

Loops under Strategies

René Thiemann and Christian Sternagel*

Institute of Computer Science, University of Innsbruck, Austria
{rene.thiemann|christian.sternagel}@uibk.ac.at

Abstract. Most techniques to automatically disprove termination of term rewrite systems search for a loop. Whereas a loop implies non-termination for full rewriting, this is not necessarily the case if one considers rewriting under strategies. Therefore, in this paper we first generalize the notion of a loop to a loop under a given strategy. In a second step we present two novel decision procedures to check whether a given loop is a context-sensitive or an outermost loop. We implemented and successfully evaluated our method in the termination prover $\mathsf{T}\mathsf{T}_2$.

1 Introduction

Termination is an important property of term rewrite systems (TRSs). Therefore, much effort has been spent on developing and automating powerful techniques for showing termination of TRSs. An important application area for these techniques is termination analysis of functional programs. Since the evaluation mechanism of functional languages is mainly term rewriting, one can transform functional programs into TRSs and prove termination of the resulting TRSs to conclude termination of the functional programs [6]. Although “full” rewriting does not impose any evaluation strategy, this approach is sound even if the underlying programming language has an evaluation strategy.

But in order to detect bugs in programs, it is at least as important to prove *non-termination* of programs or of the corresponding TRSs. Here, the evaluation strategy cannot be ignored, because a non-terminating TRS may still be terminating when considering the strategy. Thus, in order to disprove termination of programming languages with strategies, it is important to develop automated techniques to disprove termination of TRSs under strategies.

Only a few techniques for showing non-termination of TRSs have been introduced so far [4,7,9,10,12]. These techniques can be used to detect loops—a specific form of derivation which implies non-termination—and are successfully implemented in many tools (e.g., AProVE[5], Jambox [2], Matchbox [16], NTI [12], TORPA [17], $\mathsf{T}\mathsf{T}_2$ [8]).

If one wants to prove non-termination under strategies then up to now there are two different approaches. The first one is to directly analyze the loops whether they also imply non-termination under a given strategy \mathcal{S} . This approach was successfully applied for the innermost strategy in [15] where a decision procedure was given to determine whether a loop is an innermost loop.

* This author is supported by FWF (Austrian Science Fund) project P18763.

The second approach is to use a complete transformation τ_S for strategy S such that \mathcal{R} is terminating under S iff $\tau_S(\mathcal{R})$ is (innermost) terminating. Then one first applies the transformation and then searches for a loop in $\tau_S(\mathcal{R})$ afterwards. Here, the methods of [3] and [13,14] are applicable which can be used to disprove context-sensitive and outermost termination.

Although the second approach of using the transformations [3,13,14] seems to be a good solution to disprove context-sensitive and outermost termination, there are two main drawbacks. The first problem is a practical one. Often the loops of \mathcal{R} are transformed into much longer loops in $\tau_S(\mathcal{R})$ and hence, the search space for loops may become critical. And even more severe is the problem, that some loops of \mathcal{R} are not even translated to loops in $\tau_S(\mathcal{R})$ and hence, one even loses power if the search problem for loops is ignored.

Thus, there is still need to extend the first approach—to ensure or even decide that a given loop is a loop under strategies—to other strategies besides innermost. To this end, in this paper we first generalize the notion of a loop and an innermost loop to a loop under some arbitrary strategy. Then we develop two new decision procedures for context-sensitive loops and outermost loops.

The paper is structured as follows. In Sect. 2 we recapitulate the required notions of rewriting and generalize the notion of a loop for rewriting strategies. Moreover, we present a decision procedure for the question whether a given loop is a context-sensitive loop. Then in Sect. 3 we show how to formulate the same question for the outermost strategy as a set of matching problems. How these matching problems can be transformed to a simpler kind of problems—identity problems—is the content of Sect. 4. Afterwards, in Sect. 5 we provide a decision procedure for solvability of identity problems. All of our techniques have been implemented in the Tyrolean Termination Tool 2 ($\mathsf{T}\mathsf{T}\mathsf{T}\mathsf{2}$) and the empirical results are presented in Sect. 6, before we conclude in Sect. 7.

A full version of this paper containing all proofs is available at <http://cl-informatik.uibk.ac.at/~griff/experiments/lus.php>. This website also contains details about our experiments.

2 Loops

We only regard finite signatures and TRSs and refer to [1] for the basics of rewriting. We use ℓ, r, s, t, u, \dots for terms, f, g, \dots for function symbols, x, y, \dots for variables, σ, μ for substitutions, i, j, k, n, m, o for natural numbers, p, q, \dots for positions where ε is the root position, and C, D, \dots for contexts. Here, contexts are terms which contain exactly one hole \square . For contexts, the term $C[t]$ is like C where \square is replaced by t , i.e., $\square[t] = t$ and $f(s_1, \dots, C, \dots, s_n)[t] = f(s_1, \dots, C[t], \dots, s_n)$. We write $t|_p$ for the subterm of t at position p , i.e., $t|_\varepsilon = t$ and $f(s_1, \dots, s_n)|_{ip} = s_i|_p$. The set of variables is denoted by \mathcal{V} .

Throughout this paper we assume a fixed TRS \mathcal{R} and we write $t \rightarrow_p s$ if one can reduce t to s at position p with \mathcal{R} , i.e., $t = C[\ell\sigma]$ and $s = C[r\sigma]$ for some $\ell \rightarrow r \in \mathcal{R}$, substitution σ , and context C with $C|_p = \square$. Here, the term $\ell\sigma$ is called a redex at position p . The reduction is an outermost reduction, written

$t \xrightarrow{\circlearrowleft}_p s$, iff t contains no redex at a position q above p (written $q < p$). If the position is irrelevant we just write \rightarrow or $\xrightarrow{\circlearrowleft}$. The TRS \mathcal{R} is non-terminating iff there is an infinite derivation $t_1 \rightarrow t_2 \rightarrow \dots$. It is outermost non-terminating iff there is such an infinite derivation using $\xrightarrow{\circlearrowleft}$ instead of \rightarrow .

An obvious approach to disprove termination is to search for a loop, i.e., a derivation where the starting term t is reduced to a term containing an instance of t , i.e., $t \rightarrow^+ C[t\mu]$. The corresponding infinite derivation is

$$t \rightarrow^+ C[t\mu] \rightarrow^+ C[C[t\mu]\mu] \rightarrow^+ \dots \rightarrow^+ C[C[\dots C[t\mu]\dots\mu]\mu] \rightarrow^+ \dots \quad (\star)$$

where the derivation $t \rightarrow^+ C[t\mu]$ is repeated over and over again. This infinite derivation (\star) is obtained because \rightarrow is closed under both substitutions and contexts, also known as stability and monotonicity.

However, in general it is not clear whether (\star) also is an infinite derivation if one considers a specific evaluation strategy \mathcal{S} . If this is the case then and only then we speak of an \mathcal{S} -loop.

To formally define an \mathcal{S} -loop we first need to make the derivations within (\star) precise. Therefore, we must represent terms like $C[C[\dots C[t\mu]\dots\mu]\mu]$ without using “ \dots ”. Moreover, we must know the positions of the reductions since several strategies—like innermost, outermost, or context-sensitive—only allow reductions at certain positions. To this end, we define the notion of a context-substitution which combines insertion into a context with the application of a substitution.

Definition 1 (Context-substitutions). A context-substitution is a pair (C, μ) consisting of a context C and a substitution μ . The n -fold application of (C, μ) to a term t , written $t(C, \mu)^n$ is defined as follows.

- $t(C, \mu)^0 = t$
- $t(C, \mu)^{n+1} = C[t(C, \mu)^n\mu]$

From the definition it is obvious that in $t(C, \mu)^n$ the context C is added n -times above t and t is instantiated by μ^n . Note that also the added contexts are instantiated by μ . For the term $t(C, \mu)^3$ this is illustrated in Fig. 1.

The following lemma shows that context-substitutions have similar properties to both contexts and substitutions.

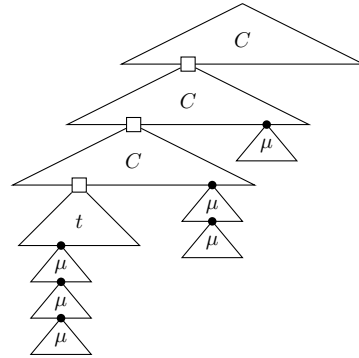


Fig. 1. The term $t(C, \mu)^3$

Lemma 2 (Properties of context-substitutions).

- (i) $t(C, \mu)^n\mu = t\mu(C\mu, \mu)^n$.
- (ii) $t(C, \mu)^m(C, \mu)^n = t(C, \mu)^{m+n}$.
- (iii) If $C|_p = \square$ then $t(C, \mu)^n|_{p^n} = t\mu^n$.
- (iv) Whenever $t \rightarrow_q s$ and $C|_p = \square$ then $t(C, \mu)^n \rightarrow_{p^n q} s(C, \mu)^n$.

Here, property (i) is similar to the fact that $C[t]\mu = C\mu[t\mu]$, and (ii) expresses that context-substitutions can be combined just as substitutions where $\sigma^m\sigma^n = \sigma^{m+n}$. Moreover, the property of contexts that $C[t]_p = t$ if $C|_p = \square$ is extended in (iii), and finally stability and monotonicity of rewriting are used to show in (iv) that rewriting is closed under context-substitutions.

With the help of context-substitutions we can now describe the infinite derivation in (\star) more concisely. Since $t \rightarrow^+ C[t\mu] = t(C, \mu)$ we obtain

$$t(C, \mu)^0 \rightarrow^+ t(C, \mu)(C, \mu)^0 = t(C, \mu)^1 \rightarrow^+ \dots \rightarrow^+ t(C, \mu)^n \rightarrow^+ \dots \quad (\star\star)$$

Hence, the terms that occur during the derivation are precisely defined and for every n the positions of the reductions are prefixed by an additional p^n where p is the position of the hole in C , cf. Lemma 2 (iv). In other words, every reduction takes place at the same position of the subterm $t\mu^n$ of $t(C, \mu)^n$.

Now it is natural to define that a derivation $t \rightarrow^+ t(C, \mu)$ is called an \mathcal{S} -loop iff all steps in $(\star\star)$ respect the strategy \mathcal{S} .¹

Definition 3 (\mathcal{S} -loops). *Let \mathcal{S} be a strategy. A loop $t_1 \rightarrow_{q_1} t_2 \rightarrow_{q_2} \dots t_n \rightarrow_{q_n} t_{n+1} = t_1(C, \mu)$ with $C|_p = \square$ is an \mathcal{S} -loop iff all reductions $t_i(C, \mu)^m \rightarrow_{p^m q_i} t_{i+1}(C, \mu)^m$ respect the strategy \mathcal{S} for all $1 \leq i \leq n$ and $m \geq 0$.*

As a direct consequence of Def. 3 one can conclude that every \mathcal{S} -loop of a rewrite system \mathcal{R} proves non-termination of \mathcal{R} under strategy \mathcal{S} .

Example 4. We consider the TRS \mathcal{R}_n for arithmetic with n -bit numbers.

$$\mathfrak{p}(0) \rightarrow 0 \quad (1) \quad \text{plus}(0, y) \rightarrow y \quad (6)$$

$$\mathfrak{p}(s(x)) \rightarrow x \quad (2) \quad \text{plus}(s(x), y) \rightarrow s(\text{plus}(x, y)) \quad (7)$$

$$\text{minus}(x, 0) \rightarrow x \quad (3) \quad \text{inf} \rightarrow s(\text{inf}) \quad (8)$$

$$\text{minus}(x, x) \rightarrow 0 \quad (4) \quad s^{2^n}(x) \rightarrow \text{overflow} \quad (9)$$

$$\text{minus}(x, s(y)) \rightarrow \mathfrak{p}(\text{minus}(x, y)) \quad (5)$$

Here, the last rule is used to model that an overflow occurred due to the n -bit restriction. We focus on the loops

$$t_1 = \text{minus}(x, \text{inf}) \rightarrow \text{minus}(x, s(\text{inf})) \rightarrow \mathfrak{p}(\text{minus}(x, \text{inf})) = C_1(t_1, \mu_1) \quad \text{and}$$

$$t_2 = \text{plus}(\text{inf}, y) \rightarrow \text{plus}(s(\text{inf}), y) \rightarrow s(\text{plus}(\text{inf}, y)) = C_2(t_2, \mu_2)$$

where $\mu_1 = \mu_2 = \{\}$, $C_1 = \mathfrak{p}(\square)$, and $C_2 = s(\square)$. Here, the first loop is an outermost loop, but the second one is not. The reason for the latter is that in every iteration one more s is created. Hence, this will lead to a redex w.r.t. Rule (9). Note that this example will be hard to handle with the transformational approaches: using [14] creates a TRS where all infinite reductions are non-looping and [13] is not even applicable due to non-left-linearity of Rule (4).

¹ Another natural definition of an \mathcal{S} -loop would just require that $t(C, \mu)^n \rightarrow^+ t(C, \mu)^{n+1}$ are \mathcal{S} -derivations for all n . This alternative was already used in the setting of dependency pairs in [4, Footnote 6]. However, there are problems using this definition which are described in [15, Sect. 2].

Note that a loop is not only determined by the precise derivation $t_1 \rightarrow^+ t_{n+1}$, but also by the specific context C that is used. To see this consider the TRS $\mathcal{R} = \{a \rightarrow f(a, a), f(f(x, y), z) \rightarrow b\}$. Then $a \rightarrow f(a, a)$ is a looping derivation. For $C = f(a, \square)$ this is an outermost loop. However, for $C' = f(\square, a)$ we do not obtain an outermost loop since $f(a, a) \xrightarrow{o} f(f(a, a), a) \not\xrightarrow{o} f(f(f(a, a), a), a)$. Hence, the choice of C is essential.

For automatic non-termination analysis under strategies one main question is whether a given loop is an \mathcal{S} -loop, i.e., whether the loop implies non-termination even under strategy \mathcal{S} . In [15] it was already shown that this question is decidable for *innermost loops*.² There, one has the problem that innermost rewriting is not stable, although it is monotonic.

In context-sensitive rewriting [11] we have the inverse situation: first, stability is given whereas monotonicity is absent. And second, whereas the decision procedure for innermost loops is quite involved and already known, for *context-sensitive loops* we can present a novel decision procedure that is rather straightforward, but nevertheless important.

Theorem 5 (Deciding context-sensitive-loops). *A loop $t \rightarrow^+ C[t\mu]$ is a context-sensitive loop (using replacement map ν) iff both the derivation $t \rightarrow^+ C[t\mu]$ respects the context-sensitive strategy and the hole in C is at a ν -replacing position.*

Proof. Let $t_1 \rightarrow_{q_1} t_2 \rightarrow_{q_2} \dots t_n \rightarrow_{q_n} t_{n+1} = t_1(C, \mu)$ be a loop where $C|_p = \square$. Then the following statements are all equivalent.

- the loop is a context-sensitive loop
- all $t_i(C, \mu)^m \rightarrow_{p^m q_i} t_{i+1}(C, \mu)^m$ are context-sensitive reductions
- all $p^m q_i$ are ν -replacing positions of $t_i(C, \mu)^m$
- p is a ν -replacing position of C and each q_i is a ν -replacing position of $t_i \mu^m$
- the hole in C is at a ν -replacing position and the derivation $t_1 \rightarrow^+ t_1(C, \mu)$ is a context-sensitive derivation □

In the rest of this paper we consider the outermost strategy. As main result we develop a decision procedure for the question whether a given loop is an *outermost loop*. Note that for outermost rewriting neither stability nor monotonicity are given. To see this consider the TRS $\mathcal{R} = \{a \rightarrow a, f(x) \rightarrow x, g(f(a)) \rightarrow a\}$. Then $a \xrightarrow{o} a$, but $f(a) \not\xrightarrow{o} f(a)$. Moreover, $g(f(x)) \xrightarrow{o} g(x)$, but $g(f(a)) \not\xrightarrow{o} g(a)$.

The problem of missing stability was already present for innermost loops. Therefore, many techniques of [15] for innermost loops can be reused for outermost loops, too. However, to handle the missing monotonicity of outermost rewriting we have to extend these techniques by an additional context. And these contexts will require significant extensions of the techniques of [15] and are not so easy to treat as in the context-sensitive case.

² Note that in [15] one did not regard contexts, i.e., for an innermost loop one just required that all reductions $t_i \mu^m \rightarrow_{q_i} t_{i+1} \mu^m$ are innermost reductions. However, that definition of an innermost loop is equivalent to Def. 3 since innermost rewriting (denoted by $\overset{i}{\rightarrow}$) is monotonic. Thus, $t_i \mu^m \overset{i}{\rightarrow}_{q_i} t_{i+1} \mu^m$ iff $t_i(C, \mu)^m \overset{i}{\rightarrow}_{p^m q_i} t_{i+1}(C, \mu)^m$.

3 Deciding Outermost Loops

Recall the definition of an outermost reduction. An outermost reduction of t at position p requires that there is no redex at a position q above p , i.e., all subterms $t|_q$ with $q < p$ must not be *matched* by some left-hand side of a rule in \mathcal{R} . Hence, the question of an outermost reduction can be formulated as a question of matching.

However, we do not have to consider a single outermost reduction but we want to know whether each reduction of a term $t(C, \mu)^m$ at position $p^m q$ is an outermost reduction (where $C|_p = \square$ and $q \in \text{Pos}(t)$). Looking at Fig. 1 one sees that there are two different cases how to obtain a subterm at a position above $p^m q$ that is matched by some left-hand side. First, the subterm may be a subterm of $t\mu^m$. Or otherwise, the subterm starts within the context. For the former case we can reuse the so called *matching problems* [15, Def. 12] and for the latter we need an extended version of matching problems containing contexts.

Definition 6 ((Extended) matching problems). A matching problem is a pair (\mathcal{M}, μ) where \mathcal{M} is a set of pairs of terms $s \succ \ell$. It is solvable iff there is a solution (k, σ) such that for all $s \succ \ell \in \mathcal{M}$ the equation $s\mu^k = \ell\sigma$ is satisfied.

An extended matching problem is a quintuple $(D \succ \ell, C, t, \mathcal{M}, \mu)$. It is solvable iff there is a solution (n, k, σ) such that the equation $D[t(C, \mu)^n]\mu^k = \ell\sigma$ is satisfied and (k, σ) is a solution to the matching problem (\mathcal{M}, μ) .

To simplify presentation we write $(D \succ \ell, C, t, \mu)$ instead of $(D \succ \ell, C, t, \emptyset, \mu)$ and we write $(s \succ \ell, \mu)$ instead of $(\{s \succ \ell\}, \mu)$. Moreover, we use the notion “matching problem” also for extended matching problems.

To check whether $t(C, \mu)^m$ has a redex above position $p^m q$ one can now construct a set of *initial matching problems*. Essentially, one considers matching problems for the subterms of t above q . Additionally, for each subterm of $t(C, \mu)^m$ that starts with a subcontext $C|_{p'}$ of C , we build an extended matching problem.

Definition 7 (Initial matching problems). Let $t \rightarrow_q u$ be a reduction and (C, μ) be a context-substitution with $C|_p = \square$. Then the following initial matching problems are created for this reduction and context-substitution.

- $(t|_{p'} \succ \ell, \mu)$ for each $\ell \rightarrow r \in \mathcal{R}$ and $p' < q$
- $(C|_{p'} \succ \ell, C\mu, t\mu, \mu)$ for each $\ell \rightarrow r \in \mathcal{R}$ and $p' < p$

Example 8. Consider the loop $t_1 = \text{minus}(x, \text{inf}) \rightarrow_2 \text{minus}(x, \text{s}(\text{inf})) = t_2 \rightarrow_\varepsilon \text{p}(\text{minus}(x, \text{inf})) = t_1(C, \mu)$ of Ex. 4 where $C = \text{p}(\square)$ and $\mu = \{\}$. For the second reduction at root position we only build the extended matching problems $MP_{1\ell} = (\text{p}(\square) \succ \ell, \text{p}(\square), \text{minus}(x, \text{s}(\text{inf})), \mu)$ for all left-hand sides ℓ of \mathcal{R} . For the first reduction we obtain the similar extended matching problems $MP_{2\ell} = (\text{p}(\square) \succ \ell, \text{p}(\square), \text{minus}(x, \text{inf}), \mu)$, but additionally we also get the matching problems $MP_{3\ell} = (\text{minus}(x, \text{inf}) \succ \ell, \mu)$.

The following theorem states that we have setup the right initial matching problems. If we consider all initial problems of all reductions $t_i \rightarrow_{q_i} t_{i+1}$ of a

loop, then solvability of one of these problems is equivalent to the property that the loop is not outermost.

Theorem 9 (Outermost loops and matching problems). *Let $t \rightarrow_q u$ and (C, μ) be given such that $C|_p = \square$. All reductions $t(C, \mu)^m \rightarrow_{p^m q} u(C, \mu)^m$ are outermost iff none of the initial matching problems for $t \rightarrow_q u$ and (C, μ) is solvable.*

Proof. We first prove that any solvable initial matching problem shows that there is at least one reduction of $t(C, \mu)^m$ at position $p^m q$ which is not an outermost reduction. There are two cases. First, if $(t|_{p'} \succ \ell, \mu)$ is solvable, then there is a solution (k, σ) such that $t|_{p'} \mu^k = \ell \sigma$. Then $t(C, \mu)^k|_{p^k p'} = t \mu^k|_{p'} = t|_{p'} \mu^k = \ell \sigma$ shows that there is a redex in $t(C, \mu)^k$ above $p^k q$. Thus, $t(C, \mu)^k \rightarrow_{p^k q} u(C, \mu)^k$ is not an outermost reduction.

Otherwise, $(C|_{p'} \succ \ell, C\mu, t\mu, \mu)$ is solvable. Hence, there is a solution (n, k, σ) such that $C|_{p'} [t\mu(C\mu, \mu)^n] \mu^k = \ell \sigma$. Here, we show that the term $t(C, \mu)^{n+1+k}$ has a redex at position $p^k p'$ which is above position $p^{n+1+k} q$:

$$\begin{aligned} t(C, \mu)^{n+1+k}|_{p^k p'} &= t(C, \mu)^n (C, \mu) (C, \mu)^k|_{p^k p'} = t(C, \mu)^n (C, \mu) \mu^k|_{p'} \\ &= C[t(C, \mu)^n \mu] \mu^k|_{p'} = C|_{p'} [t(C, \mu)^n \mu] \mu^k \\ &= C|_{p'} [t\mu(C\mu, \mu)^n] \mu^k = \ell \sigma \end{aligned}$$

For the other direction we show that if some reduction $t(C, \mu)^m$ at position $p^m q$ is not an outermost reduction, then one of the initial matching problems must be solvable. So suppose, the reduction of $t(C, \mu)^m$ is not outermost. Then there must be some position $q' < p^m q$ such that the corresponding subterm $t(C, \mu)^m|_{q'}$ is a redex $\ell \sigma$. Again, there are two cases.

First, if $q' \geq p^m$ then $q' = p^m p'$ where $p' < q$ as $q' < p^m q$. Hence, $\ell \sigma = t(C, \mu)^m|_{q'} = t(C, \mu)^m|_{p^m p'} = t \mu^m|_{p'} = t|_{p'} \mu^m$. Thus, the initial matching problem $(t|_{p'} \succ \ell, \mu)$ has the solution (m, σ) .

In the other case $q' < p^m$. Thus, we can split the position q' into $p^k p'$ where $k < m$ and $p' < p$. Moreover, there must be some $n \in \mathbb{N}$ that satisfies $m = n + 1 + k$. We conclude

$$\begin{aligned} \ell \sigma &= t(C, \mu)^m|_{q'} = t(C, \mu)^{n+1+k}|_{p^k p'} \\ &= t(C, \mu)^n (C, \mu) (C, \mu)^k|_{p^k p'} = t(C, \mu)^n (C, \mu) \mu^k|_{p'} \\ &= C[t(C, \mu)^n \mu] \mu^k|_{p'} = C[t\mu(C\mu, \mu)^n] \mu^k|_{p'} \\ &= C|_{p'} [t\mu(C\mu, \mu)^n] \mu^k. \end{aligned}$$

Thus, the initial matching problem $(C|_{p'} \succ \ell, C\mu, t\mu, \mu)$ is solvable. \square

Note that whenever $C = \square$ then there is no initial matching problem which is an extended matching problem. Hence, by Thm. 9 one can already decide whether a loop $t \rightarrow^+ t\mu$ is an outermost loop by using the techniques of [15] to decide solvability of matching problems. For example it can be detected that all matching problems $MP_{3\ell}$ of Ex. 8 are not solvable.

However, in the general case we also generate extended matching problems. Therefore, in the next section we develop a novel decision procedure for solvability of extended matching problems like $MP_{1\ell}$ and $MP_{2\ell}$ of Ex. 8.

4 Deciding Solvability of Extended Matching Problems

Since extended matching problems are only generated if $C \neq \square$, in the following sections we always assume that $C \neq \square$. We take a similar approach to [15] where we transform each matching problem into \top , \perp , or into *solved form*. Here \top and \perp represent solvability and non-solvability. And if a matching problem is in solved form then often solvability can immediately be decided. We explain all transformation rules in detail directly after the following definition.

Definition 10 (Transformation of extended matching problems). *Let $MP = (D \succ \ell_0, C, t, \mathcal{M}, \mu)$ be an extended matching problem where $\mathcal{M} = \{s_1 \succ \ell_1, \dots, s_m \succ \ell_m\}$. Then MP is in solved form iff each ℓ_i is a variable. Let $\mathcal{V}_{incr} = \{x \in \mathcal{V} \mid \exists n : x\mu^n \notin \mathcal{V}\}$ be the set of increasing variables.*

We define a relation \Rightarrow which simplifies extended matching problems that are not in solved form. So, let $\ell_j = f(\ell'_1, \dots, \ell'_{m'})$.

- (i) $MP \Rightarrow (D_{i'} \succ \ell'_{i'}, C, t, \mathcal{M} \cup \{t_i \succ \ell'_i \mid 1 \leq i \leq m', i \neq i'\}, \mu)$ if $j = 0$ and $D = f(t_1, \dots, D_{i'}, \dots, t_{m'})$.
- (ii) $MP \Rightarrow (D \succ \ell_0, C, t, (\mathcal{M} \setminus \{s_j \succ \ell_j\}) \cup \{t_i \succ \ell'_i \mid 1 \leq i \leq m'\}, \mu)$ if $j > 0$ and $s_j = f(t_1, \dots, t_{m'})$.
- (iii) $MP \Rightarrow \perp$ if $j = 0$ and $D = g(\dots)$ where $f \neq g$.
- (iv) $MP \Rightarrow \perp$ if $j > 0$ and $s_j = g(\dots)$ where $f \neq g$.
- (v) $MP \Rightarrow \perp$ if $j > 0$ and $s_j \in \mathcal{V} \setminus \mathcal{V}_{incr}$.
- (vi) $MP \Rightarrow (D\mu \succ \ell_0, C\mu, t\mu, \{s_i\mu \succ \ell_i \mid 1 \leq i \leq m\}, \mu)$ if $j > 0$ and $s_j \in \mathcal{V}_{incr}$.
- (vii) $MP \Rightarrow \top$ if $j = 0$, $D = \square$, and $(\mathcal{M} \cup \{t \succ \ell_0\}, \mu)$ is solvable.
- (viii) $MP \Rightarrow \perp$ if $j = 0$, $D = \square$, and $(\mathcal{M} \cup \{t \succ \ell_0\}, \mu)$ is not solvable.

Recall that $MP = (D \succ \ell_0, C, t, \{s_1 \succ \ell_1, \dots, s_m \succ \ell_m\}, \mu)$ is solvable iff there is a solution (n, k, σ) such that $D[t(C, \mu)^n]\mu^k = \ell_0\sigma$ and $s_i\mu^k = \ell_i\sigma$ for all $1 \leq i \leq m$. Hence, whenever $D \neq \square$ or $s_i \notin \mathcal{V}$ then one can perform a decomposition (Rules (i) and (ii)) or detect a clash (Rules (iii) and (iv)) as in a standard matching algorithm.

If $s_j = x$ is a non-increasing variable then $s_j\mu^k$ will always be a variable. Thus, Rule (v) correctly returns \perp . But if $s_j = x$ is an increasing variable then there might be a solution if $k > 0$. Hence, one can just apply μ once on the whole matching problem using Rule (vi). Note that in the result of Rule (vi) both t and C are also instantiated. This reflects the property of context-substitutions that $t(C, \mu)^n\mu = t\mu(C\mu, \mu)^n$, cf. Lemma 2.

Whereas Rule (v) and a simplified version of Rule (vi) have already been present in [15], here we also need two additional rules to handle contexts. Note that for $n = 0$ and $D = \square$ the term $D[t(C, \mu)^n]\mu^k$ is just $t\mu^k$ and thus, one only has to consider a non-extended matching problem. Now, in Rules (vii) and (viii) there is a case distinction whether this non-extended matching problem is solvable, i.e., whether $n = 0$ yields a solution or not. If it is solvable then also a solution of MP is found and Rule (vii) correctly returns \top . If it is not possible

then there is only one way to continue: apply the context-substitution at least once, and this is exactly what Rule (viii) does.

Before we formally state the soundness of the transformation rules in Thm. 12 we illustrate their application on the extended matching problems of Ex. 8.

Example 11. We first consider $MP_{1\ell} = (\mathfrak{p}(\square) \succ \ell, \mathfrak{p}(\square), \text{minus}(x, \mathfrak{s}(\text{inf})), \mu)$. If ℓ is not one of the left-hand sides $\mathfrak{p}(0)$ or $\mathfrak{p}(\mathfrak{s}(x))$ then \perp is obtained by Rule (iii). If one considers $\ell = \mathfrak{p}(0)$ then $(\mathfrak{p}(\square) \succ \mathfrak{p}(0), \mathfrak{p}(\square), \text{minus}(x, \mathfrak{s}(\text{inf})), \mu) \Rightarrow (\square \succ 0, \mathfrak{p}(\square), \text{minus}(x, \mathfrak{s}(\text{inf})), \mu)$ by Rule (i). And as $(\text{minus}(x, \mathfrak{s}(\text{inf})) \succ 0, \mu)$ is not solvable, Rule (viii) yields $(\mathfrak{p}(\square) \succ 0, \mathfrak{p}(\square), \text{minus}(x, \mathfrak{s}(\text{inf})), \mu)$. Finally, an application of Rule (iii) returns \perp and thereby shows that the matching problem is not solvable. Since the transformation for $\ell = \mathfrak{p}(\mathfrak{s}(x))$ also results in \perp , we have detected that none of the matching problems $MP_{1\ell}$ is solvable.

A similar transformation shows that none of the matching problems $MP_{2\ell}$ is solvable. Hence, the loop of Ex. 8 is an outermost loop.

Theorem 12 (Soundness and termination of the transformation rules).

- (i) If $MP \Rightarrow \perp$ then MP is not solvable.
- (ii) If $MP \Rightarrow \top$ then MP is solvable.
- (iii) If $MP \Rightarrow MP'$ then MP is solvable iff MP' is solvable.
- (iv) The relation \Rightarrow is terminating and confluent.³

Using the above theorem allows us to transform any initial matching problem into \perp , \top , or into a matching problem in solved form. In the first two cases solvability is decided, but in the last case we still need a way to extract solvability. Note that these resulting matching problems are all of the form $MP = (D \succ x_0, C, t, \{s_1 \succ x_1, \dots, s_m \succ x_m\}, \mu)$ where each $x_i \in \mathcal{V}$. Note that if all x_i are different—which is always the case if one considers left-linear TRSs—then MP is trivially solvable. One just can choose the solution (n, k, σ) where $n = k = 0$ and $\sigma = \{x_0/D[t], x_1/s_1, \dots, x_m/s_m\}$.

The only problem arises if for some $i \neq j$ we have $x_i = x_j$. Then to choose $\sigma(x_i) = \sigma(x_j)$ one has to know that $s_i\mu^k = s_j\mu^k$ for some k . This so called *identity problem* already occurred in [15]. However, if $i = 0$ then we have to answer a more difficult question, namely whether $D[t(C, \mu)^n]\mu^k = s_j\mu^k$. This new kind of problem is introduced as *extended identity problem*.

Definition 13 ((Extended) identity problems). An identity problem is a pair $(s \approx s', \mu)$. It is solvable iff there is some k such that $s\mu^k = s'\mu^k$.

An extended identity problem is a quadruple $(D \approx s, \mu, C, t)$. It is solvable iff there is a solution (n, k) such that $D[t(C, \mu)^n]\mu^k = s\mu^k$.

We now can transform matching problems in solved form into an equivalent set of (extended) identity problems.

Theorem 14 (Transforming matching problems into identity problems).

Let $MP = (D \succ x, C, t, \{s_1 \succ x_1, \dots, s_m \succ x_m\}, \mu)$ be a matching problem in solved form. It is solvable iff each of the following identity problems is solvable.

³ Here we need the assumption $C \neq \square$. Otherwise, Rule (viii) would not terminate.

- $(D \approx s_i, \mu, C, t)$ where i is the least index such that $x = x_i$.
- $(s_i \approx s_j, \mu)$ for all j where $i < j$ is the least index such that $x_i = x_j$.

One might wonder why this theorem is sound as each solution of $(s_i \approx s_j, \mu)$ might yield a different k_{ij} . The key point is that the maximum of all these k_{ij} 's is a solution for all identity problems $(s_i \approx s_j, \mu)$.

Note that [15] describes a decision procedure for solvability of identity problems $(s \approx s', \mu)$. Hence, we can already decide solvability of matching problems $(D \triangleright x, C, t, \mathcal{M}, \mu)$ in solved form where the variable x does not occur in \mathcal{M} . Nevertheless, for the general case we still need a technique to decide solvability of extended identity problems. Such a technique is described in the next section.

Example 15. Consider the TRS $\{f(x) \rightarrow g(g(x, x), f(s(x))), g(y, y) \rightarrow a\}$ with the loop $t = f(x) \rightarrow g(g(x, x), f(s(x))) = t(C, \mu)$ where $\mu = \{x/s(x)\}$ and $C = g(g(x, x), \square)$. One initial matching problem $(g(g(x, x), \square) \triangleright f(x), C\mu, t\mu, \mu)$ is trivially not solvable due to a symbol clash. But the other initial matching problem $(g(g(x, x), \square) \triangleright g(y, y), C\mu, t\mu, \mu)$ is transformed into the matching problem $MP = (\square \triangleright y, C\mu, t\mu, \{g(x, x) \triangleright y\}, \mu)$. By Thm. 14 solvability of MP is equivalent to solvability of the extended identity problem $(\square \approx g(x, x), \mu, C\mu, t\mu)$.

5 Deciding Solvability of Extended Identity Problems

In this section we describe a decision procedure for solvability of extended identity problems. To this end we first introduce the notion of a trace.

Definition 16 (Traces). *The trace of term t w.r.t. position p is the sequence of function symbols and indices that are passed when moving from ε to p in t :*

$$\text{trace}(p, t) = \begin{cases} \varepsilon & \text{if } t = x \text{ or } p = \varepsilon \\ f i \text{ trace}(q, t_i) & \text{if } p = iq \text{ and } t = f(t_1, \dots, t_n) \end{cases}$$

The trace of a context C with $C|_p = \square$ is $\text{trace}(C) = \text{trace}(p, C)$. The set of all traces of a term is $\text{Traces}(t) = \{\text{trace}(p, t) \mid p \in \mathcal{Pos}(t)\}$.

Lemma 17 (Properties of traces).

- (i) $\text{trace}(pq, C[t]) = \text{trace}(C)\text{trace}(q, t)$ if $C|_p = \square$
- (ii) $\text{trace}(p, t) = \text{trace}(p, t\mu)$ if $p \in \mathcal{Pos}(t)$
- (iii) $\text{trace}(p^n q, t(C, \mu)^n) = \text{trace}(C)^n \text{trace}(q, t\mu^n)$ if $C|_p = \square$ and $q \in \mathcal{Pos}(t)$
- (iv) $\text{trace}(p, t) \in \text{Traces}(t)$ if $\text{trace}(pq, t) \in \text{Traces}(t)$

In the following algorithm to decide solvability of extended identity problems, a Boolean disjunction over non-extended identity problems represents solvability of at least one of these identity problems.

Definition 18 (Decision procedure for extended identity problems).

Let $(D \approx s, \mu, C, t)$ be an extended identity problem where $D|_q = \square$, $C|_p = \square$.

- (i) if $\text{trace}(D)\text{trace}(C)^* \not\subseteq \bigcup_{i \in \mathbb{N}} \text{Traces}(s\mu^i) =: S$ then there is some m such that $\text{trace}(D)\text{trace}(C)^m \notin S$; return $\bigvee_{n < m} (D[t(C, \mu)^n] \approx s, \mu)$
- (ii) if $\text{trace}(C)^* \not\subseteq \bigcup_{i \in \mathbb{N}} \text{Traces}(t\mu^i)$ then return “not solvable”
- (iii) let x be a variable which infinitely often occurs in $s|_{p_0}, s\mu^1|_{p_1}, s\mu^2|_{p_2}, \dots$ where each p_i is that prefix of qp^ω which satisfies $p_i \in \mathcal{Pos}(s\mu^i)$; let i be the minimal number such that $s\mu^i|_{p_i} = x$
- (iv) let j be minimal such that $t\mu^j|_{q_j} = x$ where q_j is that prefix of p^ω which satisfies $q_j \in \mathcal{Pos}(t\mu^j)$; if there is no such j then return “not solvable”
- (v) return $\bigvee_{n \leq \max(\frac{|p_i| - |qq_j|}{|p|}, j)} (D[t(C, \mu)^n] \approx s, \mu)$

We will explain the algorithm in detail within the proof of the following theorem. Afterwards, we present algorithms to automate the non-trivial steps.

Theorem 19. *The algorithm of Def. 18 is sound and terminates.*

Proof. Termination of the algorithm is obvious. We only remark that the disjunction in Step (v) is finite, since $|p| > 0$ by the assumption $C \neq \square$.

To show soundness of the algorithm first recall the definition of solvability of $(D \approx s, \mu, C, t)$. This extended identity problem is solvable iff there is a solution (n, k) such that $D[t(C, \mu)^n]\mu^k = s\mu^k$. We observe two properties: first, whenever (n, k) is a solution then $(n, k + k')$ is also a solution. And second, if we fix n then the extended identity problem is solvable iff the identity problem $(D[t(C, \mu)^n] \approx s, \mu)$ is solvable. From the second observation we conclude that if one can bound the value of n , then one can reduce solvability of extended identity problems to solvability of identity problems and is done. And computing these bounds on n is basically all the algorithm does (in Steps (i) and (v)).

The first idea to extract a bound is to consider how the term $D[t(C, \mu)^n]\mu^k$ grows if n is increased. Looking at Fig. 1 on page 3 or using Lemma 17 we see that $D[t(C, \mu)^n]\mu^k$ has the trace $\text{trace}(D)\text{trace}(C)^n$. Thus, if (n, k) is a solution then $s\mu^k$ must have the same trace. Hence, if $\text{trace}(D)\text{trace}(C)^m \notin \bigcup_{i \in \mathbb{N}} \text{Traces}(s\mu^i) = S$ then $n < m$. This proves soundness of Step (i).

So, after Step (i) we can assume $\text{trace}(D)\text{trace}(C)^* \subseteq S$. Hence, if we increase the k of $s\mu^k$ then this term grows along the (infinite) trace $\text{trace}(D)\text{trace}(C)^\omega$. Using the first observation we know that for every solution of the extended identity problem we can increase k arbitrarily. Thus, $D[t(C, \mu)^n]\mu^k$ (which is the same term as $s\mu^k$) also has to contain longer and longer parts of the trace $\text{trace}(D)\text{trace}(C)^\omega$ when increasing k . Hence, whenever the extended identity problem is solvable then $\text{trace}(C)^* \subseteq \bigcup_{i \in \mathbb{N}} \text{Traces}(t\mu^i) =: T$ which shows soundness of Step (ii).

If the decision procedure arrives at Step (iii) then both $\text{trace}(D)\text{trace}(C)^* \subseteq S$ and $\text{trace}(C)^* \subseteq T$, i.e., when increasing k we see no difference of function symbols of the terms $D[t(C, \mu)^n]\mu^k$ and $s\mu^k$ along the path qp^ω . However, there still might be a difference between $D[t(C, \mu)^n]\mu^k$ and $s\mu^k$ for each finite value k . For example, different variables may be used to increase the terms along the path qp^ω as in $D = \square, C = f(\square), t = x, s = y, \mu = \{x/f(x), y/f(y)\}$. Or one of the terms is always a bit larger as the other one as in $D = \square, C = f(\square), t =$

$f(x), s = x, \mu = \{x/f(x)\}$. To detect the situation of different variables, Step (iv) is used, and the latter situation is done via Step (v).

As we are interested in the variables in Steps (iv) and (v), we first compute a variable x in Step (iii) which infinitely often occurs along the path qp^ω in the terms $s, s\mu, s\mu^2, \dots$. This variable must exist, since μ has finite domain and $\text{trace}(D)\text{trace}(C)^* \subseteq S$. This immediately proves soundness of Step (iv) since whenever $D[t(C, \mu)^n]\mu^k = s\mu^k$ then by the first observation we can choose k high enough such that $x = s\mu^k|_{p_k} = D[t(C, \mu)^n]\mu^k|_{p_k}$ where $p_k \leq qp^\omega$, i.e., p_k must be of the form qp^nq' where q' is a prefix of p^ω . Thus, $x = D[t(C, \mu)^n]\mu^k|_{qp^nq'} = t(C, \mu)^n\mu^k|_{p^nq'} = t\mu^{n+k}|_{q'}$ shows that Step (iv) cannot stop the algorithm with “not solvable”.

The main idea of Step (v) is as in Step (i) to bound n but now for a different reason. Observe that whenever we increase n then $s\mu^k$ stays the same whereas $D[t(C, \mu)^n]\mu^k$ has the subterm $t\mu^{n+k}$ which depends on n . And since $\text{trace}(C)^* \subseteq T$ we know that with larger n also the terms $t\mu^{n+k}$ become larger. Thus, there must be a limit where the size of $s\mu^k$ is reached and it is of no use to search for larger values of n . And this limit turns out to be $m := \max(\frac{|p_i| - |qq_i|}{|p|}, j)$ which proves soundness of Step (v). \square

Input: D, C, s, μ

Output: minimal m such that $\text{trace}(D)\text{trace}(C)^m \notin \bigcup_{i \in \mathbb{N}} \text{Traces}(s\mu^i)$ or ∞ , otherwise

- (1) $m := 0, E := D, t := s, S := \emptyset$
- (2) if $E = f(\dots E' \dots)$ and $t = f(\dots)$ where $E|_i = E'$ then $E := E', t := t|_i$, goto (2)
- (3) if $E = g(\dots), t = f(\dots)$, and $g \neq f$ then return m
- (4) if $E \neq \square$ and $t = x \notin \mathcal{V}_{\text{incr}}(\mu)$ then return m
- (5) if $E \neq \square$ and $t = x \in \mathcal{V}_{\text{incr}}(\mu)$ then $t := t\mu$, goto (2)
- (6) if $E = \square$ and $t \in S$ then return ∞
- (7) if $E = \square$ and $t \notin S$ then $S := S \cup \{t\}, m := m + 1, E := C$, goto (2)

Fig. 2. $\text{clash}(D, C, s, \mu)$.

For the automation one can use the algorithm of [15] for solvability of non-extended identity problems. To check whether $\text{trace}(D)\text{trace}(C)^* \subseteq S$ in Step (i) one can check whether $\text{clash}(D, C, s, \mu) = \infty$ (cf. Fig. 2) which also delivers the required number m in case that $\text{trace}(D)\text{trace}(C)^* \not\subseteq S$. Of course, clash can also be used for the test in Step (ii) where we call $\text{clash}(\square, C, t, \mu)$.

Theorem 20. *The algorithm clash terminates and is sound.*

For Step (iii) of the decision procedure the function $\text{var}_\infty(s, q, p, \mu)$ in Fig. 3 computes a triple (x, p_i, i) such that x infinitely often occurs in the sequence $s|_{p_0}, s\mu|_{p_1}, s\mu^2|_{p_2}, \dots$ where each $p_k \in \mathcal{Pos}(s\mu^k)$ is the maximal prefix of qp^∞ and i is the smallest number such that $s\mu^i|_{p_i} = x$. Note that the precondition is

satisfied, since var_∞ is only called if $\text{trace}(D)\text{trace}(C)^* \subseteq \bigcup_{i \in \mathbb{N}} \text{Traces}(s\mu^i)$ which directly implies $qp^* \subseteq \bigcup_{i \in \mathbb{N}} \text{Pos}(s\mu^i)$.

Preconditions: $p \neq \varepsilon$ and $qp^* \subseteq \bigcup_{j \in \mathbb{N}} \text{Pos}(s\mu^j)$

Input: s, q, p, μ

Output: (x, p_i, i) such that x infinitely often occurs in terms $s\mu^j$ along path qp^ω where i and $p_i < qp^\omega$ are minimal such that $s\mu^i|_{p_i} = x$

- (1) $i := 0, u := s, p_{\text{start}} := \varepsilon, p_{\text{end}} := q, S := \emptyset$
- (2) if $u = x$ and $(x, -, p_{\text{end}}, -) \in S$ then return $(x, p_{\text{start}}', i')$ where i' is the smallest value such that $(x, p_{\text{start}}', -, i') \in S$
- (3) if $u = x$ and $(x, -, p_{\text{end}}, -) \notin S$ then $u := u\mu, S := S \cup \{(x, p_{\text{start}}, p_{\text{end}}, i)\}, i := i + 1,$ goto (2)
- (4) (a) if $p_{\text{end}} = \varepsilon$ then $p_{\text{end}} := p$
 (b) let $p_{\text{end}} = jp'$ and $u = f(u_1, \dots, u_n); u := u_j, p_{\text{end}} := p', p_{\text{start}} := p_{\text{start}}j,$ goto (2)

Fig. 3. $\text{var}_\infty(s, q, p, \mu)$

Theorem 21. *The algorithm var_∞ terminates and is sound.*

Preconditions: $p \neq \varepsilon$ and $p^* \subseteq \bigcup_{i \in \mathbb{N}} \text{Pos}(t\mu^i)$

Input: x, t, p, μ

Output: (j, q) if j is minimal such that $t\mu^j|_q = x$ where $q \in \text{Pos}(t\mu^j)$ is prefix of p^ω or \perp , if there is no such j

- (1) $j := 0, u := t, p_{\text{start}} := \varepsilon, p_{\text{end}} := \varepsilon, S := \emptyset$
- (2) if $u = x$ then return (j, p_{start})
- (3) if $u = y \neq x$ and $(y, p_{\text{end}}) \in S$ then return \perp
- (4) if $u = y \neq x$ and $(y, p_{\text{end}}) \notin S$ then $j := j + 1, S := S \cup \{(y, p_{\text{end}})\}, u := u\mu,$ goto (2)
- (5) (a) if $p_{\text{end}} = \varepsilon$ then $p_{\text{end}} := p$
 (b) let $p_{\text{end}} = ip'$ and $u = f(u_1, \dots, u_n); u := u_i, p_{\text{end}} := p', p_{\text{start}} := p_{\text{start}}i,$ goto (2)

Fig. 4. $\text{idx}(x, t, p, \mu)$

Finally, for Step (iv) a small adaptation of var_∞ yields the last required algorithm idx in Fig. 4 to check whether x occurs along p^ω in some term $t\mu^j$.

Theorem 22. *The algorithm idx terminates and is sound.*

Note that both algorithms var_∞ and idx become unsound if one only considers equal variables in the set S , but not equal positions p_{end} .

6 Empirical Results

In the following we first give some details about the actual implementation of our method and after that empirical results. To find loops, we use unfoldings as

defined in [12], Section 3 (without any refinements mentioned in later sections). For efficiency reasons we restrict to non-variable positions. Further we do not use a combination of forward and backward unfoldings by default. Our basic method uses the following heuristic to decide the direction of the unfoldings: For systems that are duplicating but whose inverse is non-duplicating we unfold backwards. For all other systems we unfold forwards.

In contrast to finding loops for full termination, for a specific strategy \mathcal{S} , we cannot always stop when a loop was found (since it could turn out to be no longer relevant when switching from full rewriting to \mathcal{S} -rewriting). Hence we compute a lazy list of potential loops that is checked one at a time corresponding to \mathcal{S} . If the checked loop is no \mathcal{S} -loop, the next loop is requested (and lazyness makes sure that the necessary computations are only done after the previous element dropped out).

For context-sensitive rewriting as well as outermost rewriting we reduce the search space by filtering the set of unfoldings after each iteration. In both cases we remove derivations containing rewrite steps that disobey the strategy.

As already mentioned in the introduction there is the transformational approach for both, context-sensitive rewriting [3] and outermost rewriting [13,14]. In both cases a problem for finding loops, is that the length of an existing loop may increase dramatically, or even worse, a loop is transformed into a non-looping infinite derivation.

To evaluate our implementation in $\mathsf{T}\mathsf{T}_2$ we used all 291 (214) outermost examples as well as all 109 (15) context-sensitive examples of version 5.0.2 of the Termination Problems Data Base.⁴ Here, in brackets the number of those TRSs is given, where outermost resp. context-sensitive termination has not already been proven. The results on these possibly non-terminating TRSs are as follows:

	outermost TRSs			CSRs
	AProVE	Trafo	$\mathsf{T}\mathsf{T}_2$	$\mathsf{T}\mathsf{T}_2$
NO score	37	30	191	4
avg. time (msec)	6689	6772	340	38

For outermost rewriting we compare to the sum of non-termination proofs (NO score) achieved by AProVE and Trafo⁵ at the January 2009 termination competition.⁴⁶ The success of our technique is clearly visible: $\mathsf{T}\mathsf{T}_2$ was able to disprove termination of nearly 90 % of all possible non-terminating TRSs, including all examples that could be handled by AProVE and Trafo (which use the transformational approaches of [14] and [13] respectively).

For context-sensitive rewriting we just give the NOs of $\mathsf{T}\mathsf{T}_2$ since we are not aware of any other tool that has disproven context-sensitive termination of a single TRS. Here, our implementation could at least solve one quarter of the potentially non-terminating TRSs.

⁴ <http://termcomp.uibk.ac.at>

⁵ <http://www.win.tue.nl/~mraffels/trafo.html>

⁶ The numbers for $\mathsf{T}\mathsf{T}_2$ differ from those of the competition since the competition version did not feature the reduction of the search space which is described above.

7 Conclusion and Future Work

To prove non-termination of rewriting under strategy \mathcal{S} , we first extended the notion of a loop to an \mathcal{S} -loop. An \mathcal{S} -loop is an \mathcal{S} -reduction with a strong regularity which admits the same infinite reduction as an ordinary loop does for full rewriting. Afterwards, we developed two novel procedures to decide whether a given loop is a context-sensitive loop or an outermost loop. It is easy to see that the conjunction of both procedures decides context-sensitive outermost loops.

Since [6] only describes a way to prove termination of Haskell programs, it might be an interesting future work to combine our technique for outermost loops with [6] to also disprove termination of Haskell programs.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
2. J. Endrullis. *Jambox*. Available at <http://joerg.endrullis.de>.
3. J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4):379–427, 2004.
4. J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. FroCoS '05*, LNAI 3717, pages 216–231, 2005.
5. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the DP framework. In *Proc. IJCAR '06*, LNAI 4130, pages 281–286, 2006.
6. 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, pages 297–312, 2006.
7. J. Guttag, D. Kapur, and D. Musser. On proving uniform termination and restricted termination of rewriting systems. *SIAM J. Computation*, 12:189–214, 1983.
8. M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In *Proc. RTA '09*, LNCS, 2009.
9. W. Kurth. *Termination und Konfluenz von Semi-Thue-Systemen mit nur einer Regel*. PhD thesis, Technische Universität Clausthal, Germany, 1990.
10. D. Lankford and D. Musser. A finite termination criterion. Unpublished Draft. USC Information Sciences Institute, 1978.
11. S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1:1–61, 1998.
12. Étienne Payet. Loop detection in term rewriting using the eliminating unfoldings. *Theoretical Computer Science*, 403(2-3):307–327, 2008.
13. M. Raffelsieper and H. Zantema. A transformational approach to prove outermost termination automatically. In *Proc. WRS '08*, ENTCS 237, pages 3–21, 2009.
14. R. Thiemann. From outermost termination to innermost termination. In *Proc. SOFSEM '09*, LNCS 5404, pages 533–545, 2009.
15. R. Thiemann, J. Giesl, and P. Schneider-Kamp. Deciding innermost loops. In *Proc. RTA '08*, LNCS 5117, pages 366–380, 2008.
16. J. Waldmann. *Matchbox*: A tool for match-bounded string rewriting. In *Proc. RTA '04*, LNCS 3091, pages 85–94, 2004.
17. H. Zantema. Termination of string rewriting proved automatically. *Journal of Automated Reasoning*, 34:105–139, 2005.