From outermost termination to innermost termination

René Thiemann

Institute of Computer Science, University of Innsbruck, Austria
rene.thiemann@uibk.ac.at

Abstract. Rewriting is the underlying evaluation mechanism of functional programming languages. Therefore, termination analysis of term rewrite systems (TRSs) is an important technique for program verification. To capture the evaluation mechanism of a programming language one has to take care of the evaluation strategy, where we focus on the outermost strategy.
As there are only few techniques available to analyze outermost termination of TRSs directly, we introduce a new transformation such that a TRS is outermost terminating iff the transformed TRS is innermost terminating. In this way all of the several techniques that have been developed to investigate innermost termination become applicable to analyze outermost termination, too. We have implemented the transformation and successfully evaluated it on a large collection of TRSs.

1 Introduction

Termination is an essential property of programs and in the last years there has been much progress in the area of automated termination analysis. We consider term rewrite systems (TRSs) since rewriting is the basic evaluation mechanism of functional programs [17]. However, to mimic the evaluation of programs correctly it is essential to respect the evaluation strategy—like outermost or innermost.

Proving termination of TRSs without a fixed strategy (full termination) is a well-studied field with a large collection of available techniques [1,3,7,8,11,13,20]. Also for innermost termination analysis there are many techniques available [1,4,6,7,13,19] which include more powerful variants of those methods that are used for full termination. However, there are only few techniques available which investigate outermost termination of TRSs [12,18]. Whereas [12] introduces a direct method to prove outermost termination, [18] presents a transformational approach: a given TRS is transformed into another TRS such that full termination of the resulting TRS implies outermost termination of the original TRS. In this way, one can reuse all the existing techniques for proving full termination.

In this paper we present another transformational approach to analyze outermost termination, but with two major differences compared to [18]. First, in our approach one has to investigate innermost termination of the resulting TRS and second, our transformation is proven to be complete. Thus, as in [18], we can also profit from a variety of existing techniques for termination analysis

and even more important, we can additionally disprove outermost termination. Therefore—as far as the author knows—we present the first automatic method for disproving outermost termination.

The paper is structured as follows. In Sect. 2 we recapitulate the required notions of term rewriting. Then in Sect. 3 our transformation is presented. Soundness and completeness of the transformation are proven in Sect. 4 and Sect. 5, respectively. An improved version of the transformation is presented in Sect. 6. Finally, a discussion on related work and experimental data is given in Sect. 7.

2 Preliminaries

We refer to [2] for the basics of term rewriting. We always assume a countable infinite set of variables $\mathcal{V}$. We write $\mathcal{T}(\Sigma, \mathcal{V})$ for the set of terms over some signature Σ. For each $f \in \Sigma$ we write $\mathrm{ar}(f)$ for the arity of f. A TRS $\mathcal{R}$ over Σ is a set of rules $\ell \to r$ where $\ell, r \in \mathcal{T}(\Sigma, \mathcal{V})$, $\ell \notin \mathcal{V}$, and $\mathcal{V}(\ell) \supseteq \mathcal{V}(r)$. Here, $\mathcal{V}(t)$ is the set of variables occurring in a term t. We restrict ourselves to finite TRSs and often omit the signature Σ if it is clear from the context. For a TRS the defined symbols are the root symbols of the left-hand sides of the rules. The remaining symbols of Σ are constructors. A position $p \in \mathit{Pos}(t)$ is either the empty position ε or $p = iq$ where $t = f(t_1, \ldots, t_n)$, $1 \leq i \leq n$, and $q \in \mathit{Pos}(t_i)$. Position p is strictly above position q (or q is strictly below p) iff p is a proper prefix of q. A context is a term C with a hole at some position $p \in \mathit{Pos}(C)$. We write $C[t]_p$ as the term where the hole of C is replaced by t.

A term t can be reduced with $\mathcal{R}$ to s at position p, written $t \to_{\mathcal{R},p} s$ iff $t = C[\ell\sigma]_p$ and $s = C[r\sigma]_p$ for some rule $\ell \to r \in \mathcal{R}$ and substitution σ. The reduction is an *outermost reduction*, written $t \xrightarrow{\mathsf{o}}_{\mathcal{R},p} s$ iff there is no q strictly above p such that $t \to_{\mathcal{R},q} s$. It is an *innermost reduction*, written $t \xrightarrow{\mathsf{i}}_{\mathcal{R},p} s$ iff there is no q strictly below p such that $t \to_{\mathcal{R},q} s$. The rewrite relation of $\mathcal{R}$ is defined as $t \to_{\mathcal{R}} s$ iff $t \to_{\mathcal{R},p} s$ for some position p. The outer- and innermost rewrite relations $\xrightarrow{\mathsf{o}}_{\mathcal{R}}$ and $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$ are defined accordingly via $\xrightarrow{\mathsf{o}}_{\mathcal{R},p}$ and $\xrightarrow{\mathsf{i}}_{\mathcal{R},p}$. A term t is in normal form w.r.t. $\mathcal{R}$ iff there is no s such that $t \to_{\mathcal{R}} s$. A TRS is (outermost / innermost) terminating iff $\to_{\mathcal{R}}$ ($\xrightarrow{\mathsf{o}}_{\mathcal{R}}$ / $\xrightarrow{\mathsf{i}}_{\mathcal{R}}$) is well-founded.

Example 1. Consider the following TRS $\mathcal{R}_1$ which generates a list of zeros. Here, the second rule is used to stop the reduction if a cons appears at the outside.

$$\mathsf{zeros} \to \mathsf{cons}(\mathsf{0}, \mathsf{zeros}) \tag{1}$$

$$\mathsf{cons}(x, xs) \to \mathsf{terminate} \tag{2}$$

It is neither terminating nor innermost terminating due to the infinite reduction

$$\mathsf{zeros} \xrightarrow{\mathsf{i}}_{\mathcal{R}_1} \mathsf{cons}(\mathsf{0}, \mathsf{zeros}) \xrightarrow{\mathsf{i}}_{\mathcal{R}_1} \mathsf{cons}(\mathsf{0}, \mathsf{cons}(\mathsf{0}, \mathsf{zeros})) \xrightarrow{\mathsf{i}}_{\mathcal{R}_1} \ldots.$$

However, $\mathcal{R}_1$ is outermost terminating as $\mathsf{cons}(\mathsf{0}, \mathsf{zeros})$ must be outermost reduced to $\mathsf{terminate}$ because of rule (2). But if we add the following rules to compare the lengths of two lists

$$\begin{aligned}
\mathsf{longer}(\mathsf{nil}, ys) &\to \mathsf{false}\\
\mathsf{longer}(\mathsf{cons}(x, xs), \mathsf{nil}) &\to \mathsf{true}\\
\mathsf{longer}(\mathsf{cons}(x, xs), \mathsf{cons}(y, ys)) &\to \mathsf{longer}(xs, ys)
\end{aligned}$$

then the resulting TRS $\mathcal{R}_2$ is not outermost terminating.

$$\begin{aligned}
\mathsf{longer}(\mathsf{zeros}, \mathsf{zeros}) &\xrightarrow{o}_{\mathcal{R}_2} \mathsf{longer}(\mathsf{cons}(0, \mathsf{zeros}), \mathsf{zeros})\\
&\xrightarrow{o}_{\mathcal{R}_2} \mathsf{longer}(\mathsf{cons}(0, \mathsf{zeros}), \mathsf{cons}(0, \mathsf{zeros}))\\
&\xrightarrow{o}_{\mathcal{R}_2} \mathsf{longer}(\mathsf{zeros}, \mathsf{zeros}) \xrightarrow{o}_{\mathcal{R}_2} \dots
\end{aligned}$$

3 The transformation

To study the outermost termination behavior of a TRS we take a transformational approach. The idea is to transform a given TRS into another TRS such that outermost termination of the original TRS can be concluded from termination of the transformed TRS. Transformational approaches to prove termination are quite common for extensions of plain rewriting, e.g., there are various transformations for conditional TRSs and context-sensitive TRSs [14]. An overview over these transformations is given in [15] and [5].

The general structure of our transformation is similar to the structure of the complete transformation for context-sensitive TRSs [5]. (This structure is also used in [18].) First, each term of the original signature is mapped to a corresponding term over an enriched signature which contains additional symbols (also called markers). The symbol top will be used to mark the top of the original term and the symbol reduce and $\mathsf{go\_up}$ are used to indicate the position of the reduction. Then one simulates a reduction $t \xrightarrow{o}_{\mathcal{R},p} s$ of the original TRS in three phases.

- First, in the term $\mathsf{top}(\mathsf{reduce}(t))$ the reduce-marker is moved to position p.
- Second, the reduction to s is simulated by applying the corresponding rewrite rule that is used in the reduction $t \xrightarrow{o}_{\mathcal{R},p} s$. Moreover, the reduce-marker is changed into a $\mathsf{go\_up}$-marker.
- Third, the $\mathsf{go\_up}$-marker is moved back from position p to the top of the term yielding $\mathsf{top}(\mathsf{go\_up}(s))$, and finally $\mathsf{go\_up}$ is converted into the reduce-marker to be able to perform an upcoming reduction.

To simulate outermost rewriting with this scheme, the most difficult part is the first phase. Here, the rules of the transformed systems must ensure that reduce can be moved to all valid positions w.r.t. outermost rewriting to ensure that every original reduction can be simulated. Moreover, it is also desirable that reduce cannot be moved to any of the remaining positions in order to obtain a precise simulation.

In detail, one has to investigate the set of positions of a term $t = f(t_1, \dots, t_n)$ which are allowed w.r.t. outermost rewriting, and move the reduce-marker in the term $\mathsf{reduce}(t)$ accordingly. (The top-marker is not important for this part.)

There are two possibilities. Either t itself is a redex. Then it is not allowed to reduce any of t's subterms and therefore, the reduce-marker should not be moved. Or otherwise, t is not a redex. In that case it must be possible in the transformed system to rewrite $\mathsf{reduce}(t)$ to $C[\mathsf{reduce}(t_i)]$ where C is some context which also has to store $f, t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$.

Applying this idea leads to the following transformation which will be discussed in detail directly afterwards.

Definition 2 (Transformation of outermost TRSs). *Let $\mathcal{R}$ be a TRS over Σ. We define the* extended signature *as $\Sigma' = \Sigma \uplus \{\mathsf{top}, \mathsf{reduce}, \mathsf{go\_up}, \mathsf{result}\} \uplus \{\mathsf{check}_f, \mathsf{redex}_f, \mathsf{in}_{f,i} \mid f \in \Sigma, 1 \leq i \leq \mathrm{ar}(f)\}$. Moreover, we define the* transformed system *$\mathcal{R}'$ over Σ' to consist of the following rules*

$$\mathsf{reduce}(f(x_1, \ldots, x_n)) \to \mathsf{check}_f(\mathsf{redex}_f(x_1, \ldots, x_n)) \tag{3}$$
$$\mathsf{check}_f(\mathsf{redex}_f(x_1, \ldots, x_n)) \to \mathsf{in}_{f,i}(x_1, \ldots, \mathsf{reduce}(x_i), \ldots, x_n) \tag{4}$$
$$\mathsf{redex}_f(\ell_1, \ldots, \ell_n) \to \mathsf{result}(r) \tag{5}$$
$$\mathsf{check}_f(\mathsf{result}(x)) \to \mathsf{go\_up}(x) \tag{6}$$
$$\mathsf{in}_{f,i}(x_1, \ldots, \mathsf{go\_up}(x_i), \ldots, x_n) \to \mathsf{go\_up}(f(x_1, \ldots, x_n)) \tag{7}$$
$$\mathsf{top}(\mathsf{go\_up}(x)) \to \mathsf{top}(\mathsf{reduce}(x)) \tag{8}$$

where (3) and (6) are for all $f \in \Sigma$, (4) and (7) are for all $f \in \Sigma$, $1 \leq i \leq \mathrm{ar}(f)$, and (5) is for all $f(\ell_1, \ldots, \ell_n) \to r \in \mathcal{R}$.

Rules $\{(3), (5), (6)\}$ and $\{(7), (8)\}$ can be used to perform the necessary reductions in the second and third phase of the transformation scheme. To illustrate this, consider $\mathcal{R}_1$ of Ex. 1. One can simulate the outermost reduction $\mathsf{zeros} \xrightarrow{\mathrm{o}}_{\mathcal{R}_1} \mathsf{cons}(0, \mathsf{zeros})$ by applying the rules $(3), (5), (6), (8)$.

$$\begin{aligned} \mathsf{top}(\mathsf{reduce}(\mathsf{zeros})) &\xrightarrow{\mathrm{i}}_{\mathcal{R}_1'} \mathsf{top}(\mathsf{check}_{\mathsf{zeros}}(\mathsf{redex}_{\mathsf{zeros}})) \\ &\xrightarrow{\mathrm{i}}_{\mathcal{R}_1'} \mathsf{top}(\mathsf{check}_{\mathsf{zeros}}(\mathsf{result}(\mathsf{cons}(0, \mathsf{zeros})))) \\ &\xrightarrow{\mathrm{i}}_{\mathcal{R}_1'} \mathsf{top}(\mathsf{go\_up}(\mathsf{cons}(0, \mathsf{zeros}))) \\ &\xrightarrow{\mathrm{i}}_{\mathcal{R}_1'} \mathsf{top}(\mathsf{reduce}(\mathsf{cons}(0, \mathsf{zeros}))) \end{aligned} \tag{9}$$

The first phase needs some more explanation. One might think of a problem that also non-outermost reductions can be simulated in the transformed TRS since every term $\mathsf{reduce}(f(t_1, \ldots, t_n))$ can be rewritten to $C[\mathsf{reduce}(t_i)]$ with rules (3) and (4). However, this problem does not arise since innermost rewriting is considered for the transformed TRS: whenever $f(t_1, \ldots, t_n)$ is a redex w.r.t. $\mathcal{R}$ then an application of rule (4) on the term $\mathsf{check}_f(\mathsf{redex}_f(t_1, \ldots, t_n))$ is prohibited by rule (5) as the latter rule imposes a redex at a deeper position. This can also be seen when continuing in the example. After the reduction

$$\mathsf{top}(\mathsf{reduce}(\mathsf{cons}(0, \mathsf{zeros}))) \xrightarrow{\mathrm{i}}_{\mathcal{R}_1'} \mathsf{top}(\mathsf{check}_{\mathsf{cons}}(\mathsf{redex}_{\mathsf{cons}}(0, \mathsf{zeros})))$$

it is not allowed to apply rule (4), but one has to rewrite the inner subterm.

$$\mathsf{top}(\mathsf{check}_{\mathsf{cons}}(\mathsf{redex}_{\mathsf{cons}}(0, \mathsf{zeros}))) \xrightarrow{\mathrm{i}}_{\mathcal{R}_1'} \mathsf{top}(\mathsf{check}_{\mathsf{cons}}(\mathsf{result}(\mathsf{terminate}))$$

Afterwards it is only possible to rewrite to a normal form. The following main

theorem states that our transformation indeed characterizes outermost termination of the original TRS, i.e., the transformation is both sound and complete.

Theorem 3. *$\mathcal{R}$ is outermost terminating iff $\mathcal{R}'$ is innermost terminating.*

The available techniques for innermost termination analysis in combination with Thm. 3 can now successfully handle the TRSs of Ex. 1. More precisely, outermost termination of $\mathcal{R}_1$ and outermost non-termination of $\mathcal{R}_2$ are proven fully automatically. Before we discuss the empirical results in detail, we prove both directions of Thm. 3 separately in the upcoming two sections.

4 Simulation of outermost reductions

To prove soundness of the transformation we will show that every outermost reduction of the original TRS can be simulated by a series of innermost reductions in the transformed TRS (Lemma 5). To this end, we first make the following observation that each term of the original signature cannot be reduced with the transformed TRS. This observation will be useful to argue that in the simulation we only perform innermost reductions.

Lemma 4. *If $t \in \mathcal{T}(\Sigma, \mathcal{V})$ then t is a normal form w.r.t. $\mathcal{R}'$.*

Proof. Obvious, since every $f \in \Sigma$ is a constructor of $\mathcal{R}'$. □

Lemma 5. *If $t \in \mathcal{T}(\Sigma, \mathcal{V})$ and $t \xrightarrow{\mathrm{o}}_{\mathcal{R}} s$ then $\mathsf{reduce}(t) \xrightarrow{\mathrm{i}}{}^{+}_{\mathcal{R}'} \mathsf{go\_up}(s)$.*

Proof. Since t is reducible it is not a variable, so let $t = f(t_1, \ldots, t_n)$. We perform induction on the position of the redex in $t \xrightarrow{\mathrm{o}}_{\mathcal{R},p} s$.

If $p = \varepsilon$ then $t = \ell\sigma \to r\sigma = s$ for some rule $\ell = f(\ell_1, \ldots, \ell_n) \to r \in \mathcal{R}$ where $t_i = \ell_i\sigma$ for all i. Thus, we can build the following innermost reduction:

$$\begin{aligned}\mathsf{reduce}(f(t_1, \ldots, t_n)) &\xrightarrow{\mathrm{i}}_{\mathcal{R}'} \mathsf{check}_f(\mathsf{redex}_f(t_1, \ldots, t_n)) \\ &\xrightarrow{\mathrm{i}}_{\mathcal{R}'} \mathsf{check}_f(\mathsf{result}(r\sigma)) = \mathsf{check}_f(\mathsf{result}(t)) \\ &\xrightarrow{\mathrm{i}}_{\mathcal{R}'} \mathsf{go\_up}(t)\end{aligned}$$

Note that all these reductions are indeed innermost reductions due to Lemma 4.

If $p = iq$ then $t_i \xrightarrow{\mathrm{o}}_{\mathcal{R}} s_i$ and $s = f(t_1, \ldots, s_i, \ldots, t_n)$. From the induction hypothesis we conclude $\mathsf{reduce}(t_i) \xrightarrow{\mathrm{i}}{}^{+}_{\mathcal{R}'} \mathsf{go\_up}(s_i)$. Thus, it is possible to build the desired reduction as follows:

$$\begin{aligned}\mathsf{reduce}(f(t_1, \ldots, t_n)) &\xrightarrow{\mathrm{i}}_{\mathcal{R}'} \mathsf{check}_f(\mathsf{redex}_f(t_1, \ldots, t_n)) \\ &\xrightarrow{\mathrm{i}}_{\mathcal{R}'} \mathsf{in}_{f,i}(t_1, \ldots, \mathsf{reduce}(t_i), \ldots, t_n) \\ &\xrightarrow{\mathrm{i}}{}^{+}_{\mathcal{R}'} \mathsf{in}_{f,i}(t_1, \ldots, \mathsf{go\_up}(s_i), \ldots, t_n) \\ &\xrightarrow{\mathrm{i}}_{\mathcal{R}'} \mathsf{go\_up}(f(t_1, \ldots, s_i, \ldots, t_n)) \\ &= \mathsf{go\_up}(s)\end{aligned}$$

Here, the second reduction is indeed an innermost step. The reason is that by Lemma 4 we only have to guarantee that the term $\mathsf{redex}_f(t_1, \ldots, t_n)$ is not a redex of $\mathcal{R}'$. However, if this term were a redex of $\mathcal{R}'$, then by the definition of $\mathcal{R}'$ the term $f(t_1, \ldots, t_n) = t$ would be a redex of $\mathcal{R}$. But this is in contradiction

to the fact that t is reduced below the root using outermost rewriting. □

With the help of Lemma 5 it is now easy to prove one direction of Thm. 3.

Corollary 6. *If $\mathcal{R}'$ is innermost terminating then $\mathcal{R}$ is outermost terminating.*

Proof. If $\mathcal{R}$ were not outermost terminating then there would be an infinite reduction

$$t_1 \xrightarrow{o}_{\mathcal{R}} t_2 \xrightarrow{o}_{\mathcal{R}} t_3 \xrightarrow{o}_{\mathcal{R}} \dots$$

where all $t_i \in \mathcal{T}(\Sigma, \mathcal{V})$. Lemma 5 and Lemma 4 directly yield the following infinite innermost reduction of $\mathcal{R}'$.

$$\mathsf{top}(\mathsf{reduce}(t_1)) \xrightarrow{i}{}^{+}_{\mathcal{R}'} \mathsf{top}(\mathsf{go\_up}(t_2)) \xrightarrow{i}_{\mathcal{R}'} \mathsf{top}(\mathsf{reduce}(t_2)) \xrightarrow{i}{}^{+}_{\mathcal{R}'} \dots$$ □

5 Extracting outermost reductions

In this section we prove completeness of our transformation. To this end, we show that outermost termination of the original TRS implies innermost termination of the transformed TRS. This is achieved by extracting an infinite outermost reduction of $\mathcal{R}$ from every infinite innermost reduction of $\mathcal{R}'$. Here we have to deal with a new problem which did not arise in the previous section: for innermost termination we have to consider *all* terms of the extended signature, i.e., we even have to consider terms which contain multiple occurrences of top- and reduce-markers and we have to consider all possible reductions.

To solve this problem we show two main lemmas (7 and 10). In the first lemma we prove that whenever the transformed TRS is not innermost terminating then there also is a reduction of a special form which is similar to the one that we have constructed in the soundness-proof. Then in the second lemma we show how one can extract an outermost reduction from this special reduction. Combining both lemmas then directly yields completeness of our transformation.

Lemma 7. *If $\mathcal{R}'$ is not innermost terminating then there is an infinite innermost reduction of the following form where all t_i are in normal form w.r.t. $\mathcal{R}'$.*[1]

$$\mathsf{top}(\mathsf{go\_up}(t_1)) \xrightarrow{i}_{\mathcal{R}',\varepsilon} \mathsf{top}(\mathsf{reduce}(t_1)) \xrightarrow{i}{}^{*}_{\mathcal{R}',>\varepsilon} \mathsf{top}(\mathsf{go\_up}(t_2)) \xrightarrow{i}_{\mathcal{R}',\varepsilon} \dots \qquad (\star)$$

To prove this lemma we make use of Dependency Pairs [1], a powerful technique to analyze innermost termination of TRSs. Essentially, instead of analyzing the rewrite relation $\xrightarrow{i}_{\mathcal{R}'}$ directly, one considers two TRSs $\mathcal{P}$ and $\mathcal{R}'$ in combination and investigates so called *innermost-$(\mathcal{P}, \mathcal{R}')$-chains* which are reductions of the following form.

$$s_1 \xrightarrow{i}_{\mathcal{P},\varepsilon} t_1 \xrightarrow{i}{}^{*}_{\mathcal{R}',>\varepsilon} s_2 \xrightarrow{i}_{\mathcal{P},\varepsilon} t_2 \xrightarrow{i}{}^{*}_{\mathcal{R}',>\varepsilon} s_3 \xrightarrow{i}_{\mathcal{P},\varepsilon} t_3 \xrightarrow{i}{}^{*}_{\mathcal{R}',>\varepsilon} \dots$$

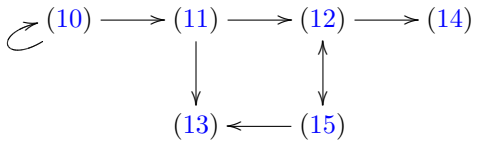
The main result of [1] states that $\mathcal{R}'$ is innermost terminating iff there are no infinite innermost-$(DP(\mathcal{R}'), \mathcal{R}')$-chains.[2] Here, $DP(\mathcal{R}')$ is the following TRS consisting of all Dependency Pairs of $\mathcal{R}'$.

[1] The relation $\xrightarrow{i}_{\mathcal{R}',>\varepsilon}$ is like $\xrightarrow{i}_{\mathcal{R}'}$ except that it is not allowed to rewrite at the root.
[2] For more details and further extensions of Dependency Pairs we refer to [1,7,11,13].

$$\mathsf{top}^\sharp(\mathsf{go\_up}(x)) \to \mathsf{top}^\sharp(\mathsf{reduce}(x)) \tag{10}$$
$$\mathsf{top}^\sharp(\mathsf{go\_up}(x)) \to \mathsf{reduce}^\sharp(x) \tag{11}$$
$$\mathsf{reduce}^\sharp(f(x_1,\ldots,x_n)) \to \mathsf{check}^\sharp_f(\mathsf{redex}_f(x_1,\ldots,x_n)) \tag{12}$$
$$\mathsf{reduce}^\sharp(f(x_1,\ldots,x_n)) \to \mathsf{redex}^\sharp_f(x_1,\ldots,x_n) \tag{13}$$
$$\mathsf{check}^\sharp_f(\mathsf{redex}_f(x_1,\ldots,x_n)) \to \mathsf{in}^\sharp_{f,i}(x_1,\ldots,\mathsf{reduce}(x_i),\ldots,x_n) \tag{14}$$
$$\mathsf{check}^\sharp_f(\mathsf{redex}_f(x_1,\ldots,x_n)) \to \mathsf{reduce}^\sharp(x_i) \tag{15}$$

We can now prove that every innermost non-terminating TRS $\mathcal{R}'$ entails an infinite reduction of the form in ($\star$).

Proof. If $\mathcal{R}'$ is not innermost terminating then there is an infinite innermost $(DP(\mathcal{R}'),\mathcal{R}')$-chain. To investigate the form of possible infinite chains we compare the root-symbols of the Dependency Pairs and obtain the following Dependency Graph [1].



This graph contains two strongly connected components $\{(10)\}$ and $\{(12),(15)\}$. Therefore, every infinite innermost chain will end in either an infinite innermost $(\{(10)\},\mathcal{R}')$- or $(\{(12),(15)\},\mathcal{R}')$-chain. We now show that there are no chains of the latter kind. The reason is that for the following polynomial order, both (12) and (15) are strictly decreasing and the only usable rules[2] (5) are weakly decreasing.

$$\begin{aligned}
[\mathsf{reduce}^\sharp](x) = [\mathsf{check}^\sharp_f](x) &= 1 + x \\
[f](x_1,\ldots,x_n) &= 1 + x_1 + \cdots + x_n \qquad \text{for all } f \in \Sigma \\
[\mathsf{redex}_f](x_1,\ldots,x_n) &= x_1 + \cdots + x_n \\
[\mathsf{result}](x) &= 0
\end{aligned}$$

Thus, there must be an infinite innermost $(\{(10)\},\mathcal{R}')$-chain. But this directly corresponds to the infinite reduction ($\star$): one just has to replace $\mathsf{top}^\sharp$ by top. □

With the help of Lemma 7 it is now possible to extract the infinite innermost reduction ($\star$) from every non-innermost terminating transformed TRS. Recall that the aim is to extract an infinite outermost reduction of the original TRS from ($\star$). The natural idea is to just take the terms $t_1, t_2, t_3, \ldots$ and then to show that these lead to an outermost reduction of $\mathcal{R}$. But here we have to face one more problem: in ($\star$) the terms $t_1, t_2, t_3, \ldots$ are terms over the extended signature Σ', but we need to extract terms over the original signature Σ. To this end we use the following mapping which always yields terms over the original signature.

Definition 8. *The mapping $\mathcal{O} : \mathcal{T}(\Sigma',\mathcal{V}) \to \mathcal{T}(\Sigma,\mathcal{V})$ is defined follows:*

- $\mathcal{O}(f(t_1,\ldots,t_n)) = f(\mathcal{O}(t_1),\ldots,\mathcal{O}(t_n))$ *for all* $f \in \Sigma$
- $\mathcal{O}(t) = x_t$, *otherwise*

For each substitution σ *the substitution* $\mathcal{O}(\sigma)$ *is defined as* $x\mathcal{O}(\sigma) = \mathcal{O}(x\sigma)$.

Before we state and prove the second main lemma, we need some properties of $\mathcal{O}$ about matching.

Lemma 9. • *If* $\ell \in \mathcal{T}(\Sigma,\mathcal{V})$ *then* $\mathcal{O}(\ell\sigma) = \ell\mathcal{O}(\sigma)$ *for all substitutions* σ.
• *If* $\mathcal{O}(t) = \ell\sigma$ *for some substitution* σ *then* $t = \ell\delta$ *for some substitution* δ.

Proof. • Straightforward structural induction on ℓ.
• First note that $\mathcal{O}$ is injective. Hence, the inverse mapping $\mathcal{O}^{-1}$ is properly defined which can again be lifted to a mapping from substitutions to substitutions where $\mathcal{O}^{-1}(\sigma)$ is defined as $x\mathcal{O}^{-1}(\sigma) = \mathcal{O}^{-1}(x\sigma)$. We show that $t = \ell\mathcal{O}^{-1}(\sigma)$ by induction on ℓ, i.e., we choose $\delta = \mathcal{O}^{-1}(\sigma)$.
If ℓ is a variable x then $\mathcal{O}(t) = x\sigma$ and thus, $t = \mathcal{O}^{-1}(\mathcal{O}(t)) = \mathcal{O}^{-1}(x\sigma) = x\mathcal{O}^{-1}(\sigma) = \ell\mathcal{O}^{-1}(\sigma)$.
Otherwise, $\ell = f(\ell_1,\ldots,\ell_n)$. Thus, $\mathcal{O}(t) = \ell\sigma = f(\ell_1\sigma,\ldots,\ell_n\sigma)$ implies that $t = f(t_1,\ldots,t_n)$, $f \in \Sigma$, and $\mathcal{O}(t) = f(\mathcal{O}(t_1),\ldots,\mathcal{O}(t_n))$. Hence, $\mathcal{O}(t_i) = \ell_i\sigma$ for all i and by induction we obtain $t_i = \ell_i\mathcal{O}^{-1}(\sigma)$. But this directly yields $t = f(\ell_1\mathcal{O}^{-1}(\sigma),\ldots,\ell_n\mathcal{O}^{-1}(\sigma)) = f(\ell_1,\ldots,\ell_n)\mathcal{O}^{-1}(\sigma) = \ell\mathcal{O}^{-1}(\sigma)$. □

We will now prove the second main lemma to achieve completeness of our transformation, namely that an innermost reduction $\mathsf{reduce}(t_i) \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}'} \mathsf{go\_up}(t_{i+1})$ in ($\star$) corresponds to an outermost reduction of the original system.

Lemma 10. *Let* $s,t \in \mathcal{T}(\Sigma',\mathcal{V})$ *where* t *is in normal form w.r.t.* $\mathcal{R}'$. *Whenever* $\mathsf{reduce}(t) \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}'} \mathsf{go\_up}(s)$ *then* $\mathcal{O}(t) \xrightarrow{\mathsf{o}}{}^*_{\mathcal{R}} \mathcal{O}(s)$ *and* s *is in normal form w.r.t.* $\mathcal{R}'$.

Proof. We perform induction on the length of the reduction. If $\mathsf{reduce}(t)$ reduces to $\mathsf{go\_up}(s)$ the first step must be done with (3) since t is in normal form. Hence, $t = f(t_1,\ldots,t_n)$ for some $f \in \Sigma$ and normal forms $t_1,\ldots,t_n$ and moreover, the reduction must start with $\mathsf{reduce}(t) \xrightarrow{\mathsf{i}}_{\mathcal{R}'} \mathsf{check}_f(\mathsf{redex}_f(t_1,\ldots,t_n))$. From the new term $\mathsf{check}_f(\mathsf{redex}_f(t_1,\ldots,t_n))$ there are only two possible ways to continue the reduction to $\mathsf{go\_up}(s)$, either using (5) or using (4).

In the former case we obtain the reduction $\mathsf{check}_f(\mathsf{redex}_f(t_1,\ldots,t_n)) \xrightarrow{\mathsf{i}}_{\mathcal{R}'} \mathsf{check}_f(\mathsf{result}(s'))$. Hence, there must be some $f(\ell_1,\ldots,\ell_n) \to r \in \mathcal{R}$ such that $t_i = \ell_i\sigma$ and $s' = r\sigma$. As σ is a normalized substitution and as $r \in \mathcal{T}(\Sigma,\mathcal{V})$ is a constructor term w.r.t. $\mathcal{R}'$, we know that s' is in normal form. Moreover, as additionally each $\ell_i \in \mathcal{T}(\Sigma,\mathcal{V})$ we can apply Lemma 9 and obtain

$$\mathcal{O}(t) = \mathcal{O}(f(\ell_1\sigma,\ldots,\ell_n\sigma)) = f(\ell_1,\ldots,\ell_n)\mathcal{O}(\sigma) \xrightarrow{\mathsf{o}}_{\mathcal{R}} r\mathcal{O}(\sigma) = \mathcal{O}(r\sigma) = \mathcal{O}(s').$$

From $\mathsf{check}_f(\mathsf{result}(s')) \xrightarrow{\mathsf{i}}{}^*_{\mathcal{R}'} \mathsf{go\_up}(s)$ and the fact that the only possible reduction of $\mathsf{check}_f(\mathsf{result}(s'))$ is via rule (6) to $\mathsf{go\_up}(s')$ we conclude that $s' = s$. Hence, by our previous results we have proven $\mathcal{O}(t) \xrightarrow{\mathsf{o}}{}^*_{\mathcal{R}} \mathcal{O}(s') = \mathcal{O}(s)$ and that $s = s'$ is in normal form. This finishes the first case.

In the second case the reduction of $\mathsf{check}_f(\mathsf{redex}_f(t_1,\ldots,t_n))$ is performed

by rule (4) which yields the new term $\mathsf{in}_{f,i}(t_1, \ldots, \mathsf{reduce}(t_i), \ldots, t_n)$.

Note that there is only one way to reach the term $\mathsf{go\_up}(s)$ from this new term: one must first reduce $\mathsf{reduce}(t_i)$ to some term $\mathsf{go\_up}(s_i)$. Since this reduction is shorter than the whole reduction we can apply the induction hypothesis to obtain $\mathcal{O}(t_i) \xrightarrow{\mathrm{o}}_{\mathcal{R}} \mathcal{O}(s_i)$ and conclude that s_i is in normal form. Moreover, there is only one way to continue the reduction towards $\mathsf{go\_up}(s)$: one has to reduce $\mathsf{in}_{f,i}(t_1, \ldots, \mathsf{go\_up}(s_i), \ldots, t_n)$ to $\mathsf{go\_up}(f(t_1, \ldots, s_i, \ldots, t_n))$ with rule (7).

However, since this last term is in normal form—$f \in \Sigma$ is a constructor of $\mathcal{R}'$—we know that s must be the normal form $f(t_1, \ldots, s_i, \ldots, t_n)$. Hence,

$$\begin{aligned}\mathcal{O}(t) &= f(\mathcal{O}(t_1), \ldots, \mathcal{O}(t_i), \ldots, \mathcal{O}(t_n)) \\ &\to_{\mathcal{R}} f(\mathcal{O}(t_1), \ldots, \mathcal{O}(s_i), \ldots, \mathcal{O}(t_n)) \\ &= \mathcal{O}(f(t_1, \ldots, s_i, \ldots, t_n)) \\ &= \mathcal{O}(s)\end{aligned}$$

It only remains to prove that the above reduction is indeed an outermost-reduction. Since we know that $\mathcal{O}(t_i) \xrightarrow{\mathrm{o}}_{\mathcal{R}} \mathcal{O}(s_i)$ we only have to prove that $\mathcal{O}(t)$ is not a redex of $\mathcal{R}$. So, suppose $\mathcal{O}(t)$ is a redex. Hence, there is some rule $\ell \to r \in \mathcal{R}$ such that $\mathcal{O}(t) = \ell\sigma$. By Lemma 9 we know that then $t = f(t_1, \ldots, t_n) = \ell\delta$ and thus, $t_i = \ell_i\delta$ for all i. Hence, the term $\mathsf{redex}_f(t_1, \ldots, t_n)$ is not in normal form w.r.t. $\mathcal{R}'$. However, this is a contradiction to the fact that $\mathsf{check}_f(\mathsf{redex}_f(t_1, \ldots, t_n))$ was innermost reduced at top-level. □

With the help of Lemmas 7 and 10 completeness of our transformation is easily established.

Corollary 11. *If $\mathcal{R}$ is outermost terminating then $\mathcal{R}'$ is innermost terminating.*

Proof. Assume that $\mathcal{R}'$ is not innermost terminating. Then by Lemma 7 there is the infinite innermost reduction

$$\mathsf{top}(\mathsf{go\_up}(t_1)) \xrightarrow{\mathrm{i}}_{\mathcal{R}',\varepsilon} \mathsf{top}(\mathsf{reduce}(t_1)) \xrightarrow{\mathrm{i}}^{*}_{\mathcal{R}',>\varepsilon} \mathsf{top}(\mathsf{go\_up}(t_2)) \xrightarrow{\mathrm{i}}_{\mathcal{R}',\varepsilon} \ldots$$

which by Lemma 10 directly leads to the infinite outermost reduction

$$\mathcal{O}(t_1) \xrightarrow{\mathrm{o}}_{\mathcal{R}} \mathcal{O}(t_2) \xrightarrow{\mathrm{o}}_{\mathcal{R}} \ldots.$$

□

6 Improved Transformation

In this section we present an improved variant of our transformation. The idea is that the check for an outermost-redex is superfluous if the outermost symbol is a constructor. Moreover, whenever a constant is reduced, then the reduction can only be at the top-level. Hence, in that case there also is no need for the check. These ideas lead to an improved transformation which produces a smaller TRS than the transformation of Def. 2. Moreover, each outermost step of the original TRS can be simulated by less innermost steps of the transformed TRS.

Definition 12 (Improved transformation of outermost TRSs). *Let $\mathcal{R}$ be a TRS over Σ. We define the* improved transformed system *$\mathcal{R}''$ over Σ' to consist of rules $\{(3) - (8)\}$ with the following additional rules.*

$$\mathsf{reduce}(f(x_1,\ldots,x_n)) \to \mathsf{in}_{f,i}(x_1,\ldots,\mathsf{reduce}(x_i),\ldots,x_n) \tag{16}$$
$$\mathsf{reduce}(f) \to \mathsf{go\_up}(r) \tag{17}$$

However, in difference to Def. 2, (3) and (6) are only for defined symbols f with $\mathrm{ar}(f) > 0$, (4) is only for defined symbols f, and (5) is only for rules $f(\ell_1,\ldots,\ell_n) \to r \in \mathcal{R}$ with $\mathrm{ar}(f) > 0$. The new rule (16) is for all constructors f and $1 \le i \le \mathrm{ar}(f)$, and the new rule (17) is for all defined constants, i.e., for all rules $f \to r \in \mathcal{R}$ where $\mathrm{ar}(f) = 0$.

To illustrate that an outermost reduction can be simulated by less innermost reductions using the improved transformation, recall the simulation of $\mathsf{zeros} \xrightarrow{o}_{\mathcal{R}} \mathsf{cons}(\mathsf{0},\mathsf{zeros})$ of Ex. 1. The transformation of Def. 2 needs four steps to reduce $\mathsf{top}(\mathsf{reduce}(\mathsf{zeros}))$ to $\mathsf{top}(\mathsf{reduce}(\mathsf{cons}(\mathsf{0},\mathsf{zeros})))$, cf. (9), whereas the improved transformation only needs two steps.

As for the transformation of Def. 2, we obtain sound- and completeness.

Theorem 13. *$\mathcal{R}$ is outermost termating iff $\mathcal{R}''$ is innermost terminating.*

This theorem is proven in the same way as Thm. 3.

7 Related Work and Experiments

Termination of programming languages with outermost strategy has also been studied in [10,16]. However, in contrast to our work their emphasis is on programs which also contain non-terminating functions, and where the goal is to prove termination of a set of starting terms.

Relating our approach to the direct technique of [12] to prove outermost termination of TRSs, we see the benefit of our approach that it can also be used to disprove outermost termination. Since the success of our approach heavily depends on the techniques to analyze innermost termination, the best way to investigate the relative proving power is by an extensive empirical comparison. Unfortunately, a fully automatic implementation of [12] is currently not available. However, a by-hand-calculation on a small set of examples has shown that in practice our technique is of incomparable power to [12], i.e., there are examples where only one of both techniques is successful.

The most similar work to ours is the transformation of [18] to prove outermost termination. Although the general structure of both transformations is similar, there are important differences. The advantage of [18] is that their transformation does not rely on innermost rewriting. Therefore, more techniques are applicable on the resulting TRSs. However, there are also some drawbacks of [18] in comparison to our transformation. First, it is unknown whether the transformation of [18] is complete. Moreover, we do not have the limitation to quasi-left-linear TRSs as in [18]. And finally, our transformation is easier to implement and cannot lead to exponentially larger resulting TRSs. Empirically, our transformation is incomparable[3] to [18].

[3] Although the detailed experiments indicate that our transformation is strictly more powerful, in practice this is not the case if one tries the transformation of [18] with several different provers as backend: then both transformations are incomparable.

For our experiments we considered only those 434 TRSs from the termination problem database[4] (TPDB 4.0) where full termination could not already be proven by the termination prover AProVE [9]. We considered three transformations to analyze outermost termination where afterwards we always used AProVE as backend. For each TRS and transformation we used a timeout of one minute. The machine was an Intel Core 2 Duo with 2.4 Ghz running under Mac OS X. The details of our experiments can be viewed at http://cl-informatik.uibk.ac.at/~thiemann/outermost. The following table gives a summary.

Transformation	$\mathcal{R}'$	$\mathcal{R}''$	[18][5]
# of TRSs where outermost termination was *disproven*	39	40	0
# of TRSs where outermost termination was *proven*	7	7	6

We first consider disproving outermost termination. Note that only 162 TRSs of the 434 TRS are detected to be non-terminating. Here we can see the success of our transformation as it is applicable on 40 TRSs whereas we know of no other method that can disprove outermost termination automatically. For the remaining 122 TRSs there are two major reasons why outermost termination cannot be disproven: first, some of these TRSs are outermost terminating.

And second, our transformation destroys looping reductions, i.e., although the original TRS might contain a looping outermost reduction, this does not necessarily correspond to a looping innermost reduction in the transformed TRS. To solve this problem, as future work we plan to develop a direct method for disproving outermost termination similar to the one in [19].

With these experiments one can also illustrate the benefit of the improved transformation. There is one TRS where only transformation $\mathcal{R}''$ is able to disprove outermost termination. Here, the failure of $\mathcal{R}'$ is just due to the length of the simulation. The looping reduction of $\mathcal{R}''$ is some steps shorter than the corresponding reduction in $\mathcal{R}'$. That this can be crucial is due to the fact that the complexity of searching for looping reductions is exponential in the length of the reduction.

Unfortunately the numbers for proving outermost termination do not look that good. However, this is mainly due to the fact that there are hardly any examples in the TPDB which are designed for outermost rewriting: there are only six.

To conclude, we have developed a successful transformation for proving and especially for disproving outermost termination of TRSs.

Acknowledgments. We thank Isabelle Gnaedig for her helpful feedback on questions about [12].

[4] Available at http://www.lri.fr/~marche/tpdb/.

[5] To be comparable we always added a fresh constant to the signature when using [18]. The reason is that otherwise [18] only proves outermost-ground termination.

References

1. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
2. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
3. A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. *Information and Computation*, 205(4):512–534, 2007.
4. J. Giesl and T. Arts. Verification of Erlang processes by dependency pairs. *Appl. Algebra in Engineering, Communication and Computing*, 12(1,2):39–72, 2001.
5. J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4):379–427, 2004.
6. J. Giesl, R. Thiemann, S. Swiderski, and P. Schneider-Kamp. Proving termination by bounded increase. In *Proc. CADE '07*, LNAI 4603, pp. 443–459, 2007.
7. J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proc. LPAR '04*, LNAI 3452, pp. 301–331, 2005.
8. J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. FroCoS '05*, LNAI 3717, 2005. 216–231.
9. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the DP framework. *Proc. IJCAR '06*, LNAI 4130, pp. 281–286, 2006.
10. J. Giesl, S. Swiderski, P. Schneider-Kamp, and R. Thiemann. Automated termination analysis for Haskell: From term rewriting to programming languages. In *Proc. RTA '06*, LNCS 4098, pp. 297–312, 2006.
11. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.
12. I. Gnaedig and H. Kirchner. Termination of rewriting under strategies. *ACM Transactions on Computational Logic*, 2008. To appear. Available at http://tocl.acm.org/accepted/315gnaedig.ps.
13. N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.
14. S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1:1–61, 1998.
15. E. Ohlebusch. Termination of logic programs: Transformational methods revisited. *Appl. Algebra in Eng., Communication and Computing*, 12(1-2):73–116, 2001.
16. S. E. Panitz and M. Schmidt-Schauss. TEA: Automatically proving termination of programs in a non-strict higher-order functional language. In *Proc. SAS '97*, LNCS 1302, pp. 345–360, 1997.
17. R. Plasmeijer and M. van Eekelen. *Functional Programming and Parallel Graph Rewriting*. Addison Wesley, 1993.
18. M. Raffelsieper and H. Zantema. A transformational approach to prove outermost termination automatically. In informal *Proc. WRS '08*, pp. 76–89, 2008. Available as technical report RISC-Linz 08-09 at http://www.risc.uni-linz.ac.at/publications/download/risc_3452/wrs2008.pdf.
19. R. Thiemann, J. Giesl, and P. Schneider-Kamp. Deciding innermost loops. In *Proc. RTA '08*, LNCS 5117, pp. 366–380, 2008.
20. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.