property relating linearity and unification is the following.

Lemma 2.1 *Let $t, u \in \mathcal{T}_{\text{lin}}(\Sigma, \mathcal{V})$ be two linear terms with $\mathcal{V}(t) \cap \mathcal{V}(u) = \emptyset$ that do not unify. Then there is a position $\pi \in Pos(t) \cap Pos(u)$ such that $t|_\pi$ and $u|_\pi$ are no variables and $\mathsf{root}(t|_\pi) \neq \mathsf{root}(u|_\pi)$.*

Proof (Sketch). We refer to [17, Section 7.7], where the standard Martelli-Montanari unification algorithm is given. For linear terms $t, u \in \mathcal{T}_{\text{lin}}(\Sigma, \mathcal{V})$ with $\mathcal{V}(t) \cap \mathcal{V}(u) = \emptyset$ we have as invariant of the algorithm that every variable occurs at most once. Thus, the only way to get the result **fail** is by having two terms that have different root symbols. □

A *Term Rewrite System (TRS)* is a set $R \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$, where instead of $(\ell, r) \in R$ we write $\ell \to r \in R$.[3] The set $\mathrm{lhs}(R)$ of left-hand sides of a TRS R is defined as $\mathrm{lhs}(R) = \{\ell \mid \ell \to r \in R\}$. A TRS is called *left-linear* if ℓ is linear for all $\ell \in \mathrm{lhs}(R)$ and we call it *quasi left-linear* if every $\ell \in \mathrm{lhs}(R)$ is an instance of a linear $\ell' \in \mathrm{lhs}(R)$. A term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ *rewrites* to a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ with a rule $\ell \to r \in R$ at a position $\pi \in Pos(s)$, denoted by $s \to_{\ell \to r, \pi} t$, iff there exists a substitution $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ such that $s|_\pi = \ell\sigma$ and $t = s[r\sigma]_\pi$. The term $s|_\pi$ is called *redex*. Instead of $s \to_{\ell \to r, \pi} t$ we also write $s \to_{R,\pi} t$, $s \to_R t$, $s \to_\pi t$, or $s \to t$ if the term rewrite system R and/or the position π are clear from the context. If for a term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ and a position $\pi \in Pos(s)$ we have for all $t \in \mathcal{T}(\Sigma, \mathcal{V})$ that $s \not\to_{R,\pi} t$ holds, then we also write $s \not\to_{R,\pi}$ or $s \not\to_\pi$.

A term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ *outermost rewrites* to a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ with a rule $\ell \to r \in R$ at a position $\pi \in Pos(s)$, denoted $s \xrightarrow{o}_{\ell \to r, \pi} t$, iff $s \to_{\ell \to r, \pi} t$ and for all positions $\pi' \in Pos(s)$ with $\pi = \pi'\pi''$ for some $\pi'' \in \mathbb{N}^*$ with $\pi'' \neq \epsilon$ we have that $s \not\to_{R,\pi'}$.

A TRS R is called *terminating* (*outermost terminating*), iff there is no infinite sequence $t_1, t_2, t_3, \ldots \in \mathcal{T}(\Sigma, \mathcal{V})$ of terms with $t_i \to_R t_{i+1}$ ($t_i \xrightarrow{o}_R t_{i+1}$) for all $i \in \mathbb{N}$. A TRS R is called *ground terminating* (*outermost ground terminating*), iff there is no infinite sequence $t_1, t_2, t_3 \ldots \in \mathcal{T}(\Sigma)$ of ground terms such that $t_i \to_R t_{i+1}$ ($t_i \xrightarrow{o}_R t_{i+1}$) for all $i \in \mathbb{N}$.

The following example shows that outermost termination for arbitrary terms may differ from outermost ground termination.

Example 2.2 Consider the following term rewrite system R over the signature $\Sigma = \{\mathsf{f}, \mathsf{a}, \mathsf{b}\}$:

$$R = \begin{cases} \mathsf{f}(\mathsf{a}, x) & \to & \mathsf{a} & \qquad \mathsf{f}(x, \mathsf{a}) & \to & \mathsf{f}(x, \mathsf{b}) \\ \mathsf{f}(\mathsf{b}, x) & \to & \mathsf{a} & \qquad \mathsf{b} & \to & \mathsf{a} \\ \mathsf{f}(\mathsf{f}(x, y), z) & \to & \mathsf{a} & & & \end{cases}$$

[3] Note that the standard restrictions $\ell \notin \mathcal{V}$ and $\mathcal{V}(r) \subseteq \mathcal{V}(\ell)$ are not essential for our results.

For arbitrary terms we have the infinite outermost reduction

$$\mathsf{f}(x, \mathsf{a}) \to \mathsf{f}(x, \mathsf{b}) \to \mathsf{f}(x, \mathsf{a}) \to \cdots$$

However, when instantiating x by any arbitrary ground term $t \in \mathcal{T}(\Sigma)$, then one of the three rules on the left is applicable at the root position. Hence, the reduction $\mathsf{f}(t, \mathsf{b}) \to \mathsf{f}(t, \mathsf{a})$ is no longer an outermost reduction. In fact, the above term rewrite system is outermost ground terminating, as we will show later.

However, this difference only occurs when fixing the signature. It is easy to see that by replacing variables in any infinite outermost reduction by fresh constants, the result is an infinite outermost ground reduction. For quasi left-linear TRSs adding one fresh constant suffices. Hence, we may and shall restrict ourselves to outermost ground termination.

Finally, we want to remark on the restrictions mentioned above in Footnote 3: If we have a rule $x \to r \in R$ for some $x \in \mathcal{V}$ then clearly the TRS R is not outermost terminating, since this rule can always be applied at the root position. However, the second restriction, $\mathcal{V}(r) \subseteq \mathcal{V}(\ell)$, does not directly lead to non-outermost termination, a counterexample is the outermost terminating TRS $R = \{\mathsf{a} \to \mathsf{f}(x), \mathsf{f}(x) \to \mathsf{b}\}$.

3 The first Transformation

The idea of the first transformation is to only allow a reduction when a certain control symbol down marks the current redex. After having reduced a term, the control symbol is replaced by another control symbol up that is moved outwards. Only when the root of the term is encountered, then the control symbol is replaced by the down symbol again. In order to find the next outermost redex, the symbol down may only descend into subterms when no left-hand side is applicable to the term. For this purpose, we need a set S_L such that its elements match exactly those terms that are not matched by a left-hand side. Such a set S_L is called *anti-matching*, which is defined below.

Definition 3.1 A set $S_L \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ is called *anti-matching* w.r.t. a set $L \subseteq \mathcal{T}(\Sigma, \mathcal{V})$, iff the following holds for all ground terms $t \in \mathcal{T}(\Sigma)$:

$$\nexists \ell \in L, \tau \in \mathrm{SUB}(\Sigma, \mathcal{V}) : t = \ell\tau \iff \exists s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V}) : t = s\sigma$$

Using such an anti-matching set, we can now formally define our transformation that implements the idea outlined above.

Definition 3.2 Let R be a TRS over a signature Σ and let $S_L \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ be an anti-matching set w.r.t. $L = \mathrm{lhs}(R)$.

Choose four fresh unary symbols $\mathsf{top}, \mathsf{up}, \mathsf{down}, \mathsf{block} \notin \Sigma$, and let $\Sigma^\natural = \{f^\natural \mid f \in \Sigma, \mathrm{arity}(f) > 0\}$ be such that $\Sigma \cap \Sigma^\natural = \emptyset$. The TRS $T_1(R)$ over the signature $\Sigma \cup \Sigma^\natural \cup \{\mathsf{top}, \mathsf{up}, \mathsf{down}, \mathsf{block}\}$ is defined to consist of the following rules:

- $\mathsf{down}(\ell) \to \mathsf{up}(r)$, for all rules $\ell \to r$ of R;

- $\mathsf{top}(\mathsf{up}(x)) \rightarrow \mathsf{top}(\mathsf{down}(x))$;
- $\mathsf{down}(f(t_1,\ldots,t_n)) \rightarrow f^\natural(\mathsf{block}(t_1),\ldots,\mathsf{down}(t_i),\ldots,\mathsf{block}(t_n))$, for all $f(t_1,\ldots,t_n) \in S_L$ and all $i \in \{1,\ldots,n\}$;
- $f^\natural(\mathsf{block}(x_1),\ldots,\mathsf{up}(x_i),\ldots,\mathsf{block}(x_n)) \rightarrow \mathsf{up}(f(x_1,\ldots,x_n))$, for all $f \in \Sigma$ and all $i \in \{1,\ldots,n\}$, where arity$(f) = n$ and $x_1,\ldots,x_n$ are distinct variables.

For an infinite TRS R, we clearly have that $T_1(R)$ is infinite, too. The TRS $T_1(R)$ can also become infinite for a TRS R that is not quasi left-linear, since then an anti-matching set S_L might be infinite. This is demonstrated in Section 5. For quasi left-linear TRSs however, we will prove the following theorem by giving a possible construction of a finite anti-matching set S_L.

Theorem 3.3 *For a quasi left-linear TRS R, there exists a finite, computable, and (up to variable renaming) unique set $S_L \subseteq \mathcal{T}_{\text{lin}}(\Sigma,\mathcal{V})$ that is anti-matching w.r.t. $L = \text{lhs}(R)$.*

The proof of this theorem is given in Section 5, which deals with the general problem of anti-matching. However, soundness of the transformation does not depend on the finiteness of S_L.

In the transformation, we introduce the already mentioned symbols down and up to control the position of the next redex. The symbol top is used to denote the root position of the term, where the search of the next redex has to turn downwards again. The purpose of the symbol block is to disallow evaluations where an up symbol appears at the root position of a term without having applied a rule of the form $\mathsf{down}(\ell) \rightarrow \mathsf{up}(r)$. Furthermore, we create a new marked symbol $f^\natural$ for every function symbol f of the original rewrite system. This makes termination proofs easier, since it can be distinguished whether a symbol is above or below one of the control symbols.

An outermost rewrite step can be modelled by a sequence of steps in the transformed system. This is shown in the following lemma.

Lemma 3.4 *Let R be a TRS over a signature Σ, let $u, v \in \mathcal{T}(\Sigma)$.*
If $u \xrightarrow{o}_R v$, then $\mathsf{down}(u) \rightarrow^+_{T_1(R)} \mathsf{up}(v)$.

Proof. Let $u, v \in \mathcal{T}(\Sigma)$ be two ground terms, let $u \xrightarrow{o}_{\ell\rightarrow r,\pi} v$ for some rule $\ell \rightarrow r \in R$ and some position $\pi \in Pos(u)$. Induction is done over the length of π.

In case $\pi = \epsilon$, we directly have the rule $\mathsf{down}(\ell) \rightarrow \mathsf{up}(r) \in T_1(R)$, which shows the desired property.

Otherwise, let $\pi = i\,\pi'$ for some $\pi' \in Pos(u|_i)$ and let $u = f(u_1,\ldots,u_n)$. Hence, $u \not\rightarrow_\epsilon$ and $v = f(u_1,\ldots,u_{i-1},v_i,u_{i+1},\ldots,u_n)$ for some $v_i \in \mathcal{T}(\Sigma)$. Since $u \not\rightarrow_\epsilon$, we have $u \neq \ell'\tau$ for all $\tau \in \text{SUB}(\Sigma,\mathcal{V})$ and all $\ell' \in \text{lhs}(R)$. Hence, there is a term $s \in S_L$ in an anti-matching set S_L w.r.t. $L = \text{lhs}(R)$ and a substitution $\sigma \in \text{SUB}(\Sigma,\mathcal{V})$ such that $s\sigma = u$. Let $s = f(s_1,\ldots,s_n)$. Then a rule $\mathsf{down}(f(s_1,\ldots,s_n)) \rightarrow f^\natural(\mathsf{block}(s_1),\ldots,\mathsf{down}(s_i),\ldots,\mathsf{block}(s_n)) \in T_1(R)$ exists that is applicable to the term $\mathsf{down}(u)$ and therefore gives the reduction $\mathsf{down}(u) = \mathsf{down}(f(u_1,\ldots,u_n)) \rightarrow f^\natural(\mathsf{block}(u_1),\ldots,\mathsf{down}(u_i),\ldots,\mathsf{block}(u_n))$. For the subterm u_i we have $u_i \xrightarrow{o}_{\ell\to r,\pi'} v_i$. Here, the induction hypothesis is applicable and yields a reduction $\mathsf{down}(u_i) \to^+ \mathsf{up}(v_i)$. When applying this together with the rule $f^\natural(\mathsf{block}(x_1), \ldots, \mathsf{up}(x_i), \ldots, \mathsf{block}(x_n)) \to \mathsf{up}(f(x_1, \ldots, x_n)) \in T_1(R)$ the desired result can be shown:

$$\begin{aligned} \mathsf{down}(u) = \mathsf{down}(f(u_1, \ldots, u_n)) &\to \quad f^\natural(\mathsf{block}(u_1), \ldots, \mathsf{down}(u_i), \ldots, \mathsf{block}(u_n)) \\ &\to^+ \ f^\natural(\mathsf{block}(u_1), \ldots, \mathsf{up}(v_i), \ldots, \mathsf{block}(u_n)) \\ &\to \quad \mathsf{up}(f(u_1, \ldots, u_{i-1}, v_i, u_{i+1}, \ldots, u_n)) = \mathsf{up}(v) \end{aligned}$$

□

Using the above lemma we can prove the main theorem which shows that the presented transformation is sound, i.e., from the termination of the transformed TRS the outermost ground termination of the original TRS can be concluded.

Theorem 3.5 *Let R be a TRS over a signature Σ for which $T_1(R)$ is terminating. Then R is outermost ground terminating.*

Proof. Assume R is not outermost ground terminating. Then there is an infinite outermost reduction $t_1 \xrightarrow{o}_R t_2 \xrightarrow{o}_R \ldots$ for some ground terms $t_1, t_2, \ldots \in \mathcal{T}(\Sigma)$. For each $t_i \xrightarrow{o}_R t_{i+1}$ we have that $\mathsf{down}(t_i) \to^+_{T_1(R)} \mathsf{up}(t_{i+1})$ by Lemma 3.4. Due to the rule $\mathsf{top}(\mathsf{up}(x)) \to \mathsf{top}(\mathsf{down}(x))$ we obtain an infinite reduction in the transformed system $T_1(R)$,

$$\mathsf{top}(\mathsf{down}(t_1)) \to^+_{T_1(R)} \mathsf{top}(\mathsf{up}(t_2)) \to_{T_1(R)} \mathsf{top}(\mathsf{down}(t_2)) \to^+_{T_1(R)} \cdots,$$

contradicting termination of $T_1(R)$. □

Let us now remark on the situation when (arbitrary) outermost termination shall be considered, not just outermost ground termination. For a quasi left-linear TRS R over the signature Σ we create a new TRS R' which has the same rules as R, but now is defined to be over the signature $\Sigma' = \Sigma \cup \{c\}$, where $c \notin \Sigma$ is a fresh constant (i.e., it has arity 0). Then an infinite reduction $t_1 \xrightarrow{o}_R t_2 \xrightarrow{o}_R \ldots$ for some terms $t_1, t_2, \ldots \in \mathcal{T}(\Sigma, \mathcal{V})$ implies that $t_1\sigma_1 \xrightarrow{o}_{R'} t_2\sigma_2 \xrightarrow{o}_{R'} \ldots$ is an infinite reduction of ground terms $t_i\sigma_i \in \mathcal{T}(\Sigma)$ for substitutions $\sigma_i = \{x := c \mid x \in \mathcal{V}(t_i)\}$ for $i \in \mathbb{N}$. This holds, since no left-hand side of the rewrite system R' matches a subterm of $t_i\sigma_i$ which R does not match, because no left-hand side of R' contains the constant c. In the other direction, one can replace a symbol c in an infinite reduction w.r.t. R' by a fresh variable, giving an infinite reduction w.r.t. R. Therefore, we have that the term rewrite system R is outermost terminating, iff the term rewrite system R' is outermost ground terminating. Such a TRS R' can then be handled by our transformation to show outermost termination of R.

4 Other Transformations based on Contexts

There are examples for which the transformed TRS $T_1(R)$ cannot be proven terminating automatically. However, for some of these examples the transformations presented in the following are successful. These transformations do not use symbols down and up to find the next redex, but only allow to rewrite a redex when a so-called *anti-matching* context is found. This notion of anti-matching contexts is given in the following definition.

Definition 4.1 For a set L of terms, a set $C_L \subseteq \mathcal{T}(\Sigma, \mathcal{V} \uplus \{\Box\})$ is called a set of *anti-matching contexts* w.r.t. L, iff

- $\Box \notin C_L$,
- $\Box$ occurs exactly once in every term $C \in C_L$, and
- If for some $t \in \mathcal{T}(\Sigma)$ we have $t|_\pi = \ell\sigma$ for some $\ell \in L$, some $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$, and some $\pi \in Pos(t)$, and $t \neq \ell'\sigma'$ holds for all $\ell' \in L$ and all $\sigma' \in \mathrm{SUB}(\Sigma, \mathcal{V})$, then there is a $C \in C_L$ and a $\tau \in \mathrm{SUB}(\Sigma, \mathcal{V} \uplus \{\Box\})$ such that $t = C\tau$ and for $\pi' \in Pos(C)$ with $C|_{\pi'} = \Box$ we have that $\pi = \pi'\,\pi''$ for some $\pi'' \in \mathbb{N}^*$.

Here, the third requirement is similar to the requirements for an anti-matching set S_L. Thus, we can define the set $C'_L = \{s[\Box]_i \mid s \in S_L, 1 \leq i \leq \mathrm{arity}(\mathsf{root}(s))\}$, for which all of the above requirements are fulfilled, as can easily be checked. Using such a set of contexts, we now define another transformation T_2.

Definition 4.2 For a TRS R over a signature Σ let C_L be a set of anti-matching contexts w.r.t. $L = \mathrm{lhs}(R)$. We define a TRS $T_2(R)$ over the signature $\Sigma \uplus \{\mathsf{top}\}$ that consists of the following rules:

- $\mathsf{top}(\ell) \to \mathsf{top}(r)$ for all $\ell \to r \in R$, and
- $C[\ell] \to C[r]$ for all $\ell \to r \in R$ and all $C \in C_L$.

It can easily be shown that T_2 is a sound transformation, but we omit this for space reasons. When using the set C'_L, then the transformed TRS $T_2(R)$ of a TRS R is finite if R is both finite and quasi left-linear. The first requirement is obviously needed. The second requirement stems from the construction of C'_L which relies on the set S_L. Here, we can observe from Theorem 3.3 that this set, and hence also the construction, can only be guaranteed to be finite if R is quasi left-linear.

One might wonder why in the construction of the set C'_L the "hole" $\Box$ is always introduced at the first argument level, and does not simply replace a variable of the term. Why this latter idea is not sound is illustrated in the next example.

Example 4.3 We consider the below TRS R over the signature $\Sigma = \{\mathsf{f}, \mathsf{b}\}$:

$$R = \begin{cases} \mathsf{b} \to \mathsf{f}(\mathsf{f}(\mathsf{b})) & \mathsf{f}(\mathsf{f}(\mathsf{f}(x))) \to \mathsf{b} \\ \mathsf{f}(\mathsf{b}) \to \mathsf{b} & \end{cases}$$

For $L = \mathrm{lhs}(R)$ we have that $S_L = \{\mathsf{f}(\mathsf{f}(\mathsf{b}))\}$, i.e., there are no variables in the only term contained in S_L. Thus, when choosing $C_L = \emptyset$ then the only rules to be considered are of the form $\mathsf{top}(\ell) \to \mathsf{top}(r)$. The TRS consisting of only these rules can be shown terminating. However, there exists the following infinite outermost ground reduction, which shows that this choice of C_L is unsound:

$$\mathsf{f}(\mathsf{f}(\mathsf{b})) \xrightarrow{o}_R \mathsf{f}(\mathsf{b}) \xrightarrow{o}_R \mathsf{b} \xrightarrow{o}_R \mathsf{f}(\mathsf{f}(\mathsf{b})) \xrightarrow{o}_R \dots$$

It should be noted that $C_L = \emptyset$ is not a set of anti-matching contexts w.r.t. L. This holds because the term $\mathsf{f}(\mathsf{f}(\mathsf{b}))$ is irreducible at root position but is reducible at position $\pi = 1$. Hence, we have a violation of the third requirement of Definition 4.1.

The next example shows that the transformation T_2 is incomplete, regardless of the set C_L of anti-matching contexts.

Example 4.4 Consider the following TRS R:

$$R = \begin{cases} \mathsf{f}(\mathsf{h}(x)) & \to & \mathsf{f}(\mathsf{i}(x)) & \qquad \mathsf{i}(x) & \to & \mathsf{h}(x) \\ \mathsf{f}(\mathsf{i}(x)) & \to & \mathsf{a} \end{cases}$$

When this TRS is transformed using the transformation T_2, then the context $\mathsf{f}(\Box)$ must be considered, since for example the term $\mathsf{f}(\mathsf{f}(\mathsf{h}(x)))$ is not matched by any left-hand side of R, but it is reducible at the position $\pi = 1$. Thus, we have the rules $\mathsf{f}(\mathsf{f}(\mathsf{h}(x))) \to \mathsf{f}(\mathsf{f}(\mathsf{i}(x))) \in T_2(R)$ and $\mathsf{f}(\mathsf{i}(x)) \to \mathsf{f}(\mathsf{h}(x)) \in T_2(R)$. For these two rules we get the following infinite reduction:

$$\mathsf{f}(\mathsf{f}(\mathsf{h}(x))) \to \mathsf{f}(\mathsf{f}(\mathsf{i}(x))) \to \mathsf{f}(\mathsf{f}(\mathsf{h}(x))) \to \dots$$

However, the TRS R is outermost ground terminating, as can be shown using the transformation T_1 of the previous section.

This example failed because we inserted the rule $\mathsf{i}(x) \to \mathsf{h}(x)$ into the context $\mathsf{f}(\Box)$, but such a reduction will never take place due to the left-hand side $\mathsf{f}(\mathsf{i}(x))$. An improvement of this transformation does not insert all rules into all contexts, but only those where an outermost reduction could possibly take place. Such an improvement is given in below, which again uses an anti-matching set S_L.

Definition 4.5 Let R be a TRS over a signature Σ, let $S_L \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ be a set of anti-matching terms w.r.t. $L = \text{lhs}(R)$. We define a TRS $T_3(R)$ over the signature $\Sigma \uplus \{\mathsf{top}\}$ which contains the following rules:

- $\mathsf{top}(\ell) \to \mathsf{top}(r)$ for all $\ell \to r \in R$, and
- $s[\ell]_i\mu \to s[r]_i\mu$ for all $s \in S_L$, all $1 \le i \le \text{arity}(\mathsf{root}(s))$, and all $\ell \to r \in R$ for which s and ℓ are variable disjoint and $s|_i$ unifies with ℓ with the most general unifier μ.

The above transformation T_3 is still sound, i.e., from termination of a TRS $T_3(R)$ one can conclude outermost ground termination of R, but we again have to omit this proof for space reasons. Also for this transformation it can be observed that a finite TRS $T_3(R)$ is only achieved if the TRS R is finite and quasi left-linear.

Next we want to compare the transformations T_2 and T_3. For this purpose, we consider Example 4.4 again, which was the motivation for the transformation T_3.

Example 4.6 (Transformed TRS $T_3(R)$ for Example 4.4)

For this example we have that $S_L = \{\mathsf{h}(x), \mathsf{a}, \mathsf{f}(\mathsf{a}), \mathsf{f}(\mathsf{f}(x))\}$.

$$T_3(R) = \left\{ \begin{array}{lcllcl} \mathsf{top}(\mathsf{f}(\mathsf{h}(x))) & \to & \mathsf{top}(\mathsf{f}(\mathsf{i}(x))) & \quad \mathsf{h}(\mathsf{f}(\mathsf{h}(x))) & \to & \mathsf{h}(\mathsf{f}(\mathsf{i}(x))) \\ \mathsf{top}(\mathsf{f}(\mathsf{i}(x))) & \to & \mathsf{top}(\mathsf{a}) & \quad \mathsf{h}(\mathsf{f}(\mathsf{i}(x))) & \to & \mathsf{h}(\mathsf{a}) \\ \mathsf{top}(\mathsf{i}(x)) & \to & \mathsf{top}(\mathsf{h}(x)) & \quad \mathsf{h}(\mathsf{i}(x)) & \to & \mathsf{h}(\mathsf{h}(x)) \\ \mathsf{f}(\mathsf{f}(\mathsf{h}(x))) & \to & \mathsf{f}(\mathsf{f}(\mathsf{i}(x))) & \quad \mathsf{f}(\mathsf{f}(\mathsf{i}(x))) & \to & \mathsf{f}(\mathsf{a}) \end{array} \right.$$

As can be seen above, the rule $\mathsf{f}(\mathsf{i}(x)) \to \mathsf{f}(\mathsf{h}(x))$ is not contained in the transformed TRS $T_3(R)$, which was the cause of $T_2(R)$ being non-terminating in Example 4.4. And indeed, the TRS $T_3(R)$ can be shown terminating automatically.

However, this improved transformation is still not complete, as demonstrated in the following example.

Example 4.7

$$R = \left\{ \begin{array}{lcl} \mathsf{from}(x) & \to & x : (\mathsf{nil} : \mathsf{from}(\mathsf{s}(x))) \\ \mathsf{s}(x) : xs & \to & \mathsf{nil} \end{array} \right.$$

Outermost ground termination of this example can again be shown automatically, using the transformation T_1. However, we have that $\mathsf{nil} : xs \in S_L$. Thus, we have the rule $\mathsf{nil} : \mathsf{from}(x) \to \mathsf{nil} : (x : \underline{(\mathsf{nil} : \mathsf{from}(\mathsf{s}(x)))}) \in T_3(R)$. For this rule we can see that the underlined part of the right-hand side is again matched by the left-hand side of that rule, leading to an infinite reduction.

5 Anti-matching

We consider a term rewrite system R and a set L matching all terms that can be rewritten by R, for example $L = \mathrm{lhs}(R)$. For our transformations we have to find an anti-matching set S_L that matches the terms which cannot be rewritten by R. Clearly, this is only depending on the left-hand sides of R. One can imagine that there are several possible sets that satisfy this condition. Our goal is to select the smallest such set and to be able to construct it finitely when this is possible.

In this section we consider the general problem of finding an anti-matching set. Only at the end of this section we restrict ourselves to linear terms, for which it will be shown that the minimal anti-matching set is finite and can be computed.

The problem of finding a set of terms that describe the complement of a set of terms is similar to the problem considered by Lassez and Marriot [14], where an explicit representation of a set is being searched that is described using counter examples. But their focus is on machine learning, therefore it is hard to directly apply their results. We also want to mention the concept of *anti-patterns* as introduced in [12]. This is more general since it allows to introduce negation of patterns at any position in a term, while we are only interested in the negation of a complete pattern. However, this work is not applicable here, since we want a representation of a set that does not match a given term, while an anti-pattern matching problem is to decide whether an anti-pattern matches a single ground term.

The anti-matching problem is similar to the constrained substitutions used for outermost narrowing in [9]. A constrained substitution is a substitution together with disequations that constrain the outermost narrowing of subterms to those where no redex exists at a prefix position. An anti-matching set can be seen as explicit solution of such disequations. This is not required for the outermost narrowing in [9], where the implicit formulation is sufficient but must be considered throughout the proof.

Below, we first define a set S'_L of terms that satisfy the desired property. This set is usually infinite and contains quite a number of redundant terms, i.e., terms that are already matched by other terms contained in S'_L. Thus, we define another set S_L that consists only of the minimal elements of S'_L w.r.t. an order that expresses whether one term matches the other.

Definition 5.1 Let $L \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ be a set of terms. On terms we define the preorder $\leq$ by

$$t \leq u \iff \exists \sigma : t\sigma = u$$

which induces the definition of its strict part to be

$$t < u \iff t \leq u \wedge \neg(u \leq t).$$

Now S_L is defined to be the set of minimal elements of the set of terms that do not unify with elements of L, i.e.,

$$\begin{aligned} S'_L &= \{t \in \mathcal{T}(\Sigma, \mathcal{V}) \mid \nexists \ell \in L, \sigma, \tau \in \mathrm{SUB}(\Sigma, \mathcal{V}) : \ell\sigma = t\tau\} \\ S_L &= \{t \in S'_L \quad\quad\quad \mid \nexists u \in S'_L : u < t\} \end{aligned}$$

One might wonder why unification is considered, while term rewriting is concerned with matching. This becomes clear when formulating what kind of terms we are looking for: the set of terms that *match* those terms which are not *matched* by left-hand sides. This means we have to consider two matchings at the same time, when assuming that the set of variables are disjoint then this gives rise to a unification problem.

As a next step we show that the set S'_L is closed under substitution. This is of interest, since we want to consider the ground terms that are matched by a term contained in S'_L. Thus, it should be the case that every instantiation of a term from S'_L is also contained in S'_L, such that especially this holds for ground instances.

Lemma 5.2 $\{s\sigma \in \mathcal{T}(\Sigma, \mathcal{V}) \mid s \in S'_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} = S'_L$.

Proof. "$\supseteq$": Holds trivially for $\sigma = id$.

"$\subseteq$": Let $s \in S'_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$. Then $s\sigma \in \{s\sigma \in \mathcal{T}(\Sigma, \mathcal{V}) \mid s \in S'_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$. We have that $\ell\sigma' \neq s\tau'$ for all $\ell \in L$ and all substitutions $\sigma', \tau' \in \mathrm{SUB}(\Sigma, \mathcal{V})$. Therefore, especially $\ell\sigma' \neq s\sigma\tau'$ holds for all substitutions $\sigma', \tau' \in \mathrm{SUB}(\Sigma, \mathcal{V})$. Hence, $s\sigma \in S'_L$. □

The set S_L is derived from the set S'_L by taking only the minimal elements of S'_L w.r.t. the order $>$. These minimal elements exist since this order is well-founded, as for example mentioned in [10, Proposition 3.2].

Lemma 5.3 *[10] The relation $>$ from Definition 5.1 is well-founded.*

This lemma can be proved rather straightforwardly for linear terms by counting the number of symbols. For non-linear terms, one must also consider variables being matched by more than one variable.

When removing larger elements from a set w.r.t. $\leq$, then all terms that are matched by removed terms are still being matched by some term in the set, as shown next. This will be used to show that S_L still matches the same terms as S'_L.

Lemma 5.4 $\{u\sigma \mid u \in U, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} = \{u\sigma \mid u \in U \cup U', \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$ *for every* $U, U' \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ *with* $U' = \{u' \mid \exists u \in U : u \leq u'\}$.

Proof. "$\subseteq$": trivial, since $U \subseteq U \cup U'$.

"$\supseteq$": Let $u' \in U \cup U'$, let $\sigma' \in \mathrm{SUB}(\Sigma, \mathcal{V})$. If $u' \in U$, then the property trivially holds. So let $u' \in U' \setminus U$. Then a $u \in U$ exists such that $u \leq u'$, i.e., there is a substitution $\tau \in \mathrm{SUB}(\Sigma, \mathcal{V})$ such that $u' = u\tau$. Hence, $u'\sigma' = u\tau\sigma' \in \{u\sigma \mid u \in U, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$. □

For the set S'_L it should be intuitively clear that all terms that are not matched by a term contained in L are matched by a term in that set. Using the above lemma, we can now show that this already holds for the set S_L.

Lemma 5.5 $\{s\sigma \in \mathcal{T}(\Sigma, \mathcal{V}) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} = S'_L$.

Proof. "$\subseteq$": Since $S_L \subseteq S'_L$, this holds due to Lemma 5.2.

"$\supseteq$": Since $>$ is well-founded due to Lemma 5.3, the existence of the minimal elements in S_L is guaranteed. Thus, Lemma 5.4 shows the desired property. □

This allows us to prove that the ground terms matched by S_L are indeed those terms that are not matched by the set L.

Lemma 5.6 $\mathcal{T}(\Sigma) \setminus \{\ell\sigma \in \mathcal{T}(\Sigma) \mid \ell \in L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} = \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$.

Proof. This lemma is shown in two steps: First it is proven that $\{\ell\sigma \in \mathcal{T}(\Sigma) \mid \ell \in L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} \cap \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} = \emptyset$ (showing "$\supseteq$"), and in the second step it is shown that $\mathcal{T}(\Sigma) = \{\ell\sigma \in \mathcal{T}(\Sigma) \mid \ell \in L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} \cup \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$ (showing "$\subseteq$").

For the first step, let $t \in \{\ell\sigma \in \mathcal{T}(\Sigma) \mid \ell \in L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} \cap \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$. Thus, there exist $\ell \in L$ and $\sigma_\ell \in \mathrm{SUB}(\Sigma, \mathcal{V})$ such that $t = \ell\sigma_\ell$ and there exist $s \in S_L \subseteq S'_L$ and $\sigma_s \in \mathrm{SUB}(\Sigma, \mathcal{V})$ such that $t = s\sigma_s$. Putting this together gives $\ell\sigma_\ell = t = s\sigma_s$, which is a contradiction to the definition of S'_L.

To show the second step, we observe that clearly $\{\ell\sigma \in \mathcal{T}(\Sigma) \mid \ell \in L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} \cup \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} \subseteq \mathcal{T}(\Sigma)$. So it remains to be shown that $\{\ell\sigma \in \mathcal{T}(\Sigma) \mid \ell \in L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\} \cup \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in$

$\mathrm{SUB}(\Sigma, \mathcal{V})\} \supseteq \mathcal{T}(\Sigma)$. For that purpose, let $t \in \mathcal{T}(\Sigma)$ be an arbitrary ground term. In case there exist $\ell \in L$ and $\sigma_\ell \in \mathrm{SUB}(\Sigma, \mathcal{V})$ such that $\ell\sigma_\ell = t$ the property has been shown. Otherwise, we may assume that for all $\ell \in L$ and all substitutions $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ that satisfy $\ell\sigma \in \mathcal{T}(\Sigma)$ we have $\ell\sigma \neq t$. Since t is a ground term, $\mathcal{V}(t) = \emptyset$ holds. This means that for any substitution $\tau \in \mathrm{SUB}(\Sigma, \mathcal{V})$ we have $t\tau = t$. Furthermore, for every term $t' \in \mathcal{T}(\Sigma, \mathcal{V})$ with $\mathcal{V}(t') \neq \emptyset$ it holds that $t \neq t'$ which allows to conclude that $t \neq \ell\sigma'$ for all substitutions $\sigma' \in \mathrm{SUB}(\Sigma, \mathcal{V})$ where $\mathcal{V}(\ell\sigma') \neq \emptyset$. Putting this together, we get that for all substitutions $\sigma, \tau \in \mathrm{SUB}(\Sigma, \mathcal{V})$ it holds that $\ell\sigma \neq t = t\tau$. From the definition of S'_L we get $t \in S'_L$, and hence $t \in \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$ by Lemma 5.5. □

In the following we restrict ourselves to sets L that only contain linear terms. It should be observed that this also covers the case of a quasi left-linear TRS R: for such a TRS we can define L to be the set of all linear left-hand sides of R and have that L still matches the same terms as $\mathrm{lhs}(R)$, due to Lemma 5.4. We want to show that for a linear set L the set S_L is finite. For that purpose we need the depth of a term, which is defined as follows.

Definition 5.7 The *depth* of a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is defined as $\mathrm{depth}(t) = 0$ if $t \in \mathcal{V}$ and $\mathrm{depth}(f(t_1, \ldots, t_n)) = 1 + \max\{\mathrm{depth}(t_1), \ldots, \mathrm{depth}(t_n)\}$ for $t = f(t_1, \ldots, t_n)$.

The *depth* of a finite set $T \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ is defined as the maximum over the depths of the terms it contains, i.e., $\mathrm{depth}(T) = \max\{\mathrm{depth}(t) \mid t \in T\}$.

Then, we have for example $\mathrm{depth}(\mathsf{f}(x, y)) = 1$, while $\mathrm{depth}(\mathsf{f}(\mathsf{a}, y)) = 2$ for the signature $\Sigma = \{\mathsf{f}, \mathsf{a}\}$. Using this notion of depth, we can now prove the following lemma. It provides an upper bound on the depth of the terms contained in S_L for sets L containing only linear terms.

Lemma 5.8 *For a set $L \subseteq \mathcal{T}_{\mathrm{lin}}(\Sigma, \mathcal{V})$ of linear terms we have that* $\mathrm{depth}(s) \leq \mathrm{depth}(L)$ *for all $s \in S_L$.*

Proof. Assume, there exists a $s \in S_L \subseteq S'_L$ with $\mathrm{depth}(s) > \mathrm{depth}(L)$. Thus, we have that $s\sigma \neq \ell\tau$ for all $\ell \in L$ and all substitutions $\sigma, \tau \in \mathrm{SUB}(\Sigma, \mathcal{V})$. W.l.o.g. we may assume that $\mathcal{V}(s)$ and $\mathcal{V}(\ell)$ are disjoint for all $\ell \in L$. Lemma 2.1 shows that for every $\ell \in L$ a position $\pi_\ell \in \mathit{Pos}(\ell) \cap \mathit{Pos}(s)$ exists such that $\mathsf{root}(\ell|_{\pi_\ell}) \neq \mathsf{root}(s|_{\pi_\ell})$. By definition of $\mathrm{depth}(L)$, we have $|\pi_\ell| < \mathrm{depth}(L)$. Let $\mathrm{trunc}_L(s) \in \mathcal{T}(\Sigma, \mathcal{V})$ denote the term that is derived from s by replacing all subterms at positions of length $\mathrm{depth}(L)$ by fresh variables. By construction, we have $\mathrm{depth}(\mathrm{trunc}_L(s)) = \mathrm{depth}(L)$, $\mathrm{trunc}_L(s) < s$, and $\mathsf{root}(s|_\pi) = \mathsf{root}(\mathrm{trunc}_L(s)|_\pi)$ for all $\pi \in \mathit{Pos}(s)$ with $|\pi| < \mathrm{depth}(L)$. Hence, $\mathsf{root}(\mathrm{trunc}_L(s)|_{\pi_\ell}) = \mathsf{root}(s|_{\pi_\ell}) \neq \mathsf{root}(\ell|_{\pi_\ell})$, i.e., for all substitutions $\sigma, \tau \in \mathrm{SUB}(\Sigma, \mathcal{V})$ we have $\mathrm{trunc}_L(s)\sigma \neq \ell\tau$. Thus $\mathrm{trunc}_L(s) \in S'_L$, which contradicts the minimality of s. □

Furthermore, we only have to consider linear terms for the set S_L, if we are only interested in the matching of ground terms.

Lemma 5.9 *For a set $L \subseteq \mathcal{T}_{\mathrm{lin}}(\Sigma, \mathcal{V})$ of linear terms, we have for every $t \in \mathcal{T}(\Sigma) \cap S'_L$ that $s \in S_L \cap \mathcal{T}_{\mathrm{lin}}(\Sigma, \mathcal{V})$ and $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ exist with $s\sigma = t$.*

Proof. Let $t \in \mathcal{T}(\Sigma) \cap S'_L$. Then for all $\ell \in L$ and all $\tau \in \mathrm{SUB}(\Sigma, \mathcal{V})$ we have that $\ell\tau \neq t$. Since $\mathcal{T}(\Sigma) \subseteq \mathcal{T}_{\text{lin}}(\Sigma, \mathcal{V})$, we get from Lemma 2.1 that a $\pi_\ell \in Pos(\ell) \cap Pos(t)$ exists with $\mathsf{root}(t|_{\pi_\ell}) \neq \mathsf{root}(\ell|_{\pi_\ell})$.

There exist $s \in S_L$ and $\sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})$ with $s\sigma = t$, due to Lemma 5.5. In case $s \in \mathcal{T}_{\text{lin}}(\Sigma, \mathcal{V})$ nothing has to be proven.

Otherwise, we start with the term $\text{lin}(s)$ that is created from s by replacing every occurrence of a variable by a fresh variable, thereby generating a linear term. Then clearly, there is a substitution σ' such that $\text{lin}(s)\sigma' = t$. If there is an $\ell' \in L$ and a substitution $\tau \in \mathrm{SUB}(\Sigma, \mathcal{V})$ such that $\text{lin}(s)\tau = \ell'\tau$ (where we assume $\text{lin}(s)$ and ℓ' to be variable disjoint), then we replace in s the variable at a position $\pi_s \in Pos(s)$ that is a prefix of π_ℓ by $f(x_1, \ldots, x_n)$, where $f = \mathsf{root}(t|_{\pi_s})$, $\mathsf{arity}(f) = n$, and the x_i are fresh variables. This variable must exist, otherwise ℓ' would match t. This process is repeated until there are no more ℓ' that unify with the thereby constructed term s'. By construction s' is linear and does not unify with any term from L. Furthermore, this term is minimal in S'_L w.r.t. $>$, therefore $s' \in S_L$, which shows our claim. □

From the above lemmas, we can give the following construction of an anti-matching set S_L w.r.t. a set L of linear terms. Let $d = \text{depth}(L)$ be the maximal depth of terms occurring in L. Start by S' being the finite set of all linear terms up to renaming of variables of depth $\leq d$. Next remove all terms from S' that unify with a term from L. Finally initialize S_L to S' and remove all non-minimal elements t from S_L, i.e., every term t for which a $u \in S$ exists with $u < t$ is removed from S_L. From Lemmas 5.8 and 5.9 we then know that all ground terms that are not matched by L are matched by S_L.

Using this construction and the above lemmas, we can now show Theorem 3.3. It states that for a quasi left-linear TRS R a finite, computable, and unique anti-matching set S_L exists that matches exactly those terms that $\text{lhs}(R)$ does not match. Please note that we only have to consider a linear set L that matches all ground terms matched by $\text{lhs}(R)$, as we already observed above.

Proof of Theorem 3.3. Let $L \subseteq \mathcal{T}_{\text{lin}}(\Sigma, \mathcal{V})$ be the finite set of linear left-hand sides of R. Then L matches all terms that can be rewritten by R due to Lemma 5.4. Let $S_L \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ be defined as given in Definition 5.1. As we can see from Lemma 5.6, we have that for all ground terms $t \in \mathcal{T}(\Sigma)$ it holds that $t \in \{s\sigma \in \mathcal{T}(\Sigma) \mid s \in S_L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$ iff $t \notin \{\ell\sigma \in \mathcal{T}(\Sigma) \mid \ell \in L, \sigma \in \mathrm{SUB}(\Sigma, \mathcal{V})\}$. From Lemma 5.8 we get that S_L is finite, since, up to variable renaming, only finitely many terms whose depth is less than or equal to $\text{depth}(L)$ exist for a finite signature Σ. Lemma 5.9 shows that $S_L = S_L \cap \mathcal{T}_{\text{lin}}(\Sigma, \mathcal{V})$, and finally the sketched construction shows that the set S_L is computable and unique since the minimal elements w.r.t. $>$ are unique. □

Finally, we want to further analyze the case of TRSs that are not quasi left-linear. For this purpose, let $L = \{\mathsf{f}(x, x)\}$ be the left-hand sides of a TRS over the signature $\Sigma = \{\mathsf{f}, \mathsf{g}\}$. Then for every $n \in \mathbb{N}$ we have $\mathsf{f}(x, \mathsf{g}^n(x)) \in S'_L$. Furthermore, there is no term $s \neq \mathsf{f}(x, \mathsf{g}^n(x)) \in S'_L$ such that $s\sigma = \mathsf{f}(x, \mathsf{g}^n(x))$, which shows that S_L is infinite. To show that this is not due to choosing the set S_L, we prove the proposition below, stating that S_L is the smallest set that has the desired property.

Proposition 5.10 *Let $L \subseteq \mathcal{T}(\Sigma, \mathcal{V})$. For every anti-matching set $S \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ w.r.t. L we have $S_L \subseteq S \subseteq S'_L$, where we disregard variable renamings.*

Proof. The inclusion $S \subseteq S'_L$ can be seen directly from the definition of S'_L.

Assume, there is such a set $S \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ with $S_L \not\subseteq S$. Then, there is a term $s' \in S_L$ such that $s' \notin S$. Furthermore, it must be the case that $\{s\sigma \mid s \in S, \sigma \in \text{SUB}(\Sigma, \mathcal{V})\} = \{s\sigma \mid s \in S_L, \sigma \in \text{SUB}(\Sigma, \mathcal{V})\} = S'_L$, i.e., there must be an $s \in S$ and a $\sigma \in \text{SUB}(\Sigma, \mathcal{V})$ such that $s\sigma = s'$. This implies that $s \leq s'$. In case we also have $s' \leq s$, then $s' \in S$, contradicting our assumption. But otherwise $s < s'$ holds, which contradicts the minimality of s'. □

As a consequence of Proposition 5.10 and the previously observed fact that for $L = \{\mathsf{f}(x, x)\}$ it holds that $S_L \supseteq \{\mathsf{f}(x, \mathsf{g}^n(x)) \mid n \in \mathbb{N}\}$, we conclude that any set S that matches those terms which are not matched by a term in L must be infinite, since already $S_L \subseteq S$ is infinite.

6 Implementation and Experiments

We have implemented the transformations as described in the previous sections. Even though the construction of the set S_L of terms that match those terms not matched by the set L of left-hand sides of the input term rewrite system can certainly be improved, the complete transformation only takes a neglegible amount of time for all of the following examples.

Our implementation allows for a number of different variants of the transformation T_1 to be used. In Section 3 only one of these was presented, this proved to be the most effective one in our experiments. In detail, one can choose whether or not to add the blocking symbol when the symbol down descends into a term that is not matched by a left-hand side of the original term rewrite system. Also, one can choose whether the symbols upon descending should be rewritten to a marked version of that symbol. As a last option, one can also use a modified version of the rules for the up symbol, however this modification proved itself not to be effective.

The transformed system was then used as input for the termination provers Jambox [3], TTT2 [13], and AProVE [6], which were the strongest tools of the 2007 termination competition in the TRS category [15]. The reason why we used multiple tools was that the transformation turned out to produce rewrite systems for which sometimes one tool succeeded in proving termination of the transformed TRS, while at least one of the other tools was unable to do so.

Below we present some examples. First, we want to show that Example 2.2 really is outermost ground terminating, as claimed above. When this example is transformed, the following TRS is created:

Example 6.1 *(Transformation of Example 2.2)*

$$T_1(R) = \begin{cases} \mathsf{top}(\mathsf{up}(x)) & \to \mathsf{top}(\mathsf{down}(x)) \quad \mathsf{down}(\mathsf{b}) \to \mathsf{up}(\mathsf{a}) \\ \mathsf{down}(\mathsf{f}(x, \mathsf{a})) & \to \mathsf{up}(\mathsf{f}(x, \mathsf{b})) \quad \mathsf{down}(\mathsf{f}(\mathsf{a}, x)) \to \mathsf{up}(\mathsf{a}) \\ \mathsf{f}^\natural(\mathsf{block}(x), \mathsf{up}(y)) & \to \mathsf{up}(\mathsf{f}(x, y)) \quad \mathsf{down}(\mathsf{f}(\mathsf{b}, x)) \to \mathsf{up}(\mathsf{a}) \\ \mathsf{f}^\natural(\mathsf{up}(x), \mathsf{block}(y)) & \to \mathsf{up}(\mathsf{f}(x, y)) \quad \mathsf{down}(\mathsf{f}(\mathsf{f}(x, y), z)) \to \mathsf{up}(\mathsf{a}) \end{cases}$$

It can be observed that in the transformed TRS there are no rules that allow the symbol down to descend into a term. This holds because we have $S_L = \{\mathsf{a}\}$, such that no rules are created for it. The transformed TRS can easily be shown terminating within a short amount of time by all of the considered tools. For the next example, this is not the case anymore.

Example 6.2

$$R_2 = \begin{cases} \mathsf{a} & \to \mathsf{f}(\mathsf{a}) \\ \mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{f}(x))))) & \to \mathsf{b} \end{cases}$$

Both AProVE and TTT2 can show termination of $T_1(R_2)$, while Jambox fails to do so. What is also interesting is that TTT2 uses RFC Match Bounds to show this, while AProVE uses only Dependency Pairs and a large number of rewriting steps, but is able to find this proof much faster than TTT2.

The next example proved to be rather difficult for all of the considered tools. It is the example from the introduction for the case $n = 1$.

Example 6.3

$$R_3 = \begin{cases} \mathsf{from}(x) & \to x : \mathsf{from}(\mathsf{s}(x)) \\ \mathsf{s}(x) : xs & \to \mathsf{overflow} \end{cases}$$

This example could only be proven outermost ground terminating using the tool Jambox, both AProVE and TTT2 failed. However, the techniques used by Jambox to prove termination, namely semantic labelling and polynomial orders, are also implemented in both of the other tools. Hence, this clearly shows that proving termination is also strongly dependent on heuristics and/or search encodings.

If we instantiate n with 2 or 3 in the example given in the introduction, then none of the considered termination tools was able to prove termination of the resulting TRS $T_1(R)$ in a reasonable amount of time. However, when using one of the transformations presented in Section 4, then for both values of n the TRSs $T_2(R)$ and $T_3(R)$ can be proven terminating using the tools Jambox and AProVE. Hence, even though these transformations are incomplete, they can be used to show outermost ground termination of examples where the transformation T_1 does not lead to a successful proof.

In the examples that we considered so far, we had that always the right-hand side of the rule that caused the outermost ground termination was a ground term. This is different in the next example.

Example 6.4

$$R_4 = \begin{cases} \mathsf{f}(\mathsf{f}(\mathsf{g}(x))) & \to x \\ \mathsf{g}(\mathsf{b}) & \to \mathsf{f}(\mathsf{g}(\mathsf{b})) \end{cases}$$

This transformed TRS can be shown terminating by the tools TTT2 and Jambox, while AProVE fails.

In the example below, it is the case that the right-hand sides are not always either growing or detecting a term that has grown too large.

Example 6.5

$$R_5 = \begin{cases} \mathsf{f}(\mathsf{f}(x,y),z) & \to & \mathsf{c} \\ \mathsf{f}(x,\mathsf{f}(y,z)) & \to & \mathsf{f}(\mathsf{f}(x,y),z) \\ \mathsf{a} & \to & \mathsf{f}(\mathsf{a},\mathsf{a}) \end{cases}$$

The transformed TRS $T_1(R_5)$ can be shown terminating by both AProVE and Jambox, while for this example TTT2 fails to show termination. If the first rule is changed to $\mathsf{f}(\mathsf{f}(x,y),z) \to \mathsf{f}(\mathsf{c},x)$, then only AProVE can show the transformed TRS to be terminating.

Next, we want to consider the 6 examples contained in the Termination Problem DataBase (TPDB) [15] that require outermost termination analysis. All of these examples can be proven outermost ground terminating by the tool Cariboo [9], which was mentioned in the introduction. Of these 6 examples, 5 are left-linear, therefore they can be directly handled by our approach. For these, we can show outermost ground termination using Jambox as termination prover. The last example shall be considered in more detail below.

Example 6.6 *(Outermost Example 6)*

$$R_6 = \begin{cases} \mathsf{f}(x,x) & \to & \mathsf{f}(\mathsf{i}(x),\mathsf{g}(\mathsf{g}(x))) & \qquad \mathsf{f}(x,\mathsf{i}(\mathsf{g}(x))) & \to & \mathsf{a} \\ \mathsf{f}(x,y) & \to & x & \qquad \mathsf{f}(x,\mathsf{i}(x)) & \to & \mathsf{f}(x,x) \\ \mathsf{g}(x) & \to & \mathsf{i}(x) & & & \end{cases}$$

As can be seen above, this example has non-linear left-hand sides for the function symbol f. However, these left-hand sides are all instances of the left-hand side $\mathsf{f}(x,y)$, which makes this TRS quasi left-linear. Hence, we only have to consider the set $L = \{\mathsf{f}(x,y),\mathsf{g}(x)\}$ of linear terms, from which we then compute S_L to be $S_L = \{\mathsf{a},\mathsf{i}(x)\}$. Using this set, our transformation yields a finite TRS $T_1(R_6)$, whose termination can be proven using any of the three considered tools.

Finally, we want to compare the strength of our approach against that of Cariboo. The following example is non-terminating for normal rewriting, since already the rule $\mathsf{h}(x) \to \mathsf{f}(\mathsf{h}(x))$ allows an infinite reduction.

Example 6.7

$$R_7 = \begin{cases} \mathsf{f}(\mathsf{h}(x)) & \to & \mathsf{f}(\mathsf{i}(x)) & \qquad \mathsf{h}(x) & \to & \mathsf{f}(\mathsf{h}(x)) \\ \mathsf{f}(\mathsf{i}(x)) & \to & x & \qquad \mathsf{i}(x) & \to & \mathsf{h}(x) \end{cases}$$

Cariboo is unable to prove outermost ground termination of the TRS R_7, while the transformed TRS $T_1(R_7)$ can be proven terminating by all considered tools. Also Example 6.4 and both variants of Example 6.5 cannot be proven outermost ground terminating by Cariboo.

There are also examples where Cariboo succeeds, whereas our transformation fails. First of all, Cariboo can also handle examples that are not quasi left-linear, while our transformation is not applicable in this case. But there are also quasi left-linear examples where Cariboo can prove outermost ground termination, but none of the considered tools can prove termination of the transformed TRS. Such an example is given below.

Example 6.8

$$R_8 = \left\{ \begin{array}{lcllcl} \mathsf{f}(\mathsf{g}(x),\mathsf{h}(x)) & \to & \mathsf{f}(\mathsf{i}(x),\mathsf{h}(x)) & \quad \mathsf{f}(x,\mathsf{j}(y)) & \to & \mathsf{a} \\ \mathsf{f}(\mathsf{i}(x),\mathsf{h}(x)) & \to & \mathsf{a} & \quad \mathsf{i}(x) & \to & \mathsf{g}(x) \\ \mathsf{f}(x,\mathsf{h}(y)) & \to & \mathsf{f}(x,\mathsf{j}(y)) & \quad \mathsf{j}(x) & \to & \mathsf{h}(x) \end{array} \right.$$

This quasi left-linear example can be shown terminating by Cariboo, whereas for all termination provers the transformed TRSs $T_1(R_8)$ and $T_3(R_8)$ are too hard, while $T_2(R_8)$ allows for an infinite reduction.

7 Conclusion

We have presented different transformations such that outermost ground termination of a TRS follows from termination of the transformed TRS. These transformations are sound for arbitrary term rewrite systems, but only for finite quasi left-linear term rewrite systems the transformed term rewrite system is finite and can be constructed automatically. For this class of term rewrite systems we implemented the transformations and we were able to prove ground outermost termination of several non-trivial examples, including all 6 examples contained in the TPDB that require an outermost strategy. When comparing the presented approach to the existing one implemented in the tool Cariboo [9], then we have shown that our approach can prove several term rewrite systems to be outermost ground terminating where Cariboo fails. However, it should also be mentioned that for some examples Cariboo succeeds but our transformation fails. Especially, Cariboo is not limited to quasi left-linear TRSs, but also when considering only such TRSs there are examples with this behavior. However, some of these quasi left-linear examples might, due to the nature of our approach, be proven outermost ground terminating in the future using the presented transformation, when the underlying termination provers become even more powerful.

As already presented in Section 4, the transformations based on contexts are not complete, i.e., from the non-termination of the transformed TRS one can not conclude that the original TRS contains an infinite outermost ground reduction. However, recent work suggests that the transformation presented in Section 3 is complete when restricting to TRSs R that satisfy the condition $\mathcal{V}(r) \subseteq \mathcal{V}(\ell)$ for all rules $\ell \to r \in R$. Proving this remains a task for the future.

In the present TPDB, the set of TRSs requiring an outermost strategy, so being non-terminating when disregarding the strategy, is quite limited: there are only 6. Therefore, more such examples should be added to the TPDB to allow for a better comparison of different approaches that try to prove outermost termination automatically.

References

[1] Baader, F. and T. Nipkow, "Term Rewriting and All That," Cambridge University Press, Cambridge, UK, 1998.

[2] Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer and C. Talcott, *The Maude 2.0 System*, in: *Proceedings of the 14th Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science **2706** (2003), pp. 76–87.

[3] Endrullis, J., *Jambox 2.0e*, downloadable from http://joerg.endrullis.de.

[4] Futatsugi, K. and R. Diaconescu, editors, "CafeOBJ Report," World Scientific Publishing Company, 1998.

[5] Giesl, J. and A. Middeldorp, *Transformation Techniques for Context-Sensitive Rewrite Systems*, Journal of Functional Programming **14** (2004), pp. 379–427.

[6] Giesl, J., P. Schneider-Kamp and R. Thiemann, *AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework*, in: *Proceedings of the 3rd International Joint Conference on Automatic Reasoning*, Lecture Notes in Computer Science **4130** (2006), pp. 281–286, downloadable from http://aprove.informatik.rwth-aachen.de.

[7] Giesl, J., S. Swiderski, P. Schneider-Kamp and R. Thiemann, *Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages*, in: *Proceedings of the 17th Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science **4098**, 2006, pp. 297–312.

[8] Giesl, J. and H. Zantema, *Liveness in Rewriting*, in: *Proceedings of the 14th Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science **2706** (2003), pp. 321–336.

[9] Gnaedig, I. and H. Kirchner, *Termination of rewriting under strategies*, ACM Transactions on Computational Logic (2007), to appear, available at http://tocl.acm.org/accepted/315gnaedig.ps.

[10] Huet, G., *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems*, J. ACM **27** (1980), pp. 797–821.

[11] Jones, S. P., editor, "Haskell 98 Language and Libraries: The Revised Report," Cambridge University Press, 2003, also available from http://www.haskell.org/definition.

[12] Kirchner, C., R. Kopetz and P.-E. Moreau, *Anti-Pattern Matching*, in: *Proceedings of the 16th European Symposium on Programming*, Lecture Notes in Computer Science **4421** (2007), pp. 110–124.

[13] Korp, M., C. Sternagel, H. Zankl and A. Middeldorp, *Tyrolean Termination Tool 2 (TTT2)*, downloadable from http://colo6-c703.uibk.ac.at/ttt2.

[14] Lassez, J.-L. and K. Marriot, *Explicit Representation of Terms Defined by Counter Examples*, Journal of Automated Reasoning **3** (1987), pp. 301–317.

[15] Marché, C. and H. Zantema, *The Termination Competition*, in: *Proceedings of the 18th Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science **4533** (2007), pp. 303–313, see also http://www.lri.fr/~marche/termination-competition.

[16] Panitz, S. E. and M. Schmidt-Schauss, *TEA: Automatically proving termination of programs in a non-strict higher order functional language*, in: *Proceedings of the 4th International Symposium on Static Analysis*, Lecture Notes in Computer Science **1302**, 1997, pp. 345–360.

[17] Terese, "Term Rewriting Systems," Cambridge University Press, Cambridge, UK, 2003.