Logical Methods in Computer Science
Vol. 13(1:11)2017, pp. 1–38
www.lmcs-online.org

Submitted Apr. 28, 2016
Published Mar. 17, 2017

TERMINATION OF CYCLE REWRITING BY TRANSFORMATION AND MATRIX INTERPRETATION

DAVID SABEL [a] AND HANS ZANTEMA [b]

[a] Goethe-University Frankfurt, Department of Computer Science and Mathematics, Computer Science Institute, 60629 Frankfurt am Main, Germany
e-mail address: sabel@ki.informatik.uni-frankfurt.de

[b] TU Eindhoven, Department of Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Radboud University Nijmegen, Institute for Computing and Information Sciences, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
e-mail address: h.zantema@tue.nl

Abstract. We present techniques to prove termination of cycle rewriting, that is, string rewriting on cycles, which are strings in which the start and end are connected. Our main technique is to transform cycle rewriting into string rewriting and then apply state of the art techniques to prove termination of the string rewrite system. We present three such transformations, and prove for all of them that they are sound and complete. In this way not only termination of string rewriting of the transformed system implies termination of the original cycle rewrite system, a similar conclusion can be drawn for non-termination.

Apart from this transformational approach, we present a uniform framework of matrix interpretations, covering most of the earlier approaches to automatically proving termination of cycle rewriting.

All our techniques serve both for proving termination and relative termination.

We present several experiments showing the power of our techniques.

1. Introduction

Cycles can be seen as strings of which the left end is connected to the right end, by which the string has no left end or right end any more. In Fig. 1 a pictorial representation of two such cycles is shown.

String rewriting can not only be applied on strings, but also on cycles. Applying string rewriting on cycles, i.e. replacing a substring of a cycle by another substring, is briefly called cycle rewriting. For instance, applying the string rewrite rule $aaa \to ababa$ to the cycle in Fig. 1 (a) results in the cycle shown Fig. 1 (b).

Rewriting behavior is strongly influenced by allowing cycles, for instance, in string rewriting the single rule $ab \to ba$ is terminating, but in cycle rewriting it is not, since the string ab represents the same cycle as ba.

In many areas cycle rewriting is more natural than string rewriting. For instance, the problem of 5 dining philosophers can be expressed as a cycle $FTFTFTFTFT$ where F

Key words and phrases: rewriting systems, string rewriting, termination, relative termination.
[a] The first author is supported by the Deutsche Forschungsgemeinschaft (DFG) under grant SA 2908/3-1.

DOI:10.23638/LMCS-13(1:11)2017

Figure 1: Illustration of Cycles and Cycle Rewriting

denotes a fork, and T denotes a thinking philosopher. Writing L for a philosopher who has picked up her left fork, but not her right fork, and E for an eating philosopher, a classical (deadlocking) modeling of the dining philosophers problem (for arbitrary many philosophers) can be expressed by the cycle rewrite system consisting of the rules $TF \to L$, $FL \to E$, $E \to FTF$. As a cycle rewrite system this is clearly not terminating.

Also from the viewpoint of graph transformation, cycle rewriting is very natural. For instance, in [3] it was shown that if all rules of a graph transformation system are string rewrite rules, termination of the transformation system coincides with termination of the cycle rewrite system, and not with termination of the string rewrite system. Developing techniques to prove cycle termination also helps to understand and develop new techniques for proving termination of graph transformation systems (see e.g. [2]).

So both string rewriting and cycle rewriting provide natural semantics for string rewrite systems, also called semi-Thue systems. Historically, string rewriting got a lot of attention as being a particular case of term rewriting, while cycle rewriting hardly got any attention until recently. In 2015 automated proving of cycle termination became a new category in the Termination Competition [11, 21]. In 2015 two tools participated, and in 2016 three tools participated in the category.

In [28] a first investigation of termination of cycle rewriting was made. Some techniques were presented to prove cycle termination, implemented in a tool `torpacyc`. Further a transformation ϕ was given such that for every string rewriting system (SRS) R, string termination of R holds if and only if cycle termination of $\phi(R)$ holds. As a consequence, cycle termination is undecidable.

However, for making use of the strong power of current tools for proving termination of string rewriting in order to prove cycle termination, a transformation the other way around is needed: transformations ψ such that for every SRS R, cycle termination of R holds if and only if string termination of $\psi(R)$ holds. The 'if' direction in this 'if and only if' is called 'sound', the 'only if' is called complete. This implies a way to prove cycle termination of an SRS R: apply a tool for proving termination of string rewriting to $\psi(R)$ for a sound transformation ψ. Conversely, a way to prove cycle non-termination of R is to prove non-termination of $\psi(R)$ for a complete transformation ψ. The main topic of this paper is to investigate such transformations, and to exploit them to prove termination of cycle rewriting, or non-termination, or relative (non-)termination. Here relative termination deals with two rewrite systems: relative termination means that every infinite reduction of the union of them contains only finitely many steps of one of them. In detail we give fully worked-out proofs of soundness and completeness for three approaches to transform cycle termination into string termination (called split, rotate, and shift), and also for relative termination.

Using transformations to exploit the power of tools for termination of term rewriting to prove a modified property was used before in [13, 12]. However, there the typical observation was that the complete transformations were complicated, and for non-trivial examples, termination of $\psi(R)$ could not be proved by the tools, while for much simpler sound (but incomplete) transformations ψ, termination of $\psi(R)$ could often be proved by the tools. In our current setting this is different: one of our introduced transformations, the transformation split, for which we prove that it is sound and complete, we show that for several systems R for which all approaches from [28] fail, cycle termination of R can be concluded from an automatic termination proof of $\mathsf{split}(R)$ generated by AProVE [10, 1] or $\mathrm{T_TT_2}$ [17, 24].

It can be shown that if strings of size n exist admitting cycle reductions in which for every rule the number of applications of that rule is more than linear in n, then all techniques from [28] fail to prove cycle termination. Nevertheless, in quite simple examples this may occur while cycle termination holds. As an example consider the following.

A number of people are in a circle, and each of them carries a number, represented in binary notation with a bounded number of bits. Each of them may increase his/her number by one, as long as it fits in the bounded number of bits. Apart from that, every person may increase the number of bits of the number of its right neighbor by two. In order to avoid trivial non-termination, the latter is only allowed if the leading bit of the number is 0, and the new leading bit is put to 1, and the other to 0, by which effectively one extra bit is added. We will prove that this process will always terminate by giving an SRS in which all of the above steps can be described by a number of cycle rewrite steps, and prove cycle termination. In order to do so we write P for person, and 0 and 1 for the bits of the binary number. For carry handling we introduce an extra symbol c of which the meaning is a 1 with a carry. Assume for every person its number is stored left from it. So if the number ends in 0, by adding one this last bit 0 is replaced by 1, expressed by the rule $0P \to 1P$. In case the number ends in 1, a carry should be created, since c represents a 1 with a carry this is expressed by the rule $1P \to cP$. Next the carry should be processed. In case it is preceded by 0, this 0 should be replaced by 1, while the c is replaced by 0; this is expressed by the rule $0c \to 10$. In case it is preceded by 1, a new carry should be created while again the old carry is replaced by 0; this is expressed by the rule $1c \to c0$. In this way adding one to any number in binary notation can be expressed by a number of rewrite steps, as long as no overflow occurs. Finally, we have to add a rule to extend the bit size of the number of the right neighbor: the leading bit should be 0, while it is replaced by 100: adding two extra bits of which the leading one is 1 and the other is 0. This is expressed by the rule $P0 \to P100$. Summarizing: we have to prove cycle termination of the SRS consisting of the five rules

$$0P \to 1P,\ 1P \to cP,\ 0c \to 10,\ 1c \to c0,\ P0 \to P100.$$

This is fundamentally impossible by the techniques presented in [28]: by one of the techniques the last rule can be removed, but starting in 0^nP a reduction can be made in which all of the remaining four rules are applied an exponential number of times, by which the techniques from [28] fail.

In this paper we give two ways to automatically prove that cycle termination holds for the above example R: $\mathrm{T_TT_2}$ succeeds in proving termination of $\mathsf{split}(R)$, and the other is a variant of matrix interpretations for which we show that it proves cycle termination. The latter is another main topic of this paper: we give a self-contained uniform presentation of trace-decreasing matrix interpretations for cycle termination that covers the tropical and arctic variant from [28] and the natural variant from [18], now also serving for relative

termination. In this way we cover all current known techniques for proving cycle termination except for match bounds ([28]).

The paper is organized as follows. Section 2 recalls the basics of cycle rewriting and introduces relative cycle termination. The main section Section 3 first adapts type introduction [26] for string rewriting and relative termination (Theorem 3.3), and then the three transformations split, rotate, and shift are presented and soundness and completeness of all of them is proved (Theorems 3.13, 3.30, and 3.46). Also adapted transformations for relative cycle transformation are presented and their soundness and completeness is shown (Theorems 3.18, 3.32, and 3.48). In Section 4 trace-decreasing matrix interpretations are presented, and we show how they can be used to prove cycle termination (Theorem 4.1) and relative cycle termination (Theorem 4.2) for the three instances of natural, tropical and arctic matrix interpretations. In Section 5 experiments on implementations of our techniques are reported. We conclude in Section 6.

Compared to our paper [18], this paper contains full proofs, extends all the termination techniques to relative termination, and presents a framework for matrix interpretations which covers and extends the previous approaches presented in [28, 18].

2. Preliminaries

In this section we briefly recall the required notions for string and cycle rewriting.

A *signature* Σ is a finite alphabet of symbols. With Σ^* we denote the set of strings over Σ. With ε we denote the empty string and for $u, v \in \Sigma^*$, we write uv for the concatenation of the strings u and v. With $|u|$ we denote the length of string $u \in \Sigma^*$ and for $a \in \Sigma$ and $n \in \mathbf{N}$, a^n denotes n replications of symbol a, i.e. $a^0 = \varepsilon$ and $a^i = aa^{i-1}$ for $i > 0$.

Given a binary relation $\to$, we write $\to^i$ for i steps, $\to^{\leq i}$ for at most i steps, $\to^{<i}$ for at most $i-1$ steps, $\to^*$ for the reflexive-transitive closure of $\to$, and $\to^+$ for the transitive closure of $\to$. For binary relations $\to_1$ and $\to_2$, we write $\to_1 \,.\, \to_2$ for the composition of $\to_1$ and $\to_2$, i.e. $a \to_1 \,.\, \to_2 c$ iff there exists a b s.t. $a \to_1 b$ and $b \to_2 c$.

2.1. **String Rewriting.** A *string rewrite system* (SRS) is a finite set R of rules $\ell \to r$ where $\ell, r \in \Sigma^*$ and $\ell \neq \varepsilon$. The *rewrite relation* $\to_R \subseteq (\Sigma^* \times \Sigma^*)$ is defined as follows: if $w = u\ell v \in \Sigma^*$ and $(\ell \to r) \in R$, then $w \to_R urv$. The *prefix-rewrite relation* $\hookrightarrow_R$ is defined as: if $w = \ell u \in \Sigma^*$ and $(\ell \to r) \in R$, then $w \hookrightarrow_R ru$. The *suffix-rewrite relation* $\hookleftarrow_R$ is defined as: if $w = u\ell \in \Sigma^*$ and $(\ell \to r) \in R$, then $w \hookleftarrow_R ur$.

A (finite or infinite) sequence of rewrite steps $w_1 \to_R w_2 \to_R \cdots$ is called a *rewrite sequence* (sometimes also a *reduction* or a *derivation*). For an SRS R, the rewrite relation $\to_R$ is called *non-terminating* if there exists a string $w \in \Sigma^*$ and an infinite rewrite sequence $w \to_R w_1 \to_R \cdots$. Otherwise, $\to_R$ is *terminating*. If $\to_R$ is terminating (non-terminating, resp.) we also say R is *string terminating* (string non-terminating, resp.).

2.2. **Cycle Rewriting.** We recall the notion of cycle rewriting from [28]. A string can be viewed as a cycle, i.e. the last symbol of the string is connected to the first symbol. To represent cycles by strings, we define the equivalence relation $\sim$ as follows:

$$u \sim v \text{ iff } u = w_1 w_2 \text{ and } v = w_2 w_1 \text{ for some strings } w_1, w_2 \in \Sigma^*$$

With $[u]$ we denote the equivalence class of string u w.r.t. $\sim$.

The *cycle rewrite relation* $\circ\!\!\to_R \subseteq (\Sigma/\!\sim \times \Sigma/\!\sim)$ of an SRS R is defined as

$$[u] \circ\!\!\to_R [v] \text{ iff } \exists w \in \Sigma^* : \ell w \in [u], (\ell \to r) \in R, \text{ and } rw \in [v]$$

The cycle rewrite relation $\circ\!\!\to_R$ is called *non-terminating* iff there exists a string $w \in \Sigma^*$ and an infinite sequence $[w] \circ\!\!\to_R [w_1] \circ\!\!\to_R [w_2] \circ\!\!\to_R \cdots$. Otherwise, $\circ\!\!\to_R$ is called *terminating*. If $\circ\!\!\to_R$ is terminating (non-terminating, resp.) we also say that R is *cycle terminating* (cycle non-terminating, resp.).

We recall some known facts about cycle rewriting.

Proposition 2.1 (see [28])**.** *Let Σ be a signature, R be an SRS, and $u, v \in \Sigma^*$.*

(1) *If $u \to_R v$ then $[u] \circ\!\!\to_R [v]$.*
(2) *If $\circ\!\!\to_R$ is terminating, then $\to_R$ is terminating.*
(3) *Termination of $\to_R$ does not necessarily imply termination of $\circ\!\!\to_R$.*
(4) *Termination of $\circ\!\!\to_R$ is undecidable.*
(5) *For every SRS R there exists a transformed SRS $\phi(R)$ s.t. the following three properties are equivalent:*
   - *$\to_R$ is terminating.*
   - *$\to_{\phi(R)}$ is terminating.*
   - *$\circ\!\!\to_{\phi(R)}$ is terminating.*

For an SRS R, the last property implies that termination of $\to_R$ can be proved by proving termination of the translated cycle rewrite relation $\circ\!\!\to_{\phi(R)}$. In [28] it was used to show that termination of cycle rewriting is undecidable and for further results on derivational complexity for cycle rewriting.

2.3. **Relative Termination.** We will also consider relative termination of cycle and string rewrite systems. Hence, in this section we recall the definition of relative termination of string rewrite systems (see e.g. [9]) and introduce relative cycle termination.

Definition 2.2. Let $\to_1 \subseteq \to_2 \subseteq (O \times O)$ be binary relations on a set O. We say $\to_1$ is *terminating relative* to $\to_2$ iff every infinite sequence $o_1 \to_2 o_2 \to_2 \cdots$ contains only finitely many $\to_1$-steps.

Let $S \subseteq R$ be string rewrite systems over an alphabet Σ. If the string rewrite relation $\to_S$ is terminating relative to the string rewrite relation $\to_R$ then we say S is *string terminating relative to R*. If the cycle rewrite relation $\circ\!\!\to_S$ is terminating relative to the cycle rewrite relation $\circ\!\!\to_R$ then we say S is *cycle terminating relative to R*.

For $S \subseteq R$, we sometimes call $R \setminus S$ the weak rules and S the strict rules. We often write rules in $R \setminus S$ as $\ell \to\!\!= r$ and rules in S as $\ell \to r$.

Note that for all SRSs R and S the identities $\to_{R \cup S} = \to_R \cup \to_S$ and $\circ\!\!\to_{R \cup S} = \circ\!\!\to_R \cup \circ\!\!\to_S$ hold, which we will sometimes use.

Since every string rewrite step, is also a cycle rewrite step (on the equivalence class w.r.t. $\sim$), any infinite string rewrite sequence giving evidence for string non-termination, also gives evidence for cycle non-termination. Thus we have:

Corollary 2.3. *If S is cycle terminating relative to R, then S is also string terminating relative to R.*

However, as for usual termination, relative cycle termination is different from relative string termination:

Example 2.4. Let $S = \{ab \to ca\}$ and $R = S \cup \{c \to\!\!= b\}$. Then S is string terminating relative to R (which is easy to prove), while S is not cycle terminating relative to R, since the infinite (looping) cycle rewrite sequence $[ab] \circ\!\!\to_S [ca] \circ\!\!\to_R [ab] \dots$ contains infinitely many $\circ\!\!\to_S$ steps.

The following proposition provides several characterizations for relative termination. We formulate it in a general form (for all binary relations). Even though its proof is quite standard, we include it for the sake of completeness.

Proposition 2.5. *Let $\to_1 \subseteq \to_2$ be binary relations on a set O. The following five propositions are equivalent:*

(1) *The relation $\to_2^* . \to_1 . \to_2^*$ is terminating.*
(2) *The relation $\to_1 . \to_2^*$ is terminating.*
(3) *The relation $\to_2^* . \to_1$ is terminating.*
(4) *$\to_1$ is terminating relative to $\to_2$.*

Proof. We show the claim by a chain of implications:

(1) $\Longrightarrow$ (2): Clearly $\to_1 . \to_2^* \subseteq \to_2^* . \to_1 . \to_2^*$, and thus non-termination of $\to_1 . \to_2^*$ implies non-termination of $\to_2^* . \to_1 . \to_2^*$.

(2) $\Longrightarrow$ (3): Assume $\to_2^* . \to_1$ is non-terminating. Then there exists an infinite sequence s.t. $o_{i,1} \to_2^* o_{i,2} \to_1 o_{i+1,1}$ for $i = 1, 2, \dots$. This implies $o_{i,2} \to_1 o_{i+1,1} \to_2^* o_{i+1,2}$ for all $i = 1, 2, \dots$ and thus $\to_1 . \to_2^*$ is non-terminating.

(3) $\Longrightarrow$ (4)**::** We show $\neg(4) \implies \neg(3)$ Thus, we assume that there exists a sequence $o_1 \to_2 o_2 \to_2 \cdots$ s.t. for every n there exists a number $m_n \geq n$ s.t. $o_{m_n} \to_1 o_{m_n+1}$. Assume that each number m_n is minimal w.r.t. n. Then the given sequence can be written as $o'_j \to_2^{k_j} o'_{j+k_j} \to_1 o'_{j+k_j+1} = o'_{j+1}$ s.t. $o'_0 = o_1$ and $k_j \geq 0$ for $j = 0, 1, \dots$. Since $o'_j \to_2^* . \to_1 o'_{j+1}$, this shows that $\to_2^* . \to_1$ is non-terminating.

(4) $\Longrightarrow$ (1):: If $\to_2^* . \to_1 . \to_2^*$ is non-terminating, then the infinite rewrite sequence consisting of infinitely many $\to_2^* . \to_1 . \to_2^*$-steps contains infinitely many $\to_1$-steps. □

For SRSs $S \subseteq R$, the previous proposition can be instantiated with $\to_1 := \to_S$ and $\to_2 := \to_R$ to derive characterizations of relative string termination, and with $\to_1 := \circ\!\!\to_S$ and $\to_2 := \circ\!\!\to_R$ to derive characterizations of relative cycle termination.

3. Transforming Cycle Termination into String Termination

The criteria given in Proposition 2.1 and the involved transformation ϕ, which transforms string rewriting into cycle rewriting, provide a method to prove string termination by proving cycle termination. However, it does not provide a method to prove termination of the cycle rewrite relation $\circ\!\!\to_R$ by proving termination of the string rewrite relations $\to_R$ or $\to_{\phi(R)}$. Hence, in this section we develop transformations ψ s.t. termination of $\to_{\psi(R)}$ implies termination of $\circ\!\!\to_R$. We call such a transformation ψ *sound*. However, there are "useless" sound transformations, for instance, transformations where $\psi(R)$ is always non-terminating. So at least one wants to find sound transformations which permit to prove termination of non-trivial cycle rewrite relations. However, a better transformation should fulfill the stronger property that $\to_{\psi(R)}$ is terminating if and only if $\circ\!\!\to_R$ is terminating. If termination of $\circ\!\!\to_R$ implies termination of $\to_{\psi(R)}$, then we say ψ is *complete*. For instance, for a complete transformation, non-termination proofs of $\to_{\psi(R)}$ also imply non-termination of $\circ\!\!\to_R$. Hence, our goal is to find sound and complete transformations ψ.

Besides such transformations, we will consider transformations $\psi_{rel}(S,R) = (S',R')$ which are sound and complete for relative termination, i.e. transformations which transform (S,R) with $S \subseteq R$, such that cycle termination of S relative to R holds, if and only if S' is string terminating relative to R'.

We will introduce and discuss three transformations split, rotate, and shift where the most important one is the transformation split, since it has the following properties: The transformation is sound and complete, and as our experimental results show, it behaves well in practice when proving termination of cycle rewriting. The other two transformations rotate and shift are also sound and complete, but rather complex and – as our experimental results show – they do not behave as well as the transformation split in practice. We include all three transformations in this paper to document some different approaches to transform cycle rewriting into string rewriting. We also consider variations of the three transformations for relative termination. We show that all three variations are sound and complete transformations for relative termination.

Since our completeness proofs, use type introduction [26], we recall this technique in Section 3.1 focused on typed string rewriting only, and prove a (novel) theorem that type introduction is correct for relative string termination. In the remaining sections 3.2, 3.3, and 3.4 we successively introduce and treat the three transformations.

3.1. **Type Introduction.** The technique of type introduction was presented in [26], for termination of term rewriting. Here we are only interested in string rewriting (being the special case of term rewriting having only unary symbols), but for our purpose need to extend this result to relative termination.

An signature Σ is *typed* if there is a set $\mathbb{T}$ of types (also called sorts), and every $a \in \Sigma$ has a *source type* $\tau_1 \in \mathbb{T}$ and a *target type* $\tau_2 \in \mathbb{T}$, notation $a : \tau_1 \to \tau_2$, $\tau_1 = \mathsf{source}(a)$, $\tau_2 = \mathsf{target}(a)$.

For a non-empty string $w \in \Sigma^+$ its target $\mathsf{target}(w)$ is defined to be the target of its first element; its source $\mathsf{source}(w)$ is defined to be the source of its last element. A non-empty string $w \in \Sigma^+$ is called *well-typed* if either it is in Σ, or it is of the shape au for $a \in \Sigma$ and $u \in \Sigma^+$ is well-typed, and $\mathsf{source}(a) = \mathsf{target}(u)$.

An SRS is called *well-typed* if for every rule $\ell \to r$ we either have

- $r = \varepsilon$ and $\mathsf{source}(\ell) = \mathsf{target}(\ell)$, or
- $r \neq \varepsilon$ and $\mathsf{source}(\ell) = \mathsf{source}(r)$ and $\mathsf{target}(\ell) = \mathsf{target}(r)$.

The following lemma is straightforward.

Lemma 3.1. *If R is a well-typed SRS over Σ and $w \to_R w'$ for $w \in \Sigma^+$ being well-typed, then w' is well-typed too.*

So in an infinite reduction with respect to a well-typed SRS, all strings are well-typed if and only if the initial string is well-typed.

For a well-typed SRS R, we say that R is *string terminating in the typed setting* if there does not exist an infinite $\to_R$-reduction consisting of well-typed strings.

For well-typed SRSs $S \subseteq R$ we say that S is *string terminating relative* to R *in the typed setting* if every infinite $\to_R$-reduction consisting of well-typed strings contains only finitely many $\to_S$-steps.

The main theorem, to be exploited several times in this paper, states that this notion of relative termination in the typed setting is equivalent to the notion of relative termination in the general setting without typing requirements.

In order to prove this theorem we need a notion of decomposition of (possibly untyped) strings and a lemma stating some key properties of this decomposition. We denote a string consisting of n strings $u_1, \ldots, u_n$ by $[u_1, \ldots, u_n]$. The decomposition $\mathsf{Dec}(u)$ of a string $u \in \Sigma^+$ is such a string of strings and is defined as follows:

- $\mathsf{Dec}(a) = [a]$,
- if $u \in \Sigma^+$ and $\mathsf{Dec}(u) = [u_1, \ldots, u_n]$, then
  - $\mathsf{Dec}(au) = [au_1, \ldots, u_n]$ if $\mathsf{source}(a) = \mathsf{target}(u_1)$, and
  - $\mathsf{Dec}(au) = [a, u_1, \ldots, u_n]$ if $\mathsf{source}(a) \neq \mathsf{target}(u_1)$.

By construction for any $u \in \Sigma^+$ with $\mathsf{Dec}(u) = [u_1, \ldots, u_n]$ we have the following properties:

- u_i is well-typed for $i = 1, \ldots, n$;
- $u = u_1 \ldots u_n$;
- if v is a well-typed substring of u, then it also a substring of u_i for some $i = 1, \ldots, n$;

As we consider well-typed SRSs only, with non-empty left hand sides, every rewrite step applied on such u applies to one of the corresponding u_i. In case of a collapsing rule, that is, a rule with empty right hand side, it may be the case that a type clash is removed, decreasing the length $|\mathsf{Dec}(u)|$ of $\mathsf{Dec}(u)$. For instance, for $a : \tau_1 \to \tau_1$, $b : \tau_2 \to \tau_2$, we have $baabab \to_R baaab$ for $R = \{b \to \varepsilon\}$, while $\mathsf{Dec}(baabab) = [b, aa, b, a, b]$ has length 5 and $\mathsf{Dec}(baaab) = [b, aaa, b]$ has length 3. In all other cases the rewriting of u takes place in one of the elements of $\mathsf{Dec}(u)$, while all other elements remain unchanged. These observations are summarized in the following lemma.

Lemma 3.2. *Let R be a well-typed SRS with non-empty left hand sides and $u \to_R v$. Then*

- *$|\mathsf{Dec}(u)| \geq |\mathsf{Dec}(v)|$, and*
- *if $|\mathsf{Dec}(u)| = |\mathsf{Dec}(v)|$, then there exists $i \in \{1, \ldots, n\}$ such that $u_i \to_R v_i$, and $u_j = v_j$ for $j \neq i$, where $\mathsf{Dec}(u) = [u_1, \ldots, u_n]$ and $\mathsf{Dec}(v) = [v_1, \ldots, v_n]$.*

Now we are prepared for the main theorem. The way it is used is as follows: for proving (relative) termination, try to find a typing such that the SRS is well-typed. Then according to the theorem the infinite reduction for which a contradiction has to be derived, may be assumed to be well-typed.

Theorem 3.3. *Let $S \subseteq R$ be well-typed SRSs with non-empty left hand sides. Then S is string terminating relative to R if and only if S is string terminating relative to R in the typed setting.*

Proof. The 'only if'-part is trivial. For the 'if'-part assume we have an infinite R-reduction $u_1 \to_R u_2 \to_R u_3 \to_R \cdots$, not well-typed; we have to prove it contains only finitely many S-steps. According to the first claim of Lemma 3.2 there exist n, N such $|\mathsf{Dec}(u_i)| = |\mathsf{Dec}(u_{i+1})| = n$ for all $i \geq N$. Write $\mathsf{Dec}(u_i) = [u_{i1}, \ldots, u_{in}]$ for $i \geq N$. According to the second part of Lemma 3.2 for every $j = 1, \ldots, n$ we have either $u_{ij} = u_{i+1,j}$ or $u_{ij} \to_R u_{i+1,j}$ for $i \geq N$. If $u_{ij} \to_R u_{i+1,j}$ occurs infinitely often this yields a well-type infinite R-reduction, containing only finitely many S-steps since S is terminating relative to R in the typed setting. If $u_{ij} \to_R u_{i+1,j}$ occurs finitely often, it also contains only finitely many S-steps. As the number of S-steps in the finite part $u_1 \to_R^* u_N$ is finite too, we conclude that the total number of S-steps in the original reduction is finite. □

By instantiating the previous theorem with $S = R$ we obtain correctness of type introduction for string termination:

Corollary 3.4. *Let R be a well-typed SRS. Then R is string terminating if and only if R is string terminating in the typed setting.*

3.2. **The Transformation Split.** The idea of the transformation split is to perform a single cycle rewrite step $[u] \circ\!\!\to_R [v]$ which uses rule $(\ell \to r) \in R$, by either applying a string rewrite step $u \to_R v$ or by splitting the rule $(\ell \to r)$ into two rules $(\ell_p \to r_p)$ and $(\ell_s \to r_s)$, where $\ell = \ell_p\ell_s$ and $r = r_pr_s$. Then a cycle rewrite step can be simulated by a prefix and a subsequent suffix rewrite step: first apply rule $\ell_s \to r_s$ to a prefix of u and then apply rule $\ell_p \to r_p$ to a suffix of the obtained string.

Example 3.5. Let $R = \{abc \to bbbb\}$ and $[bcdda] \circ\!\!\to_R [bbddbb]$. The rule $abc \to bbbb$ can be split into the rules $a \to bb$ and $bc \to bb$ s.t. $bcdda \hookrightarrow_{\{bc\to bb\}} bbdda \hookrightarrow_{\{a\to bb\}} bbddbb$.

We describe the idea of the transformation split more formally. It uses the following observation of cycle rewriting: if $[u] \circ\!\!\to_R [v]$, then $u \sim \ell w$, $(\ell \to r) \in R$, and $v \sim rw$. From $u \sim \ell w$ follows that $u = u_1u_2$ and $\ell w = u_2u_1$ for some u_1, u_2. We consider the cases for u_2:

(1) If $u_i = \varepsilon$ (for $i = 1$ or $i = 2$), then $u = \ell w$ and $u \hookrightarrow_R rw$ by a prefix string rewrite step.
(2) If ℓ is a prefix of u_2, i.e. $\ell u_2' = u_2$, then $w = u_2'u_1$, $u = u_1\ell u_2' \to_R u_1ru_2'$, and $u_1ru_2' \sim rw$.
(3) If u_2 is a proper prefix of ℓ, then there exist ℓ_p, ℓ_s with $\ell = \ell_p\ell_s$ s.t. $u_2 = \ell_p$ and ℓ_s is a non-empty prefix of u_1, i.e. $u_1 = \ell_s w$ and $u = u_1u_2 = \ell_s w\ell_p \hookrightarrow_{\{\ell_s\to r_s\}} r_s w\ell_p \hookrightarrow_{\{\ell_p\to r_p\}} r_s wr_p \sim rw$ if $r_pr_s = r$.

The three cases show that a cycle rewrite step $[u] \circ\!\!\to_{\{\ell\to r\}} [v]$ can either be performed by applying a string rewrite step $u \to_{\{\ell\to r\}} v'$ where $v' \sim v$ (cases 1 and 2) or in case 3 by splitting $\ell \to r$ into two rules $\ell_p \to r_p$ and $\ell_s \to r_s$ such that $u \hookrightarrow_{\{\ell_s\to r_s\}} u'$ replaces a prefix of u by r_s and $u' \hookrightarrow_{\{\ell_p\to r_p\}} v'$ replaces a suffix of u' by r_p s.t. $v' \sim v$.

For splitting a rule $(\ell \to r)$ into rules $\ell_p \to r_p$ and $\ell_s \to r_s$, we may choose any decomposition of r for r_p and r_s (s.t. $r = r_p\, r_s$). In the following, we will work with $r_p = r$ and $r_s = \varepsilon$.

The above cases for cycle rewriting show that a *sound* transformation of the cycle rewrite relation $\circ\!\!\to_R$ into a string rewrite relation is the SRS which consists of all rules of R and all pairs of rules $\ell_s \to \varepsilon$ and $\ell_p \to r$ for all $(\ell \to r) \in R$ and all ℓ_p, ℓ_s with $|\ell_p| > 0$, $|\ell_s| > 0$, and $\ell = \ell_p\ell_s$. However, this transformation does not ensure that the rules evolved by splitting are used as prefix and suffix rewrite steps only. Indeed, the transformation in this form is useless for nearly all cases, since whenever the right-hand side r of a rule $(\ell \to r) \in R$ contains a symbol $a \in \Sigma$ which is the first or the last symbol in ℓ, then the transformed SRS is non-terminating. For instance, for $R = \{aa \to aba\}$ the cycle rewrite relation $\circ\!\!\to_R$ is terminating, while the rule $a \to aba$ (which would be generated by splitting the left-hand side of the rule) leads to non-termination of the string rewrite relation. Note that this also holds if we choose any other decomposition of the right-hand side. Hence, in our transformation we introduce additional symbols to ensure:

- $\ell_s \to \varepsilon$ can only be applied to a prefix of the string.
- $\ell_p \to r$ can only be applied to a suffix of the string.
- If $\ell_s \to \varepsilon$ is applied to a prefix, then also $\ell_p \to r$ must be applied, in a synchronized manner (i.e. no other rule $\ell'_B \to \varepsilon$ or $\ell'_A \to r'$ can be applied in between).

In detail, we will prepend the fresh symbol B to the beginning of the string, and append the fresh symbol E to the end of the string. These symbols guarantee, that prefix rewrite steps $\ell u \hookrightarrow_{(\ell\to r)} ru$ can be expressed with usual string rewrite rules by replacing the left hand side ℓ with $\mathsf{B}\ell$ and analogous for suffix rewrite steps $u\ell \hookrightarrow_{(\ell\to r)} ur$ by replacing the left hand side ℓ with $\ell\mathsf{E}$.

Let $(\ell_i \to r_i)$ be the i^{th} rule of the SRS which is split into two rules $\ell_s \to \varepsilon$ and $\ell_p \to r_i$, where $\ell_p\ell_s = \ell_i$. After applying the rule $\ell_s \to \varepsilon$ to a prefix of the string, the symbol B will be replaced by the two fresh symbols W (for "wait") and $\mathsf{R}_{i,j}$ where i represents the i^{th} rule and j means that ℓ_i has been split after j symbols (i.e. $|\ell_p| = j$). The fresh symbol L is used to signal that the suffix has been rewritten by rule $\ell_p \to r$. Finally, we use a copy of the alphabet, to ensure completeness of the transformation: for an alphabet Σ, we denote by $\overline{\Sigma}$ a fresh copy of Σ, i.e. $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$. For a word $w \in \Sigma^*$ with $\overline{w} \in \overline{\Sigma}^*$, we denote the word w where every symbol a is replaced by $\overline{a}$. Analogously, for a word $w \in \overline{\Sigma}^*$ with $\underline{w} \in \Sigma$, we denote w where every symbol $\overline{a}$ is replaced by the symbol a.

Definition 3.6 (The transformation split)**.** Let $R = \{\ell_1 \to r_1, \ldots, \ell_n \to r_n\}$ be an SRS over alphabet Σ. Let $\overline{\Sigma}$ be a fresh copy of Σ and let $\mathsf{B}, \mathsf{E}, \mathsf{W}, \mathsf{R}_{i,j}, \mathsf{L}$ be fresh symbols (fresh for $\Sigma \cup \overline{\Sigma}$). The SRS $\mathsf{split}(R)$ over alphabet $\Sigma \cup \overline{\Sigma} \cup \{\mathsf{B}, \mathsf{E}, \mathsf{L}, \mathsf{W}\} \cup \bigcup_{i=1}^{n} \{\mathsf{R}_{i,j} \mid 1 \le j < |\ell_i|\}$ consists of the following string rewrite rules:

$$\ell_i \to r_i \quad \text{for all } (\ell_i \to r_i) \in R \quad \text{(splitA)}$$

$$\overline{a}\mathsf{L} \to \mathsf{L}a \quad \text{for all } a \in \Sigma \quad \text{(splitB)}$$

$$\mathsf{WL} \to \mathsf{B} \quad \text{(splitC)}$$

$$\mathsf{R}_{i,j}a \to \overline{a}\mathsf{R}_{i,j} \quad \text{for all } a \in \Sigma \quad \text{(splitD)}$$

for all $(\ell_i \to r_i) \in R$ and all $1 \le j < |\ell_i|$ and $\ell_p\ell_s = \ell_i$ with $|\ell_p| = j$:

$$\mathsf{B}\ell_s \to \mathsf{W}\mathsf{R}_{i,j} \quad \text{(splitE)}$$

$$\mathsf{R}_{i,j}\ell_p\mathsf{E} \to \mathsf{L}r_i\mathsf{E} \quad \text{(splitF)}$$

We describe the intended use of the rules and the extra symbols. The symbols B and E mark the start and the end of the string, i.e. for a cycle $[u]$ the SRS $\mathsf{split}(R)$ rewrites $\mathsf{B}u\mathsf{E}$.

Let $[u] \circ\!\!\to_R [w]$. The rule (splitA) covers the case that also $u \to_R w$ holds. Now assume that for $w' \sim w$ we have $u \hookrightarrow_{\{\ell_s\to\varepsilon\}} v \hookrightarrow_{\{\ell_p\to r\}} w'$ (where $(\ell_p\ell_s \to r) \in R$). Rule (splitE) performs the prefix rewrite step and replaces B by W to ensure that no other such a rule can be applied. Additionally, the symbol $\mathsf{R}_{i,j}$ corresponding to the rule and its splitting is added to ensure that only the right suffix rewrite step is applicable. Rule (splitD) moves the symbol $\mathsf{R}_{i,j}$ to right and rule (splitF) performs the suffix rewrite step. Rules (splitB) and (splitC) are used to finish the simulation of the cycle rewrite step by using the symbol L to restore the original alphabet and to finally replace WL by B.

Example 3.7. For $R_1 = \{aa \to aba\}$ the transformed string rewrite system $\mathsf{split}(R_1)$ is:

$$\begin{array}{llll} aa \to aba & \text{(splitA)} & \overline{a}\mathsf{L} \to \mathsf{L}a & \text{(splitB)} \\ \mathsf{WL} \to \mathsf{B} & \text{(splitC)} & \mathsf{B}a \to \mathsf{W}\mathsf{R}_{1,1} & \text{(splitE)} \\ \mathsf{R}_{1,1}a \to \overline{a}\mathsf{R}_{1,1} & \text{(splitD)} & \mathsf{R}_{1,1}b \to \overline{b}\mathsf{R}_{1,1} & \text{(splitD)} \end{array}$$

$$\overline{b}\mathsf{L} \to \mathsf{L}b \quad \text{(splitB)} \qquad \mathsf{R}_{1,1}a\mathsf{E} \to \mathsf{L}aba\mathsf{E} \quad \text{(splitF)}$$

For instance, the cycle rewrite step $[aba] \circ\!\!\to_{R_1} [baba]$ is simulated in the transformed system by the following sequence of string rewrite steps (where the redex is highlighted by a gray background):

$$\mathsf{B}aba\mathsf{E} \xrightarrow{splitE} \mathsf{W}\mathsf{R}_{1,1}ba\mathsf{E} \xrightarrow{splitD} \mathsf{W}\overline{b}\mathsf{R}_{1,1}a\mathsf{E} \xrightarrow{splitF} \mathsf{W}\overline{b}\mathsf{L}aba\mathsf{E} \xrightarrow{splitB} \mathsf{WL}baba\mathsf{E} \xrightarrow{splitC} \mathsf{B}baba\mathsf{E}.$$

As a further example, for the system $R_2 = \{abc \to cbacba,\ aa \to a\}$, the transformed string rewrite system $\mathsf{split}(R_2)$ is:

abc	$\to cbacba$	(splitA)	aa	$\to a$	(splitA)	$\mathsf{W}\mathsf{L}$	$\to \mathsf{B}$	(splitC)
$\overline{a}\mathsf{L}$	$\to \mathsf{L}a$	(splitB)	$\overline{b}\mathsf{L}$	$\to \mathsf{L}b$	(splitB)	$\overline{c}\mathsf{L}$	$\to \mathsf{L}c$	(splitB)
$\mathsf{B}bc$	$\to \mathsf{W}\mathsf{R}_{1,1}$	(splitE)	$\mathsf{B}c$	$\to \mathsf{W}\mathsf{R}_{1,2}$	(splitE)	$\mathsf{B}a$	$\to \mathsf{W}\mathsf{R}_{2,1}$	(splitE)
$\mathsf{R}_{1,1}a\mathsf{E}$	$\to \mathsf{L}cbacba\mathsf{E}$	(splitF)	$\mathsf{R}_{1,2}ab\mathsf{E}$	$\to \mathsf{L}cbacba\mathsf{E}$	(splitF)	$\mathsf{R}_{2,1}a\mathsf{E}$	$\to \mathsf{L}a\mathsf{E}$	(splitF)
$\mathsf{R}_{1,1}a$	$\to \overline{a}\mathsf{R}_{1,1}$	(splitD)	$\mathsf{R}_{1,2}a$	$\to \overline{a}\mathsf{R}_{1,2}$	(splitD)	$\mathsf{R}_{2,1}a$	$\to \overline{a}\mathsf{R}_{2,1}$	(splitD)
$\mathsf{R}_{1,1}b$	$\to \overline{b}\mathsf{R}_{1,1}$	(splitD)	$\mathsf{R}_{1,2}b$	$\to \overline{b}\mathsf{R}_{1,2}$	(splitD)	$\mathsf{R}_{2,1}b$	$\to \overline{b}\mathsf{R}_{2,1}$	(splitD)
$\mathsf{R}_{1,1}c$	$\to \overline{c}\mathsf{R}_{1,1}$	(splitD)	$\mathsf{R}_{1,2}c$	$\to \overline{c}\mathsf{R}_{1,2}$	(splitD)	$\mathsf{R}_{2,1}c$	$\to \overline{c}\mathsf{R}_{2,1}$	(splitD)

Termination of $\mathsf{split}(R_1)$ and $\mathsf{split}(R_2)$ can be proved by AProVE and T$_{\mathrm{T}}$T$_2$.

Proposition 3.8 (Soundness of split)**.** *If $\to_{\mathsf{split}(R)}$ is terminating then $\circ\!\!\to_R$ is terminating.*

Proof. By construction of $\mathsf{split}(R)$, it holds that if $[u] \circ\!\!\to_R [v]$, then $\mathsf{B}u'\mathsf{E} \to^+_{\mathsf{split}(R)} \mathsf{B}v'\mathsf{E}$ with $u \sim u'$ and $v \sim v'$. Thus, for every infinite sequence $[w_1] \circ\!\!\to_R [w_2] \circ\!\!\to_R \cdots$, there exists an infinite sequence $\mathsf{B}\ w_1'\ \mathsf{E} \to_{\mathsf{split}(R)} \mathsf{B}\ w_2'\ \mathsf{E} \to_{\mathsf{split}(R)} \cdots$ with $w_i \sim w_i'$ for all i. □

3.2.1. *Completeness of Split.* We use type introduction for string rewriting (see Section 3.1) and use the set of types $\mathbb{T} := \{\mathcal{A}, \overline{\mathcal{A}}, \mathcal{K}, \mathcal{T}\}$ and type the symbols used by $\mathsf{split}(R)$ as follows:

$$\begin{array}{lll} \mathsf{L} : \mathcal{A} \to \overline{\mathcal{A}} & \mathsf{W} : \overline{\mathcal{A}} \to \mathcal{T} & a \quad : \mathcal{A} \to \mathcal{A} \quad \text{for all } a \in \Sigma \\ \mathsf{B} : \mathcal{A} \to \mathcal{T} & \mathsf{E} \ : \mathcal{K} \to \mathcal{A} & \overline{a} \quad : \overline{\mathcal{A}} \to \overline{\mathcal{A}} \quad \text{for all } \overline{a} \in \overline{\Sigma} \\ & & \mathsf{R}_{i,j} : \mathcal{A} \to \overline{\mathcal{A}} \quad \text{for all } \mathsf{R}_{i,j} \end{array}$$

First one can verify that $\mathsf{split}(R)$ is a typed SRS, i.e. the left hand sides and right hand sides are well-typed with the same type: (splitA) rewrites strings of type $\mathcal{A} \to \mathcal{A}$, (splitB) and (splitD) rewrite strings of type $\mathcal{A} \to \overline{\mathcal{A}}$, (splitC) and (splitE) rewrite strings of type $\mathcal{A} \to \mathcal{T}$, and (splitF) rewrites strings of type $\mathcal{K} \to \overline{\mathcal{A}}$. Thus $\mathsf{split}(R)$ is a typed SRS.

Lemma 3.9. *If a typed string w of type $\tau_1 \to \tau_2$ with $\tau_1, \tau_2 \in \mathbb{T}$ admits an infinite reduction w.r.t. $\mathsf{split}(R)$, then there exists a typed string w' of type $\mathcal{K} \to \mathcal{T}$, which admits an infinite reduction w.r.t. $\mathsf{split}(R)$.*

Proof. Assume that w is of type $\tau_1 \to \tau_2 \neq \mathcal{K} \to \mathcal{T}$. Note that $\tau_1 \neq \mathcal{T}$ and $\tau_2 \neq \mathcal{K}$, since no well-typed, non-empty strings of these types exist.

We prepend and append symbols to w, resulting in a string uwv s.t. uwv is well-typed with type $\mathcal{K} \to \mathcal{T}$. If $\tau_2 = \mathcal{A}$ then choose $u = \mathsf{B}$, if $\tau_2 = \overline{\mathcal{A}}$ then choose $u = \mathsf{W}$, otherwise choose $u = \varepsilon$. If $\tau_1 = \mathcal{A}$, then choose $v = \mathsf{E}$, if $\tau_1 = \overline{\mathcal{A}}$ then choose $v = \mathsf{L}\mathsf{E}$.

Clearly, if $w \to_{\mathsf{split}(R)} w_1 \to_{\mathsf{split}(\mathsf{R})} \cdots$ is an infinite reduction w.r.t. $\mathsf{split}(R)$, then also $uwv \to_{\mathsf{split}(R)} uw_1v \to_{\mathsf{split}(\mathsf{R})} \cdots$ is an infinite reduction w.r.t. $\mathsf{split}(R)$. □

Inspecting the typing of the symbols shows:

Lemma 3.10. *Any well-typed string of type $\mathcal{K} \to \mathcal{T}$ is of one of the following forms:*

- $\mathsf{B}u\mathsf{E}$ *where* $u \in \Sigma^*$,
- $\mathsf{W}w\mathsf{L}u\mathsf{E}$ *where* $w \in \overline{\Sigma}^*$ *and* $u \in \Sigma^*$, *or*
- $\mathsf{W}w\mathsf{R}_{i,j}u\mathsf{E}$ *where* $w \in \overline{\Sigma}^*$ *and* $u \in \Sigma^*$.

We define a mapping from well-typed strings of type $\mathcal{K} \to \mathcal{T}$ into untyped strings over Σ as follows:

Definition 3.11. For a string $w : \mathcal{K} \to \mathcal{T}$, the string $\Phi(w) \in \Sigma^*$ is defined according to the cases of Lemma 3.10:

$$\begin{array}{lcll} \Phi(\mathsf{B}u\mathsf{E}) & := & u & \\ \Phi(\mathsf{W}w\mathsf{L}u\mathsf{E}) & := & \underline{w}u & \\ \Phi(\mathsf{W}w\mathsf{R}_{i,j}u\mathsf{E}) & := & \ell_s\underline{w}u & \text{if } (\mathsf{B}\ell_s \to \mathsf{WR}_{i,j}) \in \mathsf{split}(R). \end{array}$$

Lemma 3.12. *Let w be a well-typed string of type $\mathcal{K} \to \mathcal{T}$ and $w \to_{\mathsf{split}(R)} w'$. Then $[\Phi(w)] \circ\!\!\to_R^* [\Phi(w')]$.*

Proof. We inspect the cases of Lemma 3.10 for w:

- If $w = \mathsf{B}u\mathsf{E}$ where $u \in \Sigma^*$, then the step $w \to_{\mathsf{split}(R)} w'$ can use a rule of type (splitA) or (splitE). If rule (splitA) is applied, then $\Phi(w) \to_R \Phi(w')$ and thus $[\Phi(w)] \circ\!\!\to_R [\Phi(w')]$. If rule (splitE) is applied, then $w = \mathsf{B}\ell_2 u' \to_{\mathsf{split}(R)} \mathsf{WR}_{i,j}u' = w'$ and $\Phi(w) = \ell_2 u' = \Phi(w')$ and thus $[\Phi(w)] = [\Phi(w')]$.
- If $w = \mathsf{W}v\mathsf{L}u\mathsf{E}$ where $v \in \overline{\Sigma}^*$ and $u \in \Sigma^*$, then the step $w \to_{\mathsf{split}(R)} w'$ can use rules of type (splitA), (splitB), or (splitC). If rule (splitA) is used, then $\Phi(w) \to_R \Phi(w')$ and thus $[\Phi(w)] \circ\!\!\to_R [\Phi(w')]$. If rule (splitB) or (splitC) is used, then $\Phi(w) = \Phi(w')$ and thus $[\Phi(w)] = [\Phi(w')]$.
- If $w = \mathsf{W}v\mathsf{R}_{i,j}u\mathsf{E}$ where $v \in \overline{\Sigma}^*$ and $u \in \Sigma^*$, then the step $w \to_{\mathsf{split}(R)} w'$ can use a rule of type (splitA), (splitF), or (splitD). If rule (splitA) is used, then $\Phi(w) \to_R \Phi(w')$ and thus $[\Phi(w)] \circ\!\!\to_R [\Phi(w')]$. If rule (splitD) is used, then $\Phi(w) = \Phi(w')$ and thus $[\Phi(w)] = [\Phi(w')]$. If rule (splitF) is used, then $w = \mathsf{W}v\mathsf{R}_{i,j}\ell_p\mathsf{E}$ and $w' = \mathsf{W}v\mathsf{L}r_i\mathsf{E}$ and $\Phi(w) = \ell_s\underline{v}\ell_p \sim \ell_p\ell_s\underline{v} \to_R r_i\underline{v} \sim \underline{v}r_i = \Phi(w')$ and thus $[\Phi(w)] \circ\!\!\to_R [\Phi(w')]$. □

Theorem 3.13 (Soundness and completeness of split)**.** *The transformation* split *is sound and complete, i.e. $\to_{\mathsf{split}(R)}$ is terminating if, and only if $\circ\!\!\to_R$ is terminating.*

Proof. Soundness is proved in Proposition 3.8. It remains to show completeness. W.l.o.g. we assume that $\to_R$ is terminating, since otherwise $\circ\!\!\to_R$ is obviously non-terminating. Type introduction (Corollary 3.4) and Lemma 3.9 show that it is sufficient to construct a non-terminating cycle rewrite sequence for any well-typed string w of type $\mathcal{K} \to \mathcal{T}$ where w has an infinite $\to_{\mathsf{split}(R)}$-reduction. Let w be a well-typed string of type $\mathcal{K} \to \mathcal{T}$ s.t. w admits an infinite reduction $w \to_{\mathsf{split}(R)} w_1 \to_{\mathsf{split}(R)} w_2 \to_{\mathsf{split}(R)} \cdots$. Lemma 3.12 shows that the cycle rewrite sequence $[\Phi(w)] \circ\!\!\to_R^* [\Phi(w_1)] \circ\!\!\to_R^* [\Phi(w_2)] \circ\!\!\to_R^* \cdots$ exists. It remains to show that the constructed sequence is infinite. One can observe that the infinite $\to_{\mathsf{split}(R)}$-sequence starting with w must have infinitely many applications of rule (splitF), since every sequence of $\xrightarrow{(\text{splitA})\vee(\text{splitB})\vee(\text{splitC})\vee(\text{splitD})\vee(\text{splitE})}$-steps is terminating (since we assumed that $\to_R$ is terminating). Since $\Phi(\cdot)$ translates an (splitF)-step into exactly one $\circ\!\!\to_R$-step, the sequence $[\Phi(w)] \circ\!\!\to_R^* [\Phi(w_1)] \circ\!\!\to_R^* [\Phi(w_2)] \circ\!\!\to_R^* \ldots$ consists of infinitely many $\circ\!\!\to_R$-steps. □

Remark 3.14. Note that simulating a cycle rewrite step $w \circ\!\!\to_R w'$ requires $O(|w|)$ $\to_{\mathsf{split}(R)}$-steps. Thus the derivation height of $\mathsf{split}(R)$ (i.e. the length of a maximal string rewrite sequence for $\mathsf{B}w\mathsf{E}$) is asymptotically the square of the derivation height of R (i.e. the length of a maximal cycle rewrite sequence for w). Note also that the same arguments and properties apply to the two other transformations, shift and rotate, which will be presented later.

3.2.2. *Relative Termination.* We discuss how the transformation split can be adapted to show relative termination of cycle rewriting. With $splitA(\cdot)$ (and $splitF(\cdot)$, resp.) we denote the rules which are generated by the transformation $\mathsf{split}(\cdot)$ according to rule (splitA) (and (splitF), resp.). The idea of using the transformation split for relative termination, is to transform $S \subseteq R$ into $\mathsf{split}(R)$ where, only the rules corresponding to $splitA(S)$ and $splitF(S)$ become strict rules, while all other rules are weak rules in the transformed system.

Definition 3.15. Let $S \subseteq R$ be SRSs over alphabet Σ. The transformation split_{rel} is defined as follows:

$$\mathsf{split}_{rel}(S,R) \;:=\; (splitA(S) \cup splitF(S), \mathsf{split}(R))$$

Example 3.16. Let $S = \{abc \to bac\}$ and $R = S \cup \{bc \to\!= cb\}$. Then $\mathsf{split}_{rel}(S,R) = (S',R')$ with

$$\begin{aligned} R' = \;& \{bc \to\!= cb, \mathsf{B}c \to\!= \mathsf{WR}_{1,1}, \mathsf{B}bc \to\!= \mathsf{WR}_{2,1}, \mathsf{B}c \to\!= \mathsf{WR}_{2,2}, \mathsf{R}_{1,1}b\mathsf{E} \to\!= \mathsf{L}cb\mathsf{E},\\ & \mathsf{R}_{1,1}b \to\!= \overline{b}\mathsf{R}_{1,1}, \mathsf{R}_{2,1}b \to\!= \overline{b}\mathsf{R}_{2,1}, \mathsf{R}_{2,2}b \to\!= \overline{b}\mathsf{R}_{2,2}, \mathsf{R}_{1,1}c \to\!= \overline{c}\mathsf{R}_{1,1}, \mathsf{R}_{2,1}c \to\!= \overline{c}\mathsf{R}_{2,1},\\ & \mathsf{R}_{2,2}c \to\!= \overline{c}\mathsf{R}_{2,2}, \mathsf{R}_{1,1}a \to\!= \overline{a}\mathsf{R}_{1,1}, \mathsf{R}_{2,1}a \to\!= \overline{a}\mathsf{R}_{2,1}, \mathsf{R}_{2,2}a \to\!= \overline{a}\mathsf{R}_{2,2}, \overline{b}\mathsf{L} \to\!= \mathsf{L}b,\\ & \overline{c}\mathsf{L} \to\!= \mathsf{L}c, \overline{a}\mathsf{L} \to\!= \mathsf{L}a, \mathsf{WL} \to\!= \mathsf{B}\} \cup S'\\ S' = \;& \{\mathsf{R}_{2,1}a\mathsf{E} \to \mathsf{L}bac\mathsf{E}, \mathsf{R}_{2,2}ab\mathsf{E} \to \mathsf{L}bac\mathsf{E}, abc \to bac\} \end{aligned}$$

The termination provers AProVE and T$_{\text{T}}$T$_2$ automatically show that S' is string terminating relative to R'. Since – as we show below – split_{rel} is sound for relative termination, we can conclude that S is cycle terminating relative to R.

As another example, consider $S = \{aa \to aba\}$ and $R = S \cup \{ab \to\!= ba\}$, and let $\mathsf{split}_{rel}(S,R) = (S',R')$. Both provers show that S' is not terminating relative to R'. Since the transformation is also complete for relative termination, we can conclude that S is not cycle terminating relative to R, which can also be verified by the following counter-example: the infinite cycle rewrite sequence $[aa] \circ\!\!\to_S [aba] \circ\!\!\to_R [aab] \circ\!\!\to_S [abab] \circ\!\!\to_R [aabb] \cdots$ has infinitely many $\circ\!\!\to_S$-steps.

For $S \subseteq R$ and $\mathsf{split}_{rel}(S,R) = (S',R')$, we will show that S is cycle terminating relative to R if, and only if S' is string terminating relative to R'.

First note that for every SRS R and every cycle rewrite step, $v_1 \circ\!\!\to_R v_2$ implies $\mathsf{B}v_1'\mathsf{E} \to^+_{\mathsf{split}(R)} \mathsf{B}v_2'\mathsf{E}$ where $v_j \sim v_j'$ for $j = 1,2$ and the $\to_{\mathsf{split}(R)}$-sequence contains exactly one application of rule $splitA(R)$ or $splitF(R)$ (see Proposition 3.8 and inspect the construction of split).

Proposition 3.17. *The transformation* split_{rel} *is sound for relative termination.*

Proof. Let Σ_A be an alphabet, $S \subseteq R$ be SRSs over Σ_A, and $\mathsf{split}_{rel}(S,R) = (S',R')$. Assume that S is not cycle terminating relative to R. Then there exists an infinite reduction $[w_i] \circ\!\!\to^*_R [u_i] \circ\!\!\to_S [w_{i+1}]$ for $i = 1,2,\ldots$. Then $\mathsf{B}w_i'\mathsf{E} \to^*_{\mathsf{split}(R)} \mathsf{B}u_i'\mathsf{E} \to^+_{\mathsf{split}(S)} \mathsf{B}w_{i+1}'\mathsf{E}$ with $w_i \sim w_i', w_{i+1} \sim w_{i+1}'$ and $u_i \sim u_i'$. The sequence $u_i' \to^+_{\mathsf{split}(S)} w_{i+1}$ uses exactly once the rule $splitA(S)$ or the rule $splitF(S)$. Thus we have $\mathsf{B}w_i'\mathsf{E} \to^*_{R'} \mathsf{B}u_i'\mathsf{E} \to^*_{R'} \cdot \to_{S'} \cdot \to^*_{R'} \mathsf{B}w_{i+1}'\mathsf{E}$ for all $i = 1,2,\ldots$. Hence S' is not string terminating relative to R'. □

For proving completeness, we again use the typed variant of the string rewrite system.

Theorem 3.18. *The transformation* split_{rel} *is sound and complete for relative termination.*

Proof. Soundness was proved in Proposition 3.17. For completeness, consider $S \subseteq R$ with $\mathsf{split}_{rel}(S, R) = (S', R')$ and assume that a string w_1 of type $\mathcal{K} \to \mathcal{T}$ has an infinite reduction of the form $w_i \to_{S'} u_i \to^*_{R'} w_{i+1}$ for all $i = 1, 2, \ldots$ (which is sufficient due to Theorem 3.3 and Proposition 2.5).

We first consider the first reductions of the form $w_i \to_{S'} u_i$. If $w_i \to_{S'} u_i$ by rule $splitA(S)$, then clearly $[\Phi(w_i)] \circ\!\!\to_S [\Phi(u_i)]$. If $w_i \to_{S'} u_i$ by rule $splitF(S)$, then due to typing $w_i = \mathsf{W}v\mathsf{R}_{i,k}\ell_p\mathsf{E}$ and $u_i = \mathsf{W}v\mathsf{L}r_j\mathsf{E}$. Applying $\Phi(\cdot)$ to w_i and u_i shows that $\Phi(w_i) = \ell_s\underline{v}\ell_p \sim \ell_p\ell_s\underline{v} = \ell_j\underline{v}$ and $\Phi_S(u_i) = \underline{v}r_j$ which implies $[\Phi(w_i)] \circ\!\!\to_S [\Phi(u_i)]$.

Now, for the reduction sequences $u_i \to^*_{R'} w_{i+1}$, Lemma 3.12 shows that $[\Phi(u_i)] \circ\!\!\to^*_R [\Phi(w_{i+1})]$ holds. Thus, $[\Phi(w_i)] \circ\!\!\to_S . \circ\!\!\to^*_R [\Phi(w_{i+1})]$ for all $i = 1, 2, \ldots$. Hence, S is not cycle terminating relative to R. □

3.3. **The Transformation Shift.** We first present the general ideas of the transformation shift before giving its definition. We write $\curvearrowright$ for the relation which moves the first element of a string to the end, i.e. $au \curvearrowright ua$ for every $a \in \Sigma$ and $u \in \Sigma^*$. Clearly, $u \sim v$ if and only if $u \curvearrowright^{<|u|} v$.

For a string rewrite system R, we define $\mathrm{len}(R)$ as the size of the largest left-hand side of the rules in R, i.e. $\mathrm{len}(R) = \max_{(\ell\to r)\in R} |\ell|$.

The approach of the transformation shift is to shift at most $\mathrm{len}(R) - 1$ symbols from the left end to the right end and then to apply a string rewrite step (i.e. this the relation $\curvearrowright^{<\mathrm{len}(R)} . \to_R$).

Example 3.19. As in Example 3.5, let $R = \{abc \to bbbb\}$ and $[bcdda] \circ\!\!\to_R [bbddbb]$.

The approach of transformation shift is to simulate the cycle rewrite step, by first shifting symbols from the left end to the right end of the string until abc becomes a substring, and then applying a string rewrite step, i.e. $bcdda \curvearrowright^{<\mathrm{len}(R)} . \to_R ddbbbb$, since $bcdda \curvearrowright cddab \curvearrowright ddabc \to_R ddbbbb$.

The following lemma obviously holds:

Lemma 3.20. *Let R be an SRS over an alphabet Σ and $u, v \in \Sigma^*$: If $[u] \circ\!\!\to_R [v]$ then there exist $u', v' \in \Sigma^*$ s.t. $u \curvearrowright^* u' \to_R v' \curvearrowright^* v$.*

Moreover, we can restrict the number of $\curvearrowright$-steps before a rewrite step: For $u, v, w \in \Sigma^*$, if $u \curvearrowright^k v \to_R w$ for some $k \geq \mathrm{len}(R)$, then $u \curvearrowright^m v' \to_R v'' \curvearrowright^* w$ for some $v', v'' \in \Sigma^*$ and $m < \mathrm{len}(R)$.

Using the previous lemmas, we are able to prove the following proposition:

Proposition 3.21. *Let R be an SRS. If $\circ\!\!\to_R$ is non-terminating, then $\curvearrowright^{<\mathrm{len}(R)} . \to_R$ admits an infinite reduction.*

Proof. Let $[w_1] \circ\!\!\to_R [w_2] \circ\!\!\to_R \cdots$ be an infinite cycle rewrite sequence. By Lemma 3.20 we have $w_1 \curvearrowright^* w_1' \to_R w_1'' \curvearrowright^* w_2 \curvearrowright^* w_2' \to_R w_2'' \curvearrowright^* \cdots$ and by the second part of the lemma, we can always move more than $\mathrm{len}(R)$ $\curvearrowright$-steps to the right, and thus we derive an infinite sequence $w_1 \curvearrowright^{<\mathrm{len}(R)} u_1 \to_R u_1' \curvearrowright^{<\mathrm{len}(R)} u_2 \to_R u_2' \cdots$ and thus w_1 admits an infinite reduction of the required form. □

For an SRS R, the SRS $\mathsf{shift}(R)$ encodes the relation $\curvearrowright^{<\mathrm{len}(R)} . \to_R$ where extra symbols are used to separate the steps, and copies of the alphabet underlying R are used to ensure completeness of the transformation.

For the remainder of the section, we fix an SRS R over alphabet $\Sigma_A = \{a_1, \ldots, a_n\}$. Let us write Σ_B, Σ_C for fresh copies of the alphabet Σ_A. We use the following notation to switch between the alphabets: for $X, Y \in \{A, B, C\}$ and $w \in \Sigma_X$ we write $(\!|w|\!)_Y$ to denote the copy of w in the alphabet Y where every symbol is translated from alphabet X to alphabet Y.

Definition 3.22 (The transformation shift)**.** Let R be an SRS over alphabet Σ_A and let $N = \max(0, \mathrm{len}(R)-1)$. The SRS $\mathsf{shift}(R)$ over the alphabet $\Sigma_A \cup \Sigma_B \cup \Sigma_C \cup \{\mathsf{B}, \mathsf{E}, \mathsf{W}, \mathsf{V}, \mathsf{M}, \mathsf{L}, \mathsf{R}, \mathsf{D}\}$ (where $\mathsf{B}, \mathsf{E}, \mathsf{W}, \mathsf{V}, \mathsf{M}, \mathsf{L}, \mathsf{R}, \mathsf{D}$ are fresh for $\Sigma_A \cup \Sigma_B \cup \Sigma_C$) consists of the following rules:

$$\mathsf{B} \to \mathsf{W}\mathsf{M}^N\mathsf{V} \quad \text{(shiftA)} \qquad \mathsf{WV} \to \mathsf{RL} \quad \text{(shiftF)}$$

$$\mathsf{M} \to \varepsilon \quad \text{(shiftB)} \qquad \mathsf{L}a \to (\!|a|\!)_C\mathsf{L} \text{ for all } a \in \Sigma_A \quad \text{(shiftG)}$$

$$\mathsf{MV}a \to \mathsf{V}(\!|a|\!)_B \text{ for all } a \in \Sigma_A \quad \text{(shiftC)} \qquad \mathsf{L}\ell \to \mathsf{D}r \text{ for all } (\ell \to r) \in R \quad \text{(shiftH)}$$

$$ba \to ab \text{ for all } a \in \Sigma_A,\, b \in \Sigma_B \quad \text{(shiftD)} \qquad c\mathsf{D} \to \mathsf{D}(\!|c|\!)_A \text{ for all } c \in \Sigma_C \quad \text{(shiftI)}$$

$$b\mathsf{E} \to (\!|b|\!)_A\mathsf{E} \text{ for all } b \in \Sigma_B \quad \text{(shiftE)} \qquad \mathsf{RD} \to \mathsf{B} \quad \text{(shiftJ)}$$

The rules (shiftA) - (shiftE) encode the relation $\curvearrowright^{<\mathrm{len}(R)}$, i.e. for $uv \in \Sigma_A^*$ with $|u| < \mathrm{len}(R)$, the string $\mathsf{B}uv\mathsf{E}$ is rewritten into $\mathsf{WV}vu\mathsf{E}$ by these five rules. The sequence of symbols M generated by rule (shiftA) denotes the potential of moving at most $\mathsf{len}(R)-1$ symbols. The rules (shiftB) and (shiftC) either remove one from the potential or start the moving of one symbol. The rule (shiftD) performs the movement of a single symbol until it reaches the end of the string and rule (shiftE) finishes the movement.

The remaining rewrite rules perform a single string rewrite step, i.e. for a rule $(\ell \to r) \in R$ the string $\mathsf{WV}w_1\ell w_2\mathsf{E}$ is rewritten to $\mathsf{B}w_1rw_2\mathsf{E}$ by rules (shiftF) - (shiftJ).

Example 3.23. For $R_1 = \{aa \to aba\}$, the transformed string rewrite system $\mathsf{shift}(R_1)$ is:

$$\begin{array}{llll}
\mathsf{B} \to \mathsf{WMV} & \text{(shiftA)} & \mathsf{M} \to \varepsilon & \text{(shiftB)} \\
\mathsf{MV}b \to \mathsf{V}(\!|b|\!)_B & \text{(shiftC)} & (\!|a|\!)_B a \to a(\!|a|\!)_B & \text{(shiftD)} \\
(\!|b|\!)_B a \to a(\!|b|\!)_B & \text{(shiftD)} & (\!|b|\!)_B b \to b(\!|b|\!)_B & \text{(shiftD)} \\
(\!|b|\!)_B\mathsf{E} \to b\mathsf{E} & \text{(shiftE)} & \mathsf{WV} \to \mathsf{RL} & \text{(shiftF)} \\
\mathsf{L}b \to (\!|b|\!)_C\mathsf{L} & \text{(shiftG)} & \mathsf{L}aa \to \mathsf{D}aba & \text{(shiftH)} \\
(\!|b|\!)_C\mathsf{D} \to \mathsf{D}b & \text{(shiftI)} & \mathsf{RD} \to \mathsf{B} & \text{(shiftJ)}
\end{array}$$

$$\begin{array}{lll}
\mathsf{MV}a \to \mathsf{V}(\!|a|\!)_B & \text{(shiftC)} \\
(\!|a|\!)_B b \to b(\!|a|\!)_B & \text{(shiftD)} \\
(\!|a|\!)_B\mathsf{E} \to a\mathsf{E} & \text{(shiftE)} \\
\mathsf{L}a \to (\!|a|\!)_C\mathsf{L} & \text{(shiftG)} \\
(\!|a|\!)_C\mathsf{D} \to \mathsf{D}a & \text{(shiftI)}
\end{array}$$

The cycle rewrite step $[aba] \circ\!\!\to_{R_1} [baba]$ is simulated in the transformed system as follows:

$$\begin{aligned}
&\mathsf{B}aba\mathsf{E} \xrightarrow{shiftA} \mathsf{WMV}aba\mathsf{E} \xrightarrow{shiftC} \mathsf{WV}(\!|a|\!)_B ba\mathsf{E} \xrightarrow{shiftD} \mathsf{WV}b(\!|a|\!)_B a\mathsf{E} \xrightarrow{shiftD} \mathsf{WV}ba(\!|a|\!)_B\mathsf{E} \\
&\xrightarrow{shiftE} \mathsf{WV}baa\mathsf{E} \xrightarrow{shiftF} \mathsf{RL}baa\mathsf{E} \xrightarrow{shiftG} \mathsf{R}(\!|b|\!)_C\mathsf{L}aa\mathsf{E} \xrightarrow{shiftH} \mathsf{R}(\!|b|\!)_C\mathsf{D}aba\mathsf{E} \\
&\xrightarrow{shiftI} \mathsf{RD}baba\mathsf{E} \xrightarrow{shiftJ} \mathsf{B}baba\mathsf{E}
\end{aligned}$$

Together with Proposition 3.21, the following lemma implies soundness of the transformation shift.

Lemma 3.24. *If* $u \curvearrowright^{<\mathrm{len}(R)} v \to_R w$*, then* $\mathsf{B}u\mathsf{E} \to^+_{\mathsf{shift}(R)} \mathsf{B}w\mathsf{E}$.

Proof. Let $\mathrm{len}(R) = m+1$, $u = a_1 \dots a_n$ and $u \curvearrowright^k a_{k+1} \dots a_n\ a_1 \dots a_k = v$, and let $v = a'_1 \dots a'_n$ and $w = a'_1 \dots a'_i a''_1 \dots a''_r a'_{i+j} \dots a'_n$, i.e. the applied rewrite rule is $a'_{i+1} \dots a'_{i+j-1} \to a''_1 \dots a''_r$. Then the following rewrite sequence using $\mathsf{shift}(R)$ can be constructed:

$$
\begin{aligned}
&\mathsf{B}a_1 \dots a_n\mathsf{E} \xrightarrow{\text{(shiftA)}} \mathsf{W}\mathsf{M}^m\mathsf{V}a_1 \dots a_n\mathsf{E} \xrightarrow{\text{(shiftB)}}{}^* \mathsf{W}\mathsf{M}^k\mathsf{V}a_1 \dots a_n\mathsf{E}\\
&\xrightarrow{\text{(shiftC)}} \mathsf{W}\mathsf{M}^{k-1}\mathsf{V}\langle\!|a_1|\!\rangle_B a_2 \dots a_n\mathsf{E} \xrightarrow{\text{(shiftD)}}{}^* \mathsf{W}\mathsf{M}^{k-1}\mathsf{V}a_2 \dots a_n\langle\!|a_1|\!\rangle_B\mathsf{E}\\
&\xrightarrow{\text{(shiftE)}} \mathsf{W}\mathsf{M}^{k-1}\mathsf{V}a_2 \dots a_n a_1\mathsf{E}\left(\xrightarrow{\text{(shiftC)}} . \xrightarrow{\text{(shiftD)}}{}^* . \xrightarrow{\text{(shiftE)}}\right)^* \mathsf{W}\mathsf{V}a_{k+1} \dots a_n a_1 \dots a_k\mathsf{E}\\
&\xrightarrow{\text{(shiftF)}} \mathsf{R}\mathsf{L}a_{k+1} \dots a_n a_1 \dots a_k\mathsf{E} = \mathsf{R}\mathsf{L}a'_1 \dots a'_n\mathsf{E} \xrightarrow{\text{(shiftG)}}{}^* \mathsf{R}\langle\!|a'_1|\!\rangle_C \dots \langle\!|a'_i|\!\rangle_C\mathsf{L}a'_{i+1} \dots a'_n\mathsf{E}\\
&\xrightarrow{\text{(shiftH)}} \mathsf{R}\langle\!|a'_1|\!\rangle_C \dots \langle\!|a'_i|\!\rangle_C\mathsf{D}a''_1 \dots a''_r a'_{i+j} \dots a'_n\mathsf{E} \xrightarrow{\text{(shiftI)}}{}^* \mathsf{R}\mathsf{D}a'_1 \dots a'_i a''_1 \dots a''_r a'_{i+j} \dots a'_n\mathsf{E}\\
&\xrightarrow{\text{(shiftJ)}}{}^* \mathsf{B}a'_1 \dots a'_i a''_1 \dots a''_r a'_{i+j} \dots a'_n\mathsf{E}
\end{aligned}
$$

□

We prove completeness by using type introduction. Let us write $\widetilde{\Sigma_A} := \Sigma_A \cup \Sigma_B \cup \Sigma_C \cup \{\mathsf{B},\mathsf{E},\mathsf{W},\mathsf{V},\mathsf{M},\mathsf{L},\mathsf{R},\mathsf{D}\}$. Let $\mathbb{T} := \{\mathcal{AB},\mathcal{C},\mathcal{K},\mathcal{M},\mathcal{T}\}$ be the set of types. We assign the following types to the symbols of $\widetilde{\Sigma_A}$:

a	$: \mathcal{AB} \to \mathcal{AB}$	for all $a \in \Sigma_A$	E	$: \mathcal{K} \to \mathcal{AB}$	W	$: \mathcal{M} \to \mathcal{T}$	D	$: \mathcal{AB} \to \mathcal{C}$
b	$: \mathcal{AB} \to \mathcal{AB}$	for all $b \in \Sigma_B$	V	$: \mathcal{AB} \to \mathcal{M}$	B	$: \mathcal{AB} \to \mathcal{T}$	R	$: \mathcal{C} \to \mathcal{T}$
c	$: \mathcal{C} \to \mathcal{C}$	for all $c \in \Sigma_C$	M	$: \mathcal{M} \to \mathcal{M}$	L	$: \mathcal{AB} \to \mathcal{C}$		

We verify that $\mathsf{shift}(R)$ is a typed SRS. Rules (shiftA), (shiftF), and (shiftJ) rewrite strings of type $\mathcal{AB} \to \mathcal{T}$, rule (shiftB) rewrites strings of type $\mathcal{M} \to \mathcal{M}$, rule (shiftC) rewrites strings of type $\mathcal{AB} \to \mathcal{M}$, rule (shiftD) rewrites strings of type $\mathcal{AB} \to \mathcal{AB}$, rule (shiftE) rewrites strings of type $\mathcal{K} \to \mathcal{AB}$, and rules (shiftG), (shiftH), and (shiftI) rewrite strings of type $\mathcal{AB} \to \mathcal{C}$.

Lemma 3.25. *Let $w \in \widetilde{\Sigma_A}^*$ be a well-typed string, s.t. w admits an infinite reduction w.r.t. $\to_{\mathsf{shift}(R)}$. Then there exists a well-typed string $w' \in \widetilde{\Sigma_A}^*$ of type $\mathcal{K} \to \mathcal{T}$ which admits an infinite reduction w.r.t. $\to_{\mathsf{shift}(R)}$.*

Proof. Let w be a well-typed string of type $\tau_1 \to \tau_2 \neq \mathcal{K} \to \mathcal{T}$ where $\tau_1, \tau_2 \in \mathbb{T}$ s.t. w admits an infinite reduction w.r.t. $\to_{\mathsf{shift}(R)}$. We prepend and append symbols to w constructing a string $w' = uwv$ of type $\mathcal{K} \to \mathcal{T}$ as follows:

The string u is the empty string if $\tau_2 = \mathcal{T}$ and otherwise for any $\tau_2 \in \{\mathcal{K},\mathcal{AB},\mathcal{M},\mathcal{C}\}$ there is a sequence of type $\tau_2 \to \mathcal{T}$, which is used for u:

$$\mathsf{R} : \mathcal{C} \to \mathcal{T} \qquad \mathsf{W} : \mathcal{M} \to \mathcal{T} \qquad \mathsf{B} : \mathcal{AB} \to \mathcal{T} \qquad \mathsf{BE} : \mathcal{K} \to \mathcal{T}$$

The string v is the empty string if $\tau_1 = \mathcal{K}$ and otherwise for any $\tau_1 \in \{\mathcal{AB},\mathcal{M},\mathcal{C},\mathcal{T}\}$ there is a sequence of type $\mathcal{K} \to \tau_1$, which is used for v:

$$\mathsf{E} : \mathcal{K} \to \mathcal{AB} \qquad \mathsf{VE} : \mathcal{K} \to \mathcal{M} \qquad \mathsf{DE} : \mathcal{K} \to \mathcal{C} \qquad \mathsf{BE} : \mathcal{K} \to \mathcal{T}$$

Clearly, the infinite reduction for w can be used to construct an infinite reduction for uwv. □

By checking all possible cases, the following lemma can be verified:

Lemma 3.26. *Any well-typed string of type $\mathcal{K} \to \mathcal{T}$ is of one of the following forms:*
- $\mathsf{W}\mathsf{M}^i\mathsf{V}w\mathsf{E}$ *where* $i \geq 0$ *and* $w \in (\Sigma_A \cup \Sigma_B)^*$,
- $\mathsf{B}w\mathsf{E}$ *where* $w \in (\Sigma_A \cup \Sigma_B)^*$,
- $\mathsf{R}w_c\mathsf{L}w\mathsf{E}$ *where* $w_c \in \Sigma_C^*$ *and* $w \in (\Sigma_A \cup \Sigma_B)^*$, *or*
- $\mathsf{R}w_c\mathsf{D}w\mathsf{E}$ *where* $w_c \in \Sigma_C^*$ *and* $w \in (\Sigma_A \cup \Sigma_B)^*$.

For a string $w \in (\Sigma_A \cup \Sigma_B)^*$, let $\pi_A(w)$ be the string w where all symbols $b \in \Sigma_B$ are removed and let $\overline{\pi}_B(w)$ be the reversed string of w' where w' is w where all symbols $a \in \Sigma_A$ are removed.

Definition 3.27. For a well-typed string $w : \mathcal{K} \to \mathcal{T}$, the mapping $\Phi_S(w) \in \Sigma_A^*$ is defined according to the cases of Lemma 3.26 as follows:

$$\begin{array}{lcllcl} \Phi_S(\mathsf{W}\mathsf{M}^i\mathsf{V}w\mathsf{E}) & := & \pi_A(w)\overline{\pi}_B(w) & \Phi_S(\mathsf{B}w\mathsf{E}) & := & \pi_A(w)\overline{\pi}_B(w) \\ \Phi_S(\mathsf{R}w_c\mathsf{L}w\mathsf{E}) & := & (\!|w_c|\!)_A\pi_A(w)\overline{\pi}_B(w) & \Phi_S(\mathsf{R}w_c\mathsf{D}w\mathsf{E}) & := & (\!|w_c|\!)_A\pi_A(w)\overline{\pi}_B(w) \end{array}$$

Lemma 3.28. *Let u be a well-typed string of type $\mathcal{K} \to \mathcal{T}$. If $u \to_{\mathsf{shift}(R)} v$, then $[\Phi_S(u)] \circ\!\!\to_R^* [\Phi_S(v)]$.*

Proof. We go through the cases of Lemma 3.26 and inspect all applicable rewrite rules:
- If $u = \mathsf{W}\mathsf{M}^i\mathsf{V}w\mathsf{E}$, then there are the following cases: If rule (shiftB), (shiftD), (shiftE), or (shiftF) is applied, then $\Phi_S(u) = \Phi_S(v)$ and if rule (shiftC) is applied, then $\Phi_S(u) = aw_1w_2$ (with $w = aw'$ and $\pi_A(w) = aw_1$ and $\overline{\pi}_B(w) = w_2$) and $\Phi_S(v) = w_1w_2a$ and thus $\Phi_S(u) \sim \Phi_S(v)$.
- If $u = \mathsf{B}w\mathsf{E}$, then rules (shiftA), (shiftD), or (shiftE) may be applied and in all cases $\Phi_S(u) = \Phi_S(v)$ holds.
- For $u = \mathsf{R}w_c\mathsf{L}w\mathsf{E}$, there are two cases: If rule (shiftD), (shiftE), or (shiftG) is applied, then $\Phi_S(u) = \Phi_S(v)$. If rule (shiftH) is applied, then $\Phi_S(u) \to_R \Phi_S(v)$.
- If $u = \mathsf{R}w_c\mathsf{D}w\mathsf{E}$, then rules (shiftD), (shiftE), (shiftI), and (shiftJ) may be applied and in all cases $\Phi_S(u) = \Phi_S(v)$ holds. □

Proposition 3.29. *Let $u : \mathcal{K} \to \mathcal{T}$ s.t. u admits an infinite $\to_{\mathsf{shift}(R)}$-reduction. Then $[u]$ admits an infinite $\circ\!\!\to_R$-reduction.*

Proof. Suppose that $u : \mathcal{K} \to \mathcal{T}$ admits an infinite $\to_{\mathsf{shift}(R)}$-reduction. Applying Lemma 3.28 shows that it is possible to construct an infinite reduction of $\circ\!\!\to_R^*$-steps starting with $[u]$. Since applications of rule (shiftH) are translated into $\circ\!\!\to_R$ steps, it remains to show that the given infinite reduction of u has infinitely many applications of rule (shiftH). Therefore we show that the rewrite system consisting of all rules of $\mathsf{shift}(R)$ except for rule (shiftH) is terminating. Let us denote this system by $\mathsf{shift}_{\setminus\{(\text{shiftH})\}}(R)$, i.e. $\mathsf{shift}_{\setminus\{(\text{shiftH})\}}(R) := (\mathsf{shift}(R) \setminus \{\ell \xrightarrow{(\text{shiftH})} r\})$. For proving termination of $\mathsf{shift}_{\setminus\{(\text{shiftH})\}}(R)$ we first eliminate rule (shiftA): Let σ_0 be the polynomial interpretation defined by $\sigma_0(B) = \sigma_0(D) = \lambda x.x+1$ and $\sigma_0(y) = \lambda x.x$ for all other symbols y. It is easy to verify that $\sigma_0(\ell) > \sigma_0(r)$ for $\ell \xrightarrow{(\text{shiftA})} r$ and $\sigma_0(\ell) \geq \sigma_0(r)$ for $\ell \to r \in \mathsf{shift}_{\setminus\{(\text{shiftH})\}}(R)$. Thus it suffices to prove termination of $\mathsf{shift}_{\setminus\{(\text{shiftA}),(\text{shiftH})\}}(R) := (\mathsf{shift}(R) \setminus \{\ell \xrightarrow{(\text{shiftH})} r, \ell \xrightarrow{(\text{shiftA})} r\})$: We use the

following polynomial interpretation σ. Let $N = \max\{1, \mathrm{len}(R)\}$ and

$$\begin{array}{lll}
\sigma(a) = \lambda x.4x+4 \text{ for all } a \in \Sigma_A & \sigma(\mathsf{D}) = \lambda x.2^N x + 2^{N+1} - 1 & \sigma(\mathsf{M}) = \lambda x.2x+1 \\
\sigma(b) = \lambda x.8x+1 \text{ for all } b \in \Sigma_B & \sigma(\mathsf{R}) = \lambda x.2x+1 & \sigma(\mathsf{L}) = \lambda x.x \\
\sigma(c) = \lambda x.4x \text{ for all } c \in \Sigma_C & \sigma(\mathsf{E}) = \lambda x.7x+1 & \sigma(\mathsf{W}) = \lambda x.2x+2 \\
\sigma(\mathsf{B}) = \lambda x.2^N x + 2^{N+1} - 1 & \sigma(\mathsf{V}) = \lambda x.x &
\end{array}$$

We verify that the inequation $\sigma(\ell) > \sigma(r)$ holds for all rules of $\mathsf{shift}_{\setminus\{(\mathrm{shiftA}),(\mathrm{shiftH})\}}(R)$: For rule (shiftB), we have $\lambda x.2x+1 > \lambda x.x$, for rule (shiftC), we have $\lambda x.8x+9 > \lambda x.8x+1$, for rule (shiftD), we have $\lambda x.32x+33 > \lambda x.32x+8$, for rule (shiftE), we have $\lambda x.56x+9 > \lambda x.28x+8$, for rule (shiftF), we have $\lambda x.2x+2 > \lambda x.2x+1$, for rule (shiftG), we have $\lambda x.4x+4 > \lambda x.4x$, for rule (shiftI), we have $\lambda x.2^{N+2}x + 3\cdot 2^{N+1} + 2^{N+1} - 4 > \lambda x.2^{N+2}x + 3\cdot 2^{N+1} - 1$, and for rule (shiftJ), we have $\lambda x.2^{N+1}x + 2^{N+2} - 1 > \lambda x.2^N x + 2^{N+1} - 1$. □

Theorem 3.30. *The transformation* shift *is sound and complete.*

Proof. Soundness follows by Lemma 3.24 and Proposition 3.21. Completeness follows by type introduction (Corollary 3.4), Proposition 3.29, and Lemma 3.25. □

3.3.1. *Relative Termination.* We also provide a variation of the transformation shift to encode relative cycle termination by relative string termination.

Definition 3.31. Let $S \subseteq R$ be SRSs over an alphabet Σ. The transformation shift_{rel} is defined as:

$$\mathsf{shift}_{rel}(S,R) \;:=\; (\{\mathsf{L}\ell \to \mathsf{D}r \mid (\ell \to r) \in S\}, \mathsf{shift}(R))$$

Theorem 3.32. *The transformation* shift_{rel} *is sound and complete for relative termination.*

Proof. Let Σ_A be an alphabet, $S \subseteq R$ be SRSs over Σ_A, and $\mathsf{shift}_{rel}(S,R) = (S', R')$. For proving soundness of shift_{rel}, assume that S is not cycle terminating relative to R. Using the same arguments as in Proposition 3.21 there exists an infinite reduction such that $w_i\,(\circlearrowright^{<\mathrm{len}(R)} \cdot \to_R)^*.\ \circlearrowright^{<\mathrm{len}(S)} v_i \to_S w_{i+1}$ for $i = 1, 2, \ldots$. By Lemma 3.24 we have for all $i = 1, 2, \ldots$: $\mathsf{B}w_i\mathsf{E} \to^*_{\mathsf{shift}(R)} \mathsf{B}v_i\mathsf{E} \to^+_{\mathsf{shift}(S)} \mathsf{B}w_{i+1}\mathsf{E}$. Since $\to_{\mathsf{shift}(S)} \subseteq \to^*_{R'} \cdot \to_{S'} \cdot \to^*_{R'}$, this shows that S' is not string terminating relative to R'.

For proving completeness, we assume that S' is not string terminating relative to R'. We use typed string rewriting and apply Theorem 3.3 and Lemma 3.25. Thus there exists a typed string $w_0 : \mathcal{K} \to \mathcal{T}$ s.t. for all $i = 0, 1, 2, \ldots$ the reduction sequence $w_i \to_{S'} w'_i \to^*_{R'} w_{i+1}$ exists. Typing of w_i, Lemma 3.26, and applicability of $\to_{S'}$ show that w_i and w'_i must be of the forms $w_i = \mathsf{R}u_i\mathsf{L}\ell v_i\mathsf{E}$ and $w'_i = \mathsf{R}u_i\mathsf{D}r v_i\mathsf{E}$ where $(\ell \to r) \in S$, $u_i \in \Sigma_C$, and $v_i \in (\Sigma_A \cup \Sigma_B)^*$. Since $\Phi_S(w_i) = (\!|u_i|\!)_A \ell \pi_A(v_i) \overline{\pi}_B(v_i)$ and $\Phi_S(w'_i) = (\!|u_i|\!)_A r \pi_A(v_i) \overline{\pi}_B(v_i)$, we have $\Phi_S(w_i) \to_S \Phi_S(w'_i)$ and thus we also have $[\Phi_S(w_i)] \circ\!\!\to_S [\Phi_S(w'_i)]$. Since $R' = \mathsf{shift}(R)$, Lemma 3.28 shows that $[\Phi_S(w'_i)] \circ\!\!\to^*_R [\Phi_S(w_{i+1})]$. Thus for all $i = 1, 2, \ldots$ we have $[\Phi_S(w_i)] \circ\!\!\to_S \cdot \circ\!\!\to^*_R [\Phi_S(w_{i+1})]$ which shows that S is not cycle terminating relative to R. □

3.4. **The Transformation** rotate**.** We first present the idea of the transformation rotate. The transformation is closely related to the definition of $\circ\!\!\rightarrow$: The idea is to first rotate the string and then to apply a prefix rewrite step (i.e. both steps can be expressed by the relation $\sim . \hookrightarrow_R$).

Example 3.33. As in Example 3.5, let $R = \{abc \to bbbb\}$ and $[bcdda] \circ\!\!\rightarrow_R [bbddbb]$.

The transformation rotate simulates the cycle rewrite step, by first rotating the string (using $\sim$) until abc becomes a prefix, and then applies a prefix rewrite step. I.e., we have $bcdda \sim . \hookrightarrow_R bbbbdd$, since $bcdda \sim abcdd \hookrightarrow_R bbbbdd$. It would be possible to use shifts $\curvearrowright$ to perform the rotation, i.e. $bcdda \curvearrowright^* . \hookrightarrow_R bbbbdd$, since $bcdda \curvearrowright cddab \curvearrowright ddabc \curvearrowright dabcd \curvearrowright abcdd \hookrightarrow_R bbbbdd$. However, our transformation rotate will implement $\sim$ by moving symbols from right to left.

The following lemma obviously holds:

Lemma 3.34. *Let R be an SRS over an alphabet Σ and $u, v \in \Sigma^*$: If $[u] \circ\!\!\rightarrow_R [v]$ then there exist $u', v' \in \Sigma^*$ s.t. $u \sim u' \hookrightarrow_R v' \sim v$.*

Proposition 3.35. *Let R be an SRS. If $\circ\!\!\rightarrow_R$ is non-terminating, then $\sim . \hookrightarrow_R$ admits an infinite reduction.*

Proof. Let $[w_1] \circ\!\!\rightarrow_R [w_2] \circ\!\!\rightarrow_R \cdots$ be an infinite cycle rewrite sequence. By Lemma 3.34 we have $w_1 \sim u_1 \hookrightarrow_R v_1 \sim w_2 \sim u_2 \hookrightarrow_R v_2 \sim \cdots$, i.e. (by joining the $\sim$-steps) we get $v_0 = w_1 \sim u_1 \hookrightarrow_R v_1 \sim u_2 \hookrightarrow_R v_2 \sim \cdots$ and thus v_0 admits an infinite reduction of the required form. □

For an SRS R, the SRS $\mathsf{rotate}(R)$ will encode the relation $\sim . \hookrightarrow_R$ where extra symbols are used to separate the steps, and copies of the alphabet underlying R are used to ensure completeness of the transformations. We first introduce the transformation $\mathsf{rotate}(R)$ and then explain the ideas of the transformation in detail.

Again for the rest of the section, we fix an SRS R over alphabet $\Sigma_A = \{a_1, \ldots, a_n\}$, write $\Sigma_B, \Sigma_C, \Sigma_D, \Sigma_E$ for fresh copies of the alphabet Σ_A, and use $(\!|w|\!)_Y$ to switch between the alphabets (see Section 3.3).

Definition 3.36 (The transformation rotate)**.** Let R be an SRS over alphabet Σ_A. The SRS $\mathsf{rotate}(R)$ over the alphabet $\Sigma_A \cup \Sigma_B \cup \Sigma_C \cup \Sigma_D \cup \Sigma_E \cup \{\mathsf{B}, \mathsf{E}, \mathsf{W}, \mathsf{R}, \mathsf{G}, \mathsf{O}, \mathsf{C}, \mathsf{L}, \mathsf{S}, \mathsf{F}, \mathsf{f}\}$ (where B, E, W, R, G, O, C, L, S, F, and f are fresh for $\Sigma_A \cup \Sigma_B \cup \Sigma_C \cup \Sigma_D \cup \Sigma_E$) is:

$$\mathsf{BE} \to \mathsf{WE} \qquad \text{(rotA)}$$

$$\mathsf{B}a \to \mathsf{OC}(\!|a|\!)_D\mathsf{G} \quad \text{for all } a \in \Sigma_A \qquad \text{(rotB)}$$

$$\mathsf{G}a \to (\!|a|\!)_D\mathsf{G} \quad \text{for all } a \in \Sigma_A \qquad \text{(rotC)}$$

$$\mathsf{G}a \to \mathsf{L}(\!|a|\!)_C\mathsf{S} \quad \text{for all } a \in \Sigma_A \qquad \text{(rotD)}$$

$$\mathsf{GE} \to \mathsf{FE} \qquad \text{(rotE)}$$

$$d\mathsf{L}c \to \mathsf{L}c(\!|d|\!)_B \quad \text{for all } c \in \Sigma_C, d \in \Sigma_D \qquad \text{(rotF)}$$

$$\mathsf{CL}c \to (\!|c|\!)_E\mathsf{CR} \quad \text{for all } c \in \Sigma_C \qquad \text{(rotG)}$$

$$\mathsf{R}b \to (\!|b|\!)_D\mathsf{R} \quad \text{for all } b \in \Sigma_B \qquad \text{(rotH)}$$

$$\mathsf{RS}a \to \mathsf{L}(\!|a|\!)_C\mathsf{S} \quad \text{for all } a \in \Sigma_A \qquad \text{(rotI)}$$

$$\mathsf{RSE} \to \mathsf{FE} \qquad \text{(rotJ)}$$

$$d\mathsf{F} \to \mathsf{F}(\!|d|\!)_A \quad \text{for all } d \in \Sigma_D \qquad \text{(rotK)}$$

$$\mathsf{CF} \to \mathsf{f} \qquad \text{(rotL)}$$

$$e\mathsf{f} \to \mathsf{f}(\!|e|\!)_A \quad \text{for all } e \in \Sigma_E \qquad \text{(rotM)}$$

$$\mathsf{O}f \to \mathsf{W} \qquad \text{(rotN)}$$

$$\mathsf{W}\ell \to \mathsf{B}r \quad \text{for all } (\ell \to r) \in R \qquad \text{(rotO)}$$

We describe the intended use of the rewrite rules, where we ignore the copies of the alphabet in our explanations. The goal is that for any string $w \in \Sigma_A^*$, the string $\mathsf{B}w\mathsf{E}$ is rewritten

to $\mathsf{B}u\mathsf{E}$, where $w \sim . \hookrightarrow_R u$. The prefix rewrite step is performed by the last rule (rotO). All other rules perform the rotation $\sim$ s.t. $\mathsf{B}w\mathsf{E}$ is rewritten into $\mathsf{W}v\mathsf{E}$ where $w \sim v$. This is done by moving a suffix of the string in front of the string.

The first rewrite rule (rotA) covers the case that w is empty. Otherwise, if $w = a_1 \dots a_n$, then first choose a position to cut the string into $w_1 w_2$ (the goal is then to form the string $w_2 w_1$). The symbol G is used for the non-deterministic selection of the position. Rule (rotB) starts the rotate phase and the guessing, rule (rotC) shifts the G-marker and rule (rotD) stops the guessing. Rule (rotE) covers the case that $w_2 = \varepsilon$ and no rotation will be performed. After stopping the guessing, every symbol of w_2 is moved in front of w_1, resulting in $w_2 w_1$. A typical situation is $a_{k+1} \dots a_m a_1 \dots a_k a_{m+1} \dots a_n$ and now the symbol a_{m+1} must be moved in between a_m and a_1. To keep track of the position of a_1, the symbol C (inserted in front of a_1) marks the original beginning, and to keep track of the position of a_k, the symbol S (inserted after a_k) marks this position. The symbol L guards the movement of a_{m+1} (by rule (rotF)). When arriving at the right place (rule (rotG)), the symbol R is used to walk along the string (rule (rotH)) to find the next symbol which has to be moved (rule (rotI)). If all symbols are moved, rule (rotJ) is applied to start the clean-up phase. There the symbols F and f are used to remove the markers and to replace the copied symbols of the alphabet with the original ones (rules (rotK) – (rotN)).

In the following, we denote with $\mathrm{rot}(\Sigma_A)$ the string rewrite system consisting of the rules (rotA) – (rotN) from Definition 3.36 (excluding the rule (rotO)).

Example 3.37. For the system $R = \{aaa \to ababa\}$ and $\Sigma_A = \{a, b\}$, the transformed string rewrite system is $\mathsf{rotate}(R) = \mathrm{rot}(\Sigma_A) \cup \{\mathsf{W}aaa \to \mathsf{B}ababa\}$. The cycle rewrite step $[abbaa] \circ\!\!\to_{R_1} [ababab]$ is simulated in the system $\mathsf{rotate}(R)$ by rewriting $\mathsf{B}abbaa\mathsf{E}$ into $\mathsf{B}ababab\mathsf{E}$ as follows:

$$\mathsf{B}abbaa\mathsf{E} \xrightarrow{rotB} \mathsf{O}\mathsf{C}(\!|a|\!)_D\mathsf{G}bbaa\mathsf{E} \xrightarrow{rotC} \mathsf{O}\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D\mathsf{G}baa\mathsf{E} \xrightarrow{rotC} \mathsf{O}\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D(\!|b|\!)_D\mathsf{G}aa\mathsf{E}$$
$$\xrightarrow{rotD} \mathsf{O}\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D(\!|b|\!)_D\mathsf{L}(\!|a|\!)_C\mathsf{S}a\mathsf{E} \xrightarrow{rotF} \mathsf{O}\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D\mathsf{L}\,(\!|a|\!)_C(\!|b|\!)_B\mathsf{S}a\mathsf{E} \xrightarrow{rotF} \mathsf{O}\mathsf{C}(\!|a|\!)_D\mathsf{L}\,(\!|a|\!)_C(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}a\mathsf{E}$$
$$\xrightarrow{rotF} \mathsf{O}\mathsf{C}\mathsf{L}(\!|a|\!)_C(\!|a|\!)_B(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}a\mathsf{E} \xrightarrow{rotG} \mathsf{O}(\!|a|\!)_E\mathsf{C}\mathsf{R}(\!|a|\!)_B(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}a\mathsf{E} \xrightarrow{rotH} \mathsf{O}(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D\mathsf{R}(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}a\mathsf{E}$$
$$\xrightarrow{rotH} \mathsf{O}(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D\mathsf{R}(\!|b|\!)_B\mathsf{S}a\mathsf{E} \xrightarrow{rotH} \mathsf{O}(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D(\!|b|\!)_D\mathsf{R}\mathsf{S}a\mathsf{E}$$
$$\xrightarrow{rotI} \mathsf{O}(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D(\!|b|\!)_D\mathsf{L}(\!|a|\!)_C\mathsf{S}\mathsf{E} \xrightarrow{rotF} \mathsf{O}(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D\mathsf{L}(\!|a|\!)_C(\!|b|\!)_B\mathsf{S}\mathsf{E}$$
$$\xrightarrow{rotF} \mathsf{O}(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D\mathsf{L}(\!|a|\!)_C(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}\mathsf{E} \xrightarrow{rotF} \mathsf{O}(\!|a|\!)_E\mathsf{C}\mathsf{L}(\!|a|\!)_C(\!|a|\!)_B(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}\mathsf{E}$$
$$\xrightarrow{rotG} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}\mathsf{R}(\!|a|\!)_B(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}\mathsf{E} \xrightarrow{rotH} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D\mathsf{R}(\!|b|\!)_B(\!|b|\!)_B\mathsf{S}\mathsf{E}$$
$$\xrightarrow{rotH} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D\mathsf{R}(\!|b|\!)_B\mathsf{S}\mathsf{E} \xrightarrow{rotH} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D(\!|b|\!)_D\mathsf{R}\mathsf{S}\mathsf{E}$$
$$\xrightarrow{rotJ} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D(\!|b|\!)_D\mathsf{F}\mathsf{E} \xrightarrow{rotK} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D(\!|b|\!)_D\mathsf{F}b\mathsf{E} \xrightarrow{rotK} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}(\!|a|\!)_D\mathsf{F}bb\mathsf{E}$$
$$\xrightarrow{rotK} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{C}\mathsf{F}abb\mathsf{E} \xrightarrow{rotL} \mathsf{O}(\!|a|\!)_E(\!|a|\!)_E\mathsf{f}abb\mathsf{E} \xrightarrow{rotL} \mathsf{O}(\!|a|\!)_E\mathsf{f}aabb\mathsf{E} \xrightarrow{rotL} \mathsf{O}\mathsf{f}aaabb\mathsf{E}$$
$$\xrightarrow{rotN} \mathsf{W}aaabb\mathsf{E} \xrightarrow{rotO} \mathsf{B}ababab\mathsf{E}$$

Proposition 3.38. *If $u \sim v$ then $\mathsf{B}u\mathsf{E} \to^*_{\mathrm{rot}(\Sigma_A)} \mathsf{W}v\mathsf{E}$.*

Proof. If $u = \varepsilon$ then $v = \varepsilon$ and $\mathsf{BE} \to_{\mathrm{rot}(\Sigma_A)} \mathsf{WE}$. If $u = a_1 \dots a_n$ for some $n \geq 1$ and $v = a_k \dots a_n a_1 \dots a_{k-1}$ then there are two cases: If $k = 1$ (i.e. $u = v$) then the following

rewrite sequence for $\mathsf{B}u\mathsf{E}$ exists:

$$\mathsf{B}a_1\dots a_n\mathsf{E} \to_{\mathrm{rot}(\Sigma_A)} \mathsf{OC}\langle\!|a_1|\!\rangle_D\mathsf{G}a_2\dots a_n\mathsf{E} \to^*_{\mathrm{rot}(\Sigma_A)} \mathsf{OC}\langle\!|a_1|\!\rangle_D\dots\langle\!|a_n|\!\rangle_D\mathsf{GE}$$
$$\to_{\mathrm{rot}(\Sigma_A)} \mathsf{OC}\langle\!|a_1|\!\rangle_D\dots\langle\!|a_n|\!\rangle_D\mathsf{FE} \to^*_{\mathrm{rot}(\Sigma_A)} \mathsf{W}a_1\dots a_n\mathsf{E}$$

If $1 < k \leq n$, then the following rewrite sequence for $\mathsf{B}u\mathsf{E}$ exists:

$$\mathsf{B}a_1\dots a_n\mathsf{E} \to_{\mathrm{rot}(\Sigma_A)} \mathsf{OC}\langle\!|a_1|\!\rangle_D\mathsf{G}a_2\dots a_n\mathsf{E} \to^*_{\mathrm{rot}(\Sigma_A)} \mathsf{OC}\langle\!|a_1|\!\rangle_D\dots\langle\!|a_{k-1}|\!\rangle_D\mathsf{G}a_k\dots a_n\mathsf{E}$$
$$\to_{\mathrm{rot}(\Sigma_A)} \mathsf{OC}\langle\!|a_1|\!\rangle_D\dots\langle\!|a_{k-1}|\!\rangle_D\mathsf{L}\langle\!|a_k|\!\rangle_C\mathsf{S}a_{k+1}\dots a_n\mathsf{E} \to^*_{\mathrm{rot}(\Sigma_A)} \mathsf{O}\langle\!|a_k|\!\rangle_E\dots\langle\!|a_n|\!\rangle_E\mathsf{C}\langle\!|a_1|\!\rangle_D\dots\langle\!|a_{k-1}|\!\rangle_D\mathsf{RSE}$$
$$\to_{\mathrm{rot}(\Sigma_A)} \mathsf{O}\langle\!|a_k|\!\rangle_E\dots\langle\!|a_n|\!\rangle_E\mathsf{C}\langle\!|a_1|\!\rangle_D\dots\langle\!|a_{k-1}|\!\rangle_D\mathsf{FE} \to^*_{\mathrm{rot}(\Sigma_A)} \mathsf{W}a_k\dots a_n a_1\dots a_{k-1}\mathsf{E}$$

□

Theorem 3.39. *The transformation* rotate *is sound.*

Proof. Proposition 3.38 and the construction of $\mathsf{rotate}(R)$ show that whenever $u \sim v \hookrightarrow_R w$ then also $\mathsf{B}u\mathsf{E} \to^*_{\mathsf{rotate}(R)} \mathsf{B}w\mathsf{E}$. Now Proposition 3.35 implies soundness. □

3.4.1. *Completeness of Rotate.* We write $\widehat{\Sigma_A}$ for the extension of alphabet Σ_A by the fresh symbols B,E,W,R,G,O,C,L,S,F,f, and by four copies $\Sigma_B, \Sigma_C, \Sigma_D, \Sigma_E$ of the alphabet Σ_A.

For proving completeness of the transformation rotate, we use type introduction. Let $\mathbb{T} := \{\mathcal{K}, \mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{T}\}$ and let the symbols in $\widehat{\Sigma_A}$ be typed as follows:

$a : \mathcal{A} \to \mathcal{A}$ for all $a \in \Sigma_A$	$\mathsf{E} : \mathcal{K} \to \mathcal{A}$	$\mathsf{G}, \mathsf{F} : \mathcal{A} \to \mathcal{D}$
$b : \mathcal{B} \to \mathcal{B}$ for all $b \in \Sigma_B$	$\mathsf{S} : \mathcal{A} \to \mathcal{B}$	$\mathsf{B}, \mathsf{W} : \mathcal{A} \to \mathcal{T}$
$c : \mathcal{B} \to \mathcal{C}$ for all $c \in \Sigma_C$	$\mathsf{L} : \mathcal{C} \to \mathcal{D}$	$\mathsf{O} : \mathcal{E} \to \mathcal{T}$
$d : \mathcal{D} \to \mathcal{D}$ for all $d \in \Sigma_D$	$\mathsf{R} : \mathcal{B} \to \mathcal{D}$	$\mathsf{f} : \mathcal{A} \to \mathcal{E}$
$e : \mathcal{E} \to \mathcal{E}$ for all $e \in \Sigma_E$	$\mathsf{C} : \mathcal{D} \to \mathcal{E}$	

One can verify that with definition R is indeed a typed SRS: rule (rotA) rewrites strings of type $\mathcal{K} \to \mathcal{T}$, rules (rotB), (rotN), and (rotO) rewrite strings of type $\mathcal{A} \to \mathcal{T}$, rules (rotC), (rotD), (rotI), and (rotK) rewrite strings of type $\mathcal{A} \to \mathcal{D}$, rules (rotE) and (rotJ) rewrite strings of type $\mathcal{K} \to \mathcal{D}$, rules (rotF) and (rotH) rewrite strings of type $\mathcal{B} \to \mathcal{D}$, rule (rotG) rewrites strings of type $\mathcal{B} \to \mathcal{E}$, and rules (rotL) and (rotM) rewrite strings of type $\mathcal{A} \to \mathcal{E}$. Thus, by Corollary 3.4 type introduction can be used, i.e. $\mathsf{rotate}(R)$ is terminating if, and only if the typed system terminates (and analogously for relative termination, see Theorem 3.3).

Lemma 3.40. *Let $w \in \widehat{\Sigma_A}^*$ be a well-typed string, s.t. w admits an infinite reduction w.r.t. $\to_{\mathsf{rotate}(R)}$. Then there exists a well-typed string $w' \in \widehat{\Sigma_A}^*$ of type $\mathcal{K} \to \mathcal{T}$ which admits an infinite reduction w.r.t. $\to_{\mathsf{rotate}(R)}$.*

Proof. W.l.o.g. we can assume that $w \neq \varepsilon$ (since ε is irreducible w.r.t. $\to_{\mathsf{rotate}(R)}$). If w is not of type $\mathcal{K} \to \mathcal{T}$, then we can prepend and append symbols to w constructing a string uwv of type $\mathcal{K} \to \mathcal{T}$, since for any type $\tau \in \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{K}, \mathcal{T}\}$ there are the following sequences of type $\tau \to \mathcal{T}$ where $\tau \neq \mathcal{T}$ (used for u) and of type $\mathcal{K} \to \tau$ where $\tau \neq \mathcal{K}$ (used for v), where $c \in \Sigma_C$:

$\mathsf{B} : \mathcal{A} \to \mathcal{T}$	$\mathsf{OCL}c : \mathcal{B} \to \mathcal{T}$	$\mathsf{OCL} : \mathcal{C} \to \mathcal{T}$	$\mathsf{OC} : \mathcal{D} \to \mathcal{T}$	$\mathsf{O} : \mathcal{E} \to \mathcal{T}$	$\mathsf{BE} : \mathcal{K} \to \mathcal{T}$
$\mathsf{E} : \mathcal{K} \to \mathcal{A}$	$\mathsf{SE} : \mathcal{K} \to \mathcal{B}$	$c\mathsf{SE} : \mathcal{K} \to \mathcal{C}$	$\mathsf{L}c\mathsf{SE} : \mathcal{K} \to \mathcal{D}$	$\mathsf{fE} : \mathcal{K} \to \mathcal{E}$	

Clearly, the infinite reduction for w can be used to construct an infinite reduction for uwv. □

By inspecting the typing the following characterization of terms of type $\mathcal{T}$ holds:

Lemma 3.41. *Any well-typed string of type $\mathcal{K} \to \mathcal{T}$ is of one of the following forms:*
(1) $\mathsf{O}w_E\mathsf{C}w_D\mathsf{L}cw_B\mathsf{S}w_A\mathsf{E}$ *where* $w_E \in \Sigma_E^*$, $w_D \in \Sigma_D^*$, $c \in \Sigma_C$, $w_B \in \Sigma_B^*$, *and* $w_A \in \Sigma_A^*$.
(2) $\mathsf{O}w_E\mathsf{C}w_D\mathsf{R}w_B\mathsf{S}w_A\mathsf{E}$ *where* $w_E \in \Sigma_E^*$, $w_D \in \Sigma_D^*$, $w_B \in \Sigma_B^*$, *and* $w_A \in \Sigma_A^*$.
(3) $\mathsf{O}w_E\mathsf{C}w_D\mathsf{F}w_A\mathsf{E}$ *where* $w_E \in \Sigma_E^*$, $w_D \in \Sigma_D^*$, *and* $w_A \in \Sigma_A^*$.
(4) $\mathsf{O}w_E\mathsf{C}w_D\mathsf{G}w_A\mathsf{E}$ *where* $w_E \in \Sigma_E^*$, $w_D \in \Sigma_D^*$, *and* $w_A \in \Sigma_A^*$.
(5) $\mathsf{O}w_E\mathsf{f}w_A\mathsf{E}$ *where* $w_E \in \Sigma_E^*$ *and* $w_A \in \Sigma_A^*$.
(6) $\mathsf{B}w_A\mathsf{E}$ *where* $w_A \in \Sigma_A^*$.
(7) $\mathsf{W}w_A\mathsf{E}$ *where* $w_A \in \Sigma_A^*$.

Informally, the parts of strings in the lemma above can be describes as follows: for a string $w = w_1w_2$, w_E is a prefix of w_2 that has been rotated, while w_A is a suffix of w_2 that has to be rotated, and w_B, w_D are parts of w_1 and symbol c is the symbol that is currently processed.

Definition 3.42. For well-typed strings $w : \mathcal{K} \to \mathcal{T}$, the mapping $\Phi_R(w) \in \mathbb{P}(\Sigma_A^*)$ (where $\mathbb{P}$ denotes the power set) is defined according to the cases of Lemma 3.41 as follows:
(1) $\Phi_R(\mathsf{O}w_E\mathsf{C}w_D\mathsf{L}cw_B\mathsf{S}w_A\mathsf{E}) := \{(\!|w_D|\!)_A(\!|w_B|\!)_A(\!|w_E|\!)_A(\!|c|\!)_A w_A\}$
(2) $\Phi_R(\mathsf{O}w_E\mathsf{C}w_D\mathsf{R}w_B\mathsf{S}w_A\mathsf{E}) := \{(\!|w_D|\!)_A(\!|w_B|\!)_A(\!|w_E|\!)_A w_A\}$
(3) $\Phi_R(\mathsf{O}w_E\mathsf{C}w_D\mathsf{F}w_A\mathsf{E}) := \{(\!|w_D|\!)_A w_A(\!|w_E|\!)_A\}$
(4) $\Phi_R(\mathsf{O}w_E\mathsf{C}w_D\mathsf{G}w_A\mathsf{E}) := \{(\!|w_D|\!)_A w'_A(\!|w_E|\!)_A w''_A \mid \text{for all } w'_Aw''_A = w_A\}$
(5) $\Phi_R(\mathsf{O}w_E\mathsf{f}w_A\mathsf{E}) := \{(\!|w_E|\!)_A w_A\}$
(6) $\Phi_R(\mathsf{B}w_A\mathsf{E}) := \{w_A\}$
(7) $\Phi_R(\mathsf{W}w_A\mathsf{E}) := \{w_A\}$

Lemma 3.43. *Let $w, w' : \mathcal{K} \to \mathcal{T}$ with $w \to_{\mathrm{rot}(\Sigma_A)} w'$. Then for any $u' \in \Phi_R(w')$ there exists $u \in \Phi_R(w)$ s.t. $u \sim u'$.*

Proof. We go through the cases for w according to Lemma 3.41:
(1) If $w = \mathsf{O}w_E\mathsf{C}w_D\mathsf{L}cw_B\mathsf{S}w_A\mathsf{E}$, then rules (rotF) and (rotG) may be applied. If rule (rotF) is applied, then with $w_D = w'_Dd$ we have $w' = \mathsf{O}w_E\mathsf{C}w'_D\mathsf{L}c(\!|d|\!)_B w_B\mathsf{S}w_A\mathsf{E}$, and since $\Phi(w) = \Phi(w')$, the claim holds. If rule (rotG) is applied then $w_D = \varepsilon$, $w' = \mathsf{O}w_Ee\mathsf{C}\mathsf{R}w_B\mathsf{S}w_A\mathsf{E}$ and since $\Phi(w) = \Phi(w')$, the claim holds.
(2) If $w = \mathsf{O}w_E\mathsf{C}w_D\mathsf{R}w_B\mathsf{S}w_A\mathsf{E}$, then rules (rotH), (rotI), and (rotJ) may be applied. If rule (rotH) is applied then with $w_B = bw'_B$ we have $w' = \mathsf{O}w_E\mathsf{C}w_D(\!|b|\!)_D\mathsf{R}w'_B\mathsf{S}w_A\mathsf{E}$, and since $\Phi(w) = \Phi(w')$, the claim holds. If rule (rotI) is applied, then $w_B = \varepsilon$ and with $w_A = aw'_A$ we have $w' = \mathsf{O}w_E\mathsf{C}w_D\mathsf{L}(\!|a|\!)_C\mathsf{S}w'_A\mathsf{E}$, and since $\Phi(w) = \Phi(w')$, the claim holds. If rule (rotJ) is applied, then $w_A = w_B = \varepsilon$ and $w' = \mathsf{O}w_E\mathsf{C}w_D\mathsf{FE}$ and since $\Phi(w) = \Phi(w')$, the claim holds.
(3) If $w = \mathsf{O}w_E\mathsf{C}w_D\mathsf{F}w_A\mathsf{E}$, then rules (rotK) and (rotL) may be applied. If rule (rotK) is applied, then with $w_D = w'_Dd$ we have $w' = \mathsf{O}w_E\mathsf{C}w'_D\mathsf{F}(\!|d|\!)_A w_A\mathsf{E}$, and since $\Phi(w) = \Phi(w')$, the claim holds. If rule (rotL) is applied, then $w_D = \varepsilon$, $w' = \mathsf{O}w_E\mathsf{f}w_A\mathsf{E}$, and since $\Phi(w) = \{w_A(\!|w_E|\!)_A\}$, $\Phi(w') = \{(\!|w_E|\!)_A w_A\}$, and $w_A(\!|w_E|\!)_A \sim (\!|w_E|\!)_A w_A$, the claim holds.
(4) If $w = \mathsf{O}w_E\mathsf{C}w_D\mathsf{G}w_A\mathsf{E}$, then rules (rotC), (rotD), and (rotE) may be applied. If rule (rotC) is applied, then with $w_A = av_A$ we have $w' = \mathsf{O}w_E\mathsf{C}w_D(\!|a|\!)_D\mathsf{G}v_A\mathsf{E}$, and since $\Phi(w) \supseteq \Phi(w') \neq \emptyset$, the claim holds. If rule (rotD) is applied, then with $w_A = av_A$ we

have $w' = \mathsf{O}w_E\mathsf{C}w_D\mathsf{L}(\!|a|\!)_C\mathsf{S}v_A\mathsf{E}$, and since $\Phi(w) \supseteq \Phi(w') \neq \emptyset$, the claim holds. If rule (rotE) is applied, then $w_A = \varepsilon$ and $w' = \mathsf{O}w_E\mathsf{C}w_D\mathsf{F}\mathsf{E}$, and since $\Phi(w) = \Phi(w')$, the claim holds.

(5) If $w = \mathsf{O}w_E\mathsf{f}w_A\mathsf{E}$, then rules (rotM) and (rotN) may be applied: If rule (rotM) is applied, then with $w_E = w'_E e$ we have $w' = \mathsf{O}w'_E\mathsf{f}(\!|e|\!)_A w_A\mathsf{E}$, and since $\Phi(w) = \Phi(w')$, the claim holds. If rule (rotN) is applied then $w_E = \varepsilon$, $w' = \mathsf{W}w_A\mathsf{E}$, and since $\Phi(w) = \Phi(w')$, the claim holds.

(6) If $w = \mathsf{B}w_A\mathsf{E}$, then rules (rotA) or (rotB) may be applied: If rule (rotA) is applied then $\Phi(w) = \{\varepsilon\} = \Phi(w')$ and thus the claim holds. If rule (rotB) is applied, then with $w_A = aw'_A$ we have $w' = \mathsf{OC}(\!|a|\!)_D\mathsf{G}w'_A$, and since $\Phi(w) = \Phi(w')$ the claim holds.

(7) If $w = \mathsf{W}w_A\mathsf{E}$. then no rule is applicable. □

Lemma 3.44. *The SRS* $\mathrm{rot}(\Sigma_A)$ *is terminating.*

Proof. Let σ be the following polynomial interpretation:

$$\begin{array}{llll} \sigma(\mathsf{W}) = \lambda x.x & \sigma(\mathsf{R}) = \lambda x.2x & \sigma(\mathsf{F}) = \lambda x.2x+2 & \sigma(\mathsf{f}) = \lambda x.2x+2 \\ \sigma(\mathsf{E}) = \lambda x.x+1 & \sigma(\mathsf{L}) = \lambda x.2x+1 & \sigma(\mathsf{S}) = \lambda x.3x & \sigma(\mathsf{O}) = \lambda x.x+9 \\ \sigma(\mathsf{C}) = \lambda x.2x & \sigma(\mathsf{G}) = \lambda x.6x+1 & \sigma(\mathsf{B}) = \lambda x.12x+18 & \\ \sigma(h) = \lambda x.4x+1 \text{ for all } h \in (\Sigma_A \cup \Sigma_B \cup \Sigma_C \cup \Sigma_D \cup \Sigma_E) & & & \end{array}$$

We verify that for every rule $(\ell \to r) \in \mathrm{rot}(\Sigma_A)$ the inequation $\sigma(\ell) > \sigma(r)$ holds: For rule (rotA), we have $\lambda x.12x + 30 > \lambda x.x + 1$, for rule (rotB), we have $\lambda x.48x + 30 > \lambda x.48x + 19$, for rule (rotC), we have $\lambda x.24x + 7 > \lambda x.24x + 5$, for rule (rotD), we have $\lambda x.24x+7 > \lambda x.24x+3$, for rule (rotE), we have $\lambda x.6x+7 > \lambda x.2x+4$, for rule (rotF), we have $\lambda x.32x+13 > \lambda x.32x+11$, for rule (rotG), we have $\lambda x.16x+6 > \lambda x.16x+1$, for rule (rotH), we have $\lambda x.8x+2 > \lambda x.8x+1$, for rule (rotI), we have $\lambda x.24x+6 > \lambda x.24x+3$, for rule (rotJ), we have $\lambda x.6x+6 > \lambda x.2x+4$, for rule (rotK), we have $\lambda x.8x+9 > \lambda x.8x+4$, for rule (rotL), we have $\lambda x.4x+4 > \lambda x.2x+2$, for rule (rotM), we have $\lambda x.8x+9 > \lambda x.8x+4$, and for rule (rotN), we have $\lambda x.2x + 11 > \lambda x.x$. □

Proposition 3.45. *If* $w : \mathcal{K} \to \mathcal{T}$ *admits an infinite* $\to_{\mathsf{rotate}(R)}$*-reduction, then there exists* $u \in \Phi_R(w)$ *s.t.* $[u]$ *admits an infinite* $\circ\!\!\to_R$*-reduction.*

Proof. Let $w \to_{\mathsf{rotate}(R)} w_1 \to_{\mathsf{rotate}(R)} w_2 \to_{\mathsf{rotate}(R)} \dots$ be an infinite $\to_{\mathsf{rotate}(R)}$-reduction. This sequence must have infinitely many steps using rules (rotO), since the rewrite system $\mathrm{rot}(\Sigma_A)$ is terminating (Lemma 3.44), i.e.

$$w = v_0 \to^*_{\mathrm{rot}(\Sigma_A)} v_1 \to_{(\mathrm{rotO})} v'_1 \to^*_{\mathrm{rot}(\Sigma_A)} v_2 \to_{(\mathrm{rotO})} v'_2 \to^*_{\mathrm{rot}(\Sigma_A)} \cdots$$

We consider all sub-sequences $v_i \to_{(\mathrm{rotO})} v'_i \to^*_{\mathrm{rot}(\Sigma_A)} v_{i+1}$ for $i = 1, 2, \dots$. Typing of all strings, Lemma 3.43 and the definition of the rules (rotO) imply that the steps $v_i \to_{(\mathrm{rotO})} v'_i$ must be of the form $\mathsf{W}\ell w_A\mathsf{E} \to \mathsf{B}r w_A\mathsf{E}$ where $(\ell \to r) \in R$. Since $\Phi_R(v_i) = \{\ell w_A\}$ and $\Phi_R(v'_i) = \{r w_A\}$, there exist (unique) strings $u_i \in \Phi_R(v_i)$ and $u'_i \in \Phi_R(v'_i)$ (namely $u_i = \ell w_A$ and $u'_i = r w_A$) s.t. $u_i \to_R u'_i$. Lemma 3.43 shows that for any sub-sequence $v'_i \to^*_{\mathrm{rot}(\Sigma_A)} v_{i+1}$ and any $u_{i+1} \in \Phi_R(v_{i+1})$ there exists $u''_i \in \Phi_R(v'_i)$ with $u''_i \sim u_{i+1}$. Since $\Phi_R(v'_i) = \{r w_A\} = \{u'_i\}$, the equality $u''_i = u'_i$ must hold. Thus, we have $u_i \to_R u'_i \sim u_{i+1}$ where $\Phi_R(v_i) = \{u_i\}$ and $\Phi_R(v_{i+1}) = \{u_{i+1}\}$. Since this holds for all $i = 1, 2, \dots$, and since $\Phi_R(v_j)$ is a singleton for all v_j, we can construct the infinite sequence $u_1 \to_R . \sim u_2 \to_R . \sim u_3 \to_R . \sim \cdots$. This implies $[u_1] \circ\!\!\to_R [u_2] \circ\!\!\to_R [u_3] \dots$ and thus $\circ\!\!\to_R$ is non-terminating. □

Theorem 3.46. *The transformation* rotate *is sound and complete.*

Proof. Soundness is shown in Theorem 3.39, completeness follows by type introduction (Corollary 3.4), Lemma 3.40, and Proposition 3.45. □

3.4.2. *Relative Termination.* We provide a variant of the transformation rotate for relative termination:

Definition 3.47. Let $S \subseteq R$ be SRSs over an alphabet Σ_A. The transformation rotate_{rel} is defined as:

$$\mathsf{rotate}_{rel}(S,R) \;:=\; (\{\mathsf{W}\ell \to \mathsf{B}r \mid (\ell \to r) \in S\}, \mathsf{rotate}(R))$$

We show soundness and completeness:

Theorem 3.48. *The transformation* rotate_{rel} *is sound and complete for relative termination.*

Proof. Let $S \subseteq R$ and $\mathsf{rotate}_{rel}(S,R) = (S',R')$. Soundness of the transformation rotate implies (see proof of Theorem 3.46) that

$$[u] \circ\!\!\to_R [v] \text{ implies } \mathsf{B}u\mathsf{E} \to^*_{R'} \mathsf{B}w\mathsf{E} \tag{3.1}$$

Since $\mathsf{rotate}(S) \subseteq R'$, this also shows:

$$[u] \circ\!\!\to_S [v] \text{ implies } \mathsf{B}u\mathsf{E} \to^*_{R'} . \to_{S'} \mathsf{B}w\mathsf{E}. \tag{3.2}$$

For proving soundness of rotate_{rel}, let us assume that S is not cycle terminating relative to R. Then there exists an infinite derivation $[w_1] \circ\!\!\to^*_R . \circ\!\!\to_S [w_2] \circ\!\!\to^*_R . \circ\!\!\to_S \cdots$. Applying the implications (3.1) and (3.2) shows that $\mathsf{B}w_i\mathsf{E} \to^*_{R'} . \to^*_{R'} . \to_{S'} \mathsf{B}w_{i+1}\mathsf{E}$ for $i = 1,2,\ldots$. This shows that $\mathsf{B}w_i\mathsf{E} \to^*_{R'} . \to_{S'} \mathsf{B}w_{i+1}\mathsf{E}$ for $i = 1,2,\ldots$ and thus S' is not string terminating relative to R'.

For proving completeness of rotate_{rel}, we again use the types introduced in Section 3.4.1. Let us assume that S' is not string terminating relative to R'. By Theorem 3.3 and Lemma 3.40 there exists a well-typed string $w_0 : \mathcal{K} \to \mathcal{T}$ s.t. for all $i = 0,1,2\ldots$ the derivation $w_i \to_{S'} . \to^*_{R'} w_{i+1}$ exists. Each such derivation can be written as

$$\begin{aligned} w_i &\to^*_{\mathrm{rot}(\Sigma_A)} w_{0,0,i} \to_{rotO(S)} w_{0,1,i} \\ &\to^*_{\mathrm{rot}(\Sigma_A)} w_{1,0,i} \to_{rotO(R)} w_{1,1,i} \\ &\cdots \\ &\to^*_{\mathrm{rot}(\Sigma_A)} w_{m_i,0,i} \to_{rotO(R)} w_{m_i,1,i} = w_{i+1} \end{aligned}$$

where $m_i \geq 0$. Starting with $w_{0,0,i}$ and appending the sequence $w_{m_i,1,i} \to^*_{\mathrm{rot}(\Sigma_A)} w_{0,0,i+1}$ this can be formulated as follows: There exists a derivation

$$\begin{aligned} w_{0,0,i} &\to_{rotO(S)} w_{0,1,i} \\ &\to^*_{\mathrm{rot}(\Sigma_A)} w_{1,0,i} \to_{rotO(R)} w_{1,1,i} \\ &\cdots \\ &\to^*_{\mathrm{rot}(\Sigma_A)} w_{m_i,0,i} \to_{rotO(R)} w_{m_i,1,i} \to^*_{\mathrm{rot}(\Sigma_A)} w_{0,0,i+1} \end{aligned}$$

for all $i = 0,1,\ldots$ where $m_i \geq 0$. Typing of the strings and the cases in Lemma 3.41 show that $\Phi_R(w_{k,l,i})$ is a singleton $\{u_{k,l,i}\}$ for all i and $k = 0,\ldots,m_i$, $l = 0,1$. Due to

applicability of rule (rotO), we also have $u_{0,0,i} \to_S u_{0,1,i}$ and $u_{k,0,i} \to_R u_{k,1,i}$ for $k = 1, \ldots, m_i$. Lemma 3.43 implies that $u_{k,0,i} \sim u_{k-1,1,i}$ for $k = 1, \ldots, m_i + 1$. Thus, we have

$$u_{0,0,i} \to_S u_{0,1,i} \sim u_{1,0,i} \to_R u_{1,1,i} \sim \cdot \to_R \cdots \sim \cdot \to_R u_{m_i,1,i} \sim u_{0,0,i+1}$$

which shows $[u_{0,0,i}] \circ\!\!\to_S [u_{1,0,i}] \circ\!\!\to_R^* [u_{0,0,i+1}]$. Since this holds for all $i = 0, 1, \ldots$, we have shown that S is not cycle terminating relative to R. □

4. Trace Decreasing Matrix Interpretations

In this section we present a variant of matrix interpretations suitable for proving cycle termination. The basics of matrix interpretations for string and term rewriting were presented in [14, 15, 7, 16]. The special case of tropical and arctic matrix interpretations for cycle rewriting was presented in [28], in the setting of *type graphs*. Natural matrix interpretations for cycle rewriting were presented in [18], inspired by an approach proposed by Johannes Waldmann. Here we present a self-contained uniform framework covering all these cases as we show by subsequently instantiating the framework for three semi-rings. At the end of the section we discuss and illustrate the approach by examples and also discuss some limitations.

4.1. **A Uniform Framework for Matrix Interpretations.** Fix a commutative *semi-ring*, that is, a set X, a zero element $0 \in X$, a unit element $1 \in X$, and two operations $+, \times$ that are commutative and associative, and satisfying

$$x + 0 = x,\ x \times 1 = x,\ x \times 0 = 0,\ x \times (y + z) = (x \times y) + (x \times z)$$

for all $x, y, z \in X$. Further we assume a well-founded order $>$ on X.

Next we fix a dimension $d > 0$ and choose M to be a set of $d \times d$ matrices over X. Multiplication of matrices is defined as usual, now using $+$ and $\times$ from the semi-ring as basic operations on elements:

$$(AB)_{ij} = \sum_{k=1}^{d} A_{ik} \times B_{kj}$$

for all $i, j = 1, \ldots, d$, where $\sum$ extends the operation $+$. Note that in the following we only require matrix multiplication and no addition. Thus we assume that M is closed under multiplication and contains the identity matrix.

For a $d \times d$ matrix A over X, its *trace* $\mathsf{tr}(A)$ is defined to be $\sum_{i=1}^{d} A_{ii}$: the sum of its diagonal.

On M we assume relations $>, \geq$ satisfying

$$A > B \implies (AC > BC \wedge CA > CB) \tag{4.1}$$

$$A \geq B \implies (AC \geq BC \wedge CA \geq CB) \tag{4.2}$$

$$A > B \implies \mathsf{tr}(A) > \mathsf{tr}(B) \tag{4.3}$$

$$A \geq B \implies \mathsf{tr}(A) \geq \mathsf{tr}(B) \tag{4.4}$$

for all $A, B, C \in M$. Note the overloading of $>$ and $\geq$: when applied on matrices it refers to the relations $>$, $\geq$ we assume on M, when applied to elements of X it refers to the well-founded order on X, where $x \geq y \iff x > y \vee x = y$.

Note that on the one hand these requirements imply that $>$ is irreflexive by $A > B \implies \mathsf{tr}(A) > \mathsf{tr}(B)$, and on the other hand the requirements imply that M does not consist of all $d \times d$ matrices over X: for $A > B$ and C being the matrix 0 having 0 on all positions, we have $AC = 0 = BC$, violating $A > B \implies AC > BC$.

A *matrix interpretation* $\langle\cdot\rangle$ for a signature Σ is defined to be a mapping from Σ to M. It is extended to $\langle\cdot\rangle : \Sigma^* \to M$ by defining inductively $\langle\varepsilon\rangle = I$ and $\langle ua\rangle = \langle u\rangle \times \langle a\rangle$ for all $u \in \Sigma^*$, $a \in \Sigma$, where I is the identity matrix.

Theorem 4.1. *Let $S \subseteq R$ be SRSs over Σ and let $\langle\cdot\rangle : \Sigma \to M$ satisfy the above properties and*

- *$\langle\ell\rangle > \langle r\rangle$ for all $(\ell \to r) \in S$, and*
- *$\langle\ell\rangle \geq \langle r\rangle$ for all $(\ell \to r) \in R \setminus S$*

Then $\circ\!\!\to_S$ is terminating relative to $\circ\!\!\to_R$.

Proof. If $u \sim v$ then $u = u_1u_2$ and $v = u_2u_1$ for some u_1, u_2. Write $A = \langle u_1\rangle$ and $B = \langle u_2\rangle$. Then

$$\mathsf{tr}(\langle u\rangle) = \mathsf{tr}(AB) = \sum_{i=1}^{d}\sum_{k=1}^{d} A_{i,k} \times B_{k,i} = \mathsf{tr}(BA) = \mathsf{tr}(\langle v\rangle).$$

If $u \to_{R\setminus S} v$ then there is a rule $(\ell \to r) \in R \setminus S$ and $x, y \in \Sigma^*$ such that $u = x\ell y$ and $v = xry$, so

$$\langle u\rangle = \langle x\rangle\langle\ell\rangle\langle y\rangle \geq \langle x\rangle\langle r\rangle\langle y\rangle = \langle v\rangle$$

using $A \geq B \implies (AC \geq BC \wedge CA \geq CB)$, hence $\mathsf{tr}(\langle u\rangle) \geq \mathsf{tr}(\langle v\rangle)$.

If $u \to_S v$ then there is a rule $(\ell \to r) \in S$ and $x, y \in \Sigma^*$ such that $u = x\ell y$ and $v = xry$, so

$$\langle u\rangle = \langle x\rangle\langle\ell\rangle\langle y\rangle > \langle x\rangle\langle r\rangle\langle y\rangle = \langle v\rangle$$

using $A > B \implies (AC > BC \wedge CA > CB)$, hence $\mathsf{tr}(\langle u\rangle) > \mathsf{tr}(\langle v\rangle)$.

Assume $\circ\!\!\to_S$ is not terminating relative to $\circ\!\!\to_R$, then there are $u_i, v_i \in \Sigma^*$ for $i \in \mathbb{N}$ such that

$$u_1 \to_R v_1 \sim u_2 \to_R v_2 \sim u_3 \to_R v_3 \sim \cdots,$$

containing infinitely many steps $u_i \to_S v_i$. By the above observations we have $\mathsf{tr}(v_i) = \mathsf{tr}(u_{i+1})$ for all i, $\mathsf{tr}(\langle u_i\rangle) > \mathsf{tr}(\langle v_i\rangle)$ for infinitely many i, and $\mathsf{tr}(\langle u_i\rangle) \geq \mathsf{tr}(\langle v_i\rangle)$ for all other i, yielding an infinite strictly decreasing sequence in X, contradicting well-foundedness of the order $>$ on X. □

A typical way to apply Theorem 4.1 to prove cycle termination of any SRS R is as follows: choose an instance of a semi-ring and interpretations $\langle a\rangle \in M$ for every $a \in \Sigma$, and for all rules $\ell \to r$ we either have $\langle\ell\rangle > \langle r\rangle$ or $\langle\ell\rangle \geq \langle r\rangle$. If $\langle\ell\rangle > \langle r\rangle$ holds for all rules we are done, otherwise the remaining proof obligation is to prove cycle termination of the rules for which $\langle\ell\rangle \geq \langle r\rangle$.

Although the proof of Theorem 4.1 is not very hard, it is quite subtle: while in the final argument $\mathsf{tr}(\langle\cdot\rangle)$ is applied on the strings, it is essential to require $\langle\ell\rangle > \langle r\rangle$ and not the weaker requirement $\mathsf{tr}(\langle\ell\rangle) > \mathsf{tr}(\langle r\rangle)$. Surprisingly, the proof does not need further requirements on the relation $>$ and $\geq$ like transitivity or $> \subseteq \geq$.

In order to prove relative termination, we give the following extension of Theorem 4.1.

Theorem 4.2. *Let $S' \subseteq S$, $R' \subseteq R$, $S' \subseteq R'$, and $S \subseteq R$ be SRSs over Σ and let $\langle\cdot\rangle : \Sigma \to M$ satisfy the above properties and*

- $\circ\!\!\to_{S'}$ *is terminating relative to* $\circ\!\!\to_{R'}$*, and*
- $\langle \ell \rangle \geq \langle r \rangle$ *for all* $(\ell \to r) \in ((R' \setminus S) \cup S')$
- $\langle \ell \rangle > \langle r \rangle$ *for all* $(\ell \to r) \in (R \setminus ((R' \setminus S) \cup S'))$

Then $\circ\!\!\to_S$ *is terminating relative to* $\circ\!\!\to_R$*.*

Proof. Assume $[u_1] \circ\!\!\to_R [u_2] \circ\!\!\to_R [u_3] \circ\!\!\to_R \cdots$; we have to prove that it contains finitely many $\circ\!\!\to_S$-steps. As in the proof of Theorem 4.1 we have $\mathsf{tr}(\langle u_i \rangle) > \mathsf{tr}(\langle u_{i+1} \rangle)$ for applications of rules $(\ell \to r) \in (R \setminus ((R' \setminus S) \cup S'))$ and $\mathsf{tr}(\langle u_i \rangle) \geq \mathsf{tr}(\langle u_{i+1} \rangle)$ for applications of remaining rules. Since $>$ is well-founded on X, there are only finitely many steps of the former type, so after a finite initial part the infinite reduction only consists of $((R' \setminus S) \cup S')$-steps. But then applying that $\circ\!\!\to_{S'}$ is terminating relative to $\circ\!\!\to_{R'}$, these remaining $((R' \setminus S) \cup S')$-steps only contain finitely many S'-steps. All other steps are $(R' \setminus S)$-steps and thus do not contain S-steps. Hence in total there are only finitely many $\circ\!\!\to_S$-steps. □

Indeed Theorem 4.1 is an instance of Theorem 4.2 if $S' = \emptyset$ and $R' = R \setminus S$. A typical way to apply Theorem 4.2 to prove cycle termination of any SRS S relative to R is very similar to the typical way to apply Theorem 4.1: remove the rules for which '$>$' is obtained, both from S and R.

In order to apply Theorem 4.1 or Theorem 4.2 we need an instance of a semi-ring X, a set M of matrices over X and relations $>, \geq$ on M such that all assumed properties holds. We give three such instances: the tropical, arctic and natural matrix interpretations, all depending on dimension d. For all these three instances the search for applying Theorem 4.1 has been implemented in our tool `torpacyc` by transforming the requirements to SMT format and calling the external SMT solver `Yices` [5, 25]. The same has been done in our tool `tdmi`, also covering Theorem 4.2.

A nice point is that we do not need to try separately for which rules we require $\langle \ell \rangle \geq \langle r \rangle$ and for which rules $\langle \ell \rangle > \langle r \rangle$, but we just specify in the SMT formula that the latter occurs at least once and the former for all other rules.

4.2. **Natural Matrix Interpretations.** In natural matrix interpretations we have $X = \mathbb{N}$, and $0, 1, +, \times, >$ have their usual meaning. We define M to consist of all $d \times d$ matrices A satisfying $A_{11} > 0$. On M we define the relations $>$ and $\geq$ by

$$A > B \iff A_{11} > B_{11} \wedge \forall i, j : A_{ij} \geq B_{ij}, \qquad A \geq B \iff \forall i, j : A_{ij} \geq B_{ij}.$$

For M with these relations the required properties (4.1), (4.2), (4.3), and (4.4) are all easily checked. Hence this yields a way to prove (relative) cycle termination by Theorem 4.1. As an example we consider the single rule $aa \to aba$ and choose

$$\langle a \rangle = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \ \langle b \rangle = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

yielding

$$\langle aa \rangle = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} > \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \langle aba \rangle,$$

proving cycle termination by Theorem 4.1.

The original versions of matrix interpretations in [14, 7] are not suitable for proving cycle termination since they succeed in proving termination of $ab \to ba$ for which cycle termination does not hold. Even more, the same holds for the original killer example for

the method of matrix interpretations was $aa \to bc,\ bb \to ac,\ cc \to ab$, for which cycle termination does not hold due to $[ccaa] \circ\!\!\to [abaa] \circ\!\!\to [abbc] \circ\!\!\to [aacc]$.

Main differences are our conditions (4.3) and (4.4) on the trace of the matrices and that in our setting the interpretation of symbols is multiplication by a matrix, while in [14, 7] it combines such a matrix multiplication by adding a vector.

4.3. **Tropical Matrix Interpretations.** In tropical matrix interpretations we choose the semi-ring $X = \mathbb{N} \cup \{\infty\}$, with min being the semi-ring addition and the normal addition being the semi-ring multiplication, both extended to X by defining

$$\min(\infty, x) = \min(x, \infty) = x \ \text{ and } \ \infty + x = x + \infty = \infty$$

for all $x \in \mathbb{N} \cup \{\infty\}$. Now ∞ acts as the semi-ring zero and 0 acts as the semi-ring unit; it is easily checked that all semi-ring requirements hold. This semi-ring is called the *tropical semi-ring* after its study by the Brazilian mathematician Imre Simon [19]. Over this semi-ring multiplication of matrices becomes

$$(AB)_{i,j} = \min(\{A_{i,k} + B_{k,j} \mid k = 1, \ldots, d\}),$$

so being quite different from the usual matrix operations. On this semi-ring we define the well-founded order $>$ to be the extension on $>$ on $\mathbb{N}$ defined by

$$x > y \iff (x, y \in \mathbb{N} \wedge x > y) \vee (x = \infty \wedge y \in \mathbb{N}).$$

So in this semi-ring the zero element is not the smallest but the largest element.

We define M to consist of all $d \times d$ matrices A satisfying $A_{11} \neq \infty$. On M we define the relation $>$ by

$$A > B \iff \forall i, j : (A_{ij} > B_{ij} \vee A_{ij} = B_{ij} = \infty),$$

and the relation $\geq$ by

$$A \geq B \iff \forall i, j : (A_{ij} > B_{ij} \vee A_{ij} = B_{ij}).$$

For M with these relations the required properties (4.1), (4.2), (4.3), and (4.4) are all easily checked; note that for every $A \in M$ we have $\mathsf{tr}(A) = \min_i A_{ii} \neq \infty$ since $A_{11} \neq \infty$. For these requirements it is essential that we defined $A > B$ by $A_{ij} > B_{ij}$ on all positions and not only at 1,1, since from $a > b$ we cannot conclude $\min(a, c) > \min(b, c)$, but from $a > b \wedge c > d$ we can conclude $\min(a, c) > \min(b, d)$.

As an example we again consider the single rule $aa \to aba$ and choose

$$\langle a \rangle = \begin{pmatrix} 1 & \infty \\ 0 & 1 \end{pmatrix},\ \langle b \rangle = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix},$$

by using the tropical matrix multiplication yielding

$$\langle aa \rangle = \begin{pmatrix} 1 & \infty \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \infty \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & \infty \\ 1 & 2 \end{pmatrix} > \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \infty \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & \infty \\ 0 & 1 \end{pmatrix} = \langle aba \rangle,$$

proving cycle termination by Theorem 4.1.

4.4. **Arctic Matrix Interpretations.** In arctic matrix interpretations we choose the semi-ring $X = \mathbb{N} \cup \{-\infty\}$, with max being the semi-ring addition and the normal addition being the semi-ring multiplication, both extended to X by defining

$$\max(-\infty, x) = \max(x, -\infty) = x \quad \text{and} \quad -\infty + x = x + -\infty = -\infty$$

for all $x \in \mathbb{N} \cup \{-\infty\}$. Now $-\infty$ acts as the semi-ring zero and 0 acts as the semi-ring unit; it is easily checked that all semi-ring requirements hold. This semi-ring is called the *arctic semi-ring* as it is a kind of opposite to the tropical semi-ring. Over this semi-ring multiplication of matrices becomes

$$(AB)_{i,j} = \max(\{A_{i,k} + B_{k,j} \mid k = 1, \ldots, d\}),$$

On this semi-ring we define the well-founded order $>$ to be the extension on $>$ on $\mathbb{N}$:

$$x > y \iff (x, y \in \mathbb{N} \wedge x > y) \vee (x \in \mathbb{N} \wedge y = -\infty).$$

So in this semi-ring the zero element is the smallest element, just like in the natural semi-ring.

We define M to consist of all $d \times d$ matrices A satisfying $A_{11} \neq -\infty$. On M we define the relation $>$ by

$$A > B \iff \forall i, j : (A_{ij} > B_{ij} \vee A_{ij} = B_{ij} = -\infty),$$

and the relation $\geq$ by

$$A \geq B \iff \forall i, j : (A_{ij} > B_{ij} \vee A_{ij} = B_{ij}).$$

For M with these relations all required properties hold, again providing a method for proving (relative) cycle termination by Theorem 4.1. Cycle termination of our example $aa \to aba$ can also be proved by arctic matrix interpretation, now by choosing

$$\langle a \rangle = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \ \langle b \rangle = \begin{pmatrix} 0 & -\infty \\ -\infty & -\infty \end{pmatrix}.$$

4.5. **Examples and Bounds on Reduction Lengths.** As an example of combining various versions of matrix interpretations we consider the system from the introduction:

$$0P \to 1P,\ 1P \to cP,\ 0c \to 10,\ 1c \to c0,\ P0 \to P100$$

for which the following proof is found fully automatically by `torpacyc`.

First the following tropical matrix interpretation is found:

$$\langle P \rangle = \begin{pmatrix} 0 & \infty \\ 0 & \infty \end{pmatrix}, \ \langle 0 \rangle = \begin{pmatrix} 2 & 2 \\ 0 & 0 \end{pmatrix}, \ \langle 1 \rangle = \begin{pmatrix} 2 & 1 \\ \infty & 0 \end{pmatrix}, \ \langle c \rangle = \begin{pmatrix} 1 & 0 \\ \infty & 0 \end{pmatrix}.$$

In this interpretation for the first four rules we have $\langle \ell \rangle \geq \langle r \rangle$ and for the last rule we have $\langle \ell \rangle > \langle r \rangle$. So by Theorem 4.1 the last rule may be removed, and `torpacyc` continues with the remaining four rules.

Next the following natural matrix interpretation is found:

$$\langle P \rangle = \begin{pmatrix} 1 & 0 \\ 2 & 0 \end{pmatrix}, \ \langle 0 \rangle = \begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix}, \ \langle 1 \rangle = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}, \ \langle c \rangle = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

For these interpretations we obtain $\langle 0P \rangle > \langle 1P \rangle$, $\langle 1P \rangle > \langle cP \rangle$, $\langle 0c \rangle \geq \langle 10 \rangle$ and $\langle 1c \rangle \geq \langle c0 \rangle$, hence by Theorem 4.1 it suffices to prove cycle termination of $0c \to 10,\ 1c \to c0$, for which `torpacyc` finds a simple counting argument. These simple counting arguments can be

seen as the instance of tropical matrix interpretations of dimension $d = 1$, not using ∞: interpret every $a \in \Sigma$ by a natural number, and then $\langle u \rangle$ is the sum of the interpretations of all symbols in u.

The method for proving cycle termination induced by Theorem 4.1 has similar limitations as the method of matrix interpretations in [15] for string termination: Since the entries of a product of n matrices are bounded by an exponential function in n, the method cannot prove cycle termination of systems which allow reduction sequences where every rewrite rule is applied more often than exponentially often.

Example 4.3. The rewrite system $R_1 := \{ab \to bca, cb \to bbc\}$ allows for string derivations of a length which is a tower of exponentials (see [15]), i.e. the string $a^k b^k$ has such a long derivation, since the derivation $ab^n \to^*_{R_1} b^{2^n-1}c^n a$ exists and this can be iterated for every a in a^k. Moreover, the number of applications of the first and of the second rule of R_1 is a tower of exponentials. This shows that the matrix interpretations in [15] are unable to prove string termination of R_1. The system $\phi(R_1) := \{RE \to LE, aL \to La', bL \to Lb', cL \to Lc', Ra' \to aR, Rb' \to bR, Rc' \to cR, abL \to bcaR, cbL \to bbcR\}$ uses the transformation ϕ from [28] and transforms the string rewrite system R_1 into a cycle rewrite system s.t. R_1 is string terminating iff $\phi(R_1)$ is cycle terminating. One can verify that $[ab^n LE] \circ\!\!\to^*_{\phi(R_1)} [b^{2^n-1}c^n aLE]$ which can also be iterated s.t. $[a^k b^k LE]$ has a cycle rewriting sequence whose length is a tower of k exponentials. Inspecting all nine rules of $\phi(R_1)$, the number of applications of any of the rules in this rewrite sequence is also a tower of k exponentials and thus it is impossible to prove cycle termination of $\phi(R_1)$ using Theorem 4.1. Consequently, our tool `torpacyc` does not find a termination proof for $\phi(R_1)$.

Remark 4.4. As expected our tool `torpacyc` does not find a termination proof for $\phi(R_1)$ from Example 4.3. On the other hand, with our transformational approach cycle termination can be proved: AProVE proves string termination of split($\phi(R_1)$).

A further question is whether matrix interpretations are limited to cycle rewrite systems with exponential derivation lengths only. The following example shows that this is not true:

Example 4.5. The SRS $R_2 := \{ab \to baa, cb \to bbc\}$ (see [15]) has derivations of doubly exponential length (since $ac^k b \to^*_{R_2} b^{2^k} a^{2^{2^k}} c^k$ and any rewrite step adds one symbol), but its string termination can be proved by relative termination and matrix interpretations by first removing the rule $cb \to bbc$ and then removing the other rule. This is possible, since the second rule is applied only exponentially often. For cycle rewriting the encoding ϕ from [28] is $\phi(R_2) = \{RE \to LE, aL \to La', bL \to Lb', cL \to Lc', Ra' \to aR, Rb' \to bR, Rc' \to cR, abL \to baaR, cbL \to bbcR\}$ and $\phi(R_2)$ is cycle terminating iff R_2 is string terminating. The system $\phi(R_2)$ also has doubly exponential cycle derivations, e.g. $[ac^k bLE] \circ\!\!\to^*_{\phi(R_2)} [b^{2^k} a^{2^{2^k}} c^k LE]$. However, `torpacyc` proves cycle termination of $\phi(R_2)$ by first removing the last rule using the matrix interpretation

$$\langle R \rangle = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}, \langle E \rangle = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}, \langle L \rangle = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}, \langle a \rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \langle a' \rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$\langle b \rangle = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \langle b' \rangle = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \langle c \rangle = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}, \langle c' \rangle = \begin{pmatrix} 1 & 0 \\ 1 & 3 \end{pmatrix}.$$

Thereafter the remaining rules (which now only have derivations of exponential length) are eliminated by matrix interpretations and counting arguments.

Note that the methods in [28] are not able to prove cycle termination of $\phi(R_2)$, since they can only remove rules which are applied polynomially often in any derivation.

Also the transformational approach successfully proves cycle termination of $\phi(R_2)$: $\mathrm{T_TT_2}$ proves string termination of $\mathsf{split}(\phi(R_2))$. Interestingly, we did not find a termination proof of $\mathsf{split}(\phi(R_2))$ using AProVE.

5. Tools and Experimental Results

In this section we first explain which tools and which benchmark sets were used to evaluate the techniques presented in this paper. Thereafter we summarize and analyze the obtained results.

5.1. **Tools for Proving Cycle Termination.** We implemented several tools for proving cycle termination and cycle non-termination, which will be explained in this section.

The command line tool `cycsrs` is mainly a wrapper which allows to call different other tools and termination provers. It also allows to call the different tools in succession on a given termination problem and to distribute the time limit among the tools. The tool participated in the category on cycle rewriting in the Termination Competition 2015 [21] and in the Termination and Complexity Competition 2016 [22].

For the transformational approaches, `cycsrs` is able to perform the transformations split, rotate, or shift for a given input problem and then it calls the termination provers AProVE or $\mathrm{T_TT_2}$ to prove string termination of the transformed problem. Analogously, `cycsrs` can apply transformations split_{rel}, rotate_{rel}, and shift_{rel}, to enable the proof of relative cycle termination by showing relative string termination.

For the search for matrix interpretations described in Section 4, we implemented two tools: The prover `torpacyc` searches for matrix interpretations by using the SMT-solver `yices` targeting the logic of unquantified linear integer arithmetic with uninterpreted sort and function symbols.

The tool `tdmi` uses a similar approach, applying `yices` to find matrix interpretations, but targets the logic of quantifier-free formulas over the theory of fixed-size bit vectors (similarly as proposed in [4]). Moreover, `tdmi` is able to prove relative termination. Another difference is that `torpacyc` checks for match bound proofs, see [28]. Match bound proofs can be seen as proofs by tropical matrix interpretation, but with dimensions of often more than 100, being far beyond the dimension that is feasible for direct search for tropical matrix interpretations, typically being 3.

For proving non-termination, one technique is to prove string non-termination of the cycle rewrite problem by using a termination prover like AProVE or $\mathrm{T_TT_2}$ (which is correct due to Proposition 2.1). Another technique is to apply one of the transformations and then proving string non-termination of the transformed problem (which is correct since the transformations are complete). Additionally, we implemented a tool `cycnt` which performs a brute-force search for cycle non-termination.

The automatic transformation and the prover can also be used online via a web interface available via `http://www.ki.informatik.uni-frankfurt.de/research/cycsrs/` where also the tools and experimental data can be found.

5.2. **Benchmark Sets and Test Environment.** A problem for doing experiments is that no real appropriate test set for cycle rewriting is available. We played around with several examples like the ones in this paper, but we also wanted to test larger scale experiments. For proving cycle termination, we use the SRS_Standard-branch of the Termination Problem Data Base [23] (TPDB), which is a benchmark set for proving termination of string rewrite systems, but it was also used as problem set in the cycle rewriting category of the Termination Competition 2015 and slightly extended by some new problems in 2016. This set contains 1315 termination problems.

For relative cycle termination, we use the SRS_Relative-branch of the Termination Problem Data Base, which is a benchmark set containing 205 relative string termination problems.

In [18] we also tested some of the techniques on 50000 randomly generated string rewrite systems. We excluded them in the new tests, since most of the problems where either trivially cycle terminating or could easily be proved to be cycle non-terminating. A further difference to the benchmarks presented in [18] is the test environment: the current benchmarks where all performed on StarExec [20] – a cross community logic solving service – which made it possible to rapidly run the tests and thus also to perform tests with higher time-outs.

5.3. **Experimental Results on Cycle Termination.** Table 1 summarizes the obtained results for running several techniques to prove cycle termination on the SRS_Standard-branch of the TPDB. The first column lists the different techniques, thereafter there are two main columns: the first one lists the results where a time limit of 60 secs. was used, and the second main column lists the result for a time limit of 300 secs. There are three kinds of results: YES means that cycle termination has been proved, NO means that cycle non-termination has been proved (and thus cycle termination was disproved), and MAYBE means that no result was obtained.

The rows of Table 1 consist of five main parts.

- The first part consists of the results for the transformational approach, where each of the three transformations split, rotate, and shift was applied to the input problem and thereafter either AProVE or T_TT_2 was applied to the transformed system. We also list the numbers for combining the results of the two solvers (in the rows named "any"), which sometimes shows that the termination techniques perform differently on the same problems.
- The second part contains the summarized results of applying the two tools `torpacyc` and `tdmi` to the problems. Both tools try to prove cycle termination by searching for matrix interpretations. Note that both tools are unable to show cycle non-termination. Again the row named with "any" combines the results of both techniques.
- The third part shows the result of applying the brute-force search for cycle non-termination, using our tool `cycnt`, and the results of applying AProVE and T_TT_2 to prove string non-termination of the input problem (which implies cycle termination). We also list the numbers for combining the three techniques to prove cycle non-termination (listed in the row named "any").
- The fourth part consists of the results for a combination of the termination techniques. Here we use AProVE or T_TT_2, respectively, as back-end prover to show string termination and non-termination. In detail, the command line tool `cycsrs` is used to first run

SRS_Standard		Time limit 60 secs. YES	NO	MAYBE	Time limit 300 secs. YES	NO	MAYBE
Transformational Approach							
	AProVE	26	316	973	41	324	950
split	T_TT_2	27	164	1124	46	180	1089
	any	35	316	964	53	324	938
	AProVE	10	48	1257	10	53	1252
rotate	T_TT_2	3	0	1312	9	0	1306
	any	10	48	1257	11	53	1251
	AProVE	10	79	1226	10	89	1216
shift	T_TT_2	8	0	1307	8	1	1306
	any	10	79	1226	10	89	1216
Trace-Decreasing Matrix Interpretations							
`torpacyc`		44	0	1271	49	0	1266
`tdmi`		31	0	1284	46	0	1269
any		48	0	1267	63	0	1252
Non-Termination Check							
`cycnt`		0	580	735	0	588	727
AProVE (SRS)		0	99	1216	0	109	1206
T_TT_2 (SRS)		0	33	1282	0	62	1253
any		0	583	732	0	591	724
Combination of Techniques							
back-end: AProVE		51	511	753	62	540	713
back-end: T_TT_2		50	511	754	68	539	708
any		**60**	**583**	**682**	**83**	**591**	**651**

Table 1: Experimental results for proving cycle (non)-termination on the 1315 problems of the SRS_Standard branch of the TPDB

`torpacyc` (with 25% of the time limit), then `tdmi` (14 %), then the back-end prover (9 %) and `cycnt` (10 %) to prove cycle non-termination, and finally to apply the transformation split together with a subsequent call of the back-end prover to show (non-)termination of the transformed system (42 %).

- The last row of the table combines all results of the previous rows (where the YES- and NO-results are summed up per problem).

An overall observation is that our techniques were able to obtain a result for about the half of the problems, while the other half of the problems seem to be too hard to be proved by the techniques. This is not really surprising since the test set contains already 'hard' instances for proving *string termination*: In the Termination Competition 2015 the winning prover AProVE solved 832 out of the 1325 string termination problems (with a time limit of 300 secs.).

Considering cycle termination, we were now able to solve 643 problems with a time limit of 60 secs. (by combining all of our techniques), while in [18] only 399 problems were solved in the same time limit and on the same problem set (but on a different environment, not on

SRS_Relative		Time limit 60 secs. YES	NO	MAYBE	Time limit 300 secs. YES	NO	MAYBE
Transformational Approach							
split	AProVE	8	7	190	14	7	184
	T_TT_2	1	8	196	0	8	197
	any	8	8	189	14	8	183
rotate	AProVE	1	0	204	1	0	204
	T_TT_2	0	0	205	0	0	205
	any	1	0	204	1	0	204
shift	AProVE	1	0	204	2	0	203
	T_TT_2	0	0	205	0	1	204
	any	1	0	204	2	1	202
Trace-Decreasing Matrix Interpretations							
`tdmi`		11	0	194	21	0	184
Non-Termination Check							
`cycnt`		0	13	192	0	13	192
AProVE (SRS)		0	1	204	0	1	204
T_TT_2 (SRS)		0	1	204	0	1	204
any		0	13	192	0	13	192
Combination of Techniques							
back-end: AProVE		10	13	182	17	13	175
back-end: T_TT_2		8	13	184	16	13	176
any		**12**	**13**	**180**	**21**	**13**	**171**

Table 2: Experimental results for proving relative cycle (non)-termination on the 205 problems of the SRS_Relative branch of the TPDB

StarExec). This increase in the number of solved problems mainly comes from adding the tool `cycnt` to search for cycle non-termination. The results for checking non-termination also show that in the benchmark set many problems seem to cycle non-terminating, while they are string terminating. We expected this, since a substantial part of the problems may contain a renaming of the rule $ab \to ba$.

A further observation is that increasing the time limit allows to solve more problems, where the increase on proving cycle non-termination is rather small, while for cycle termination the increase is noticeable.

Comparing the three transformations, the transformation split leads to much better results than the other two transformations, which holds for termination and for non-termination proofs.

For proving cycle termination, problem specific methods (i.e. `torpacyc` and `tdmi`) seem to perform a little bit better than the transformational method using split.

Comparing the back-end prover, there is surprisingly no clear winner: AProVE seems to perform better for short time limits and for proving non-termination, while T_TT_2 seems to perform better for longer time limits.

5.4. **Relative Cycle Termination.** Table 2 summarizes the results on applying different techniques to the 205 problems in the SRS_Relative-branch of the TPDB. Again, the tests were run with a time limit of 60 secs. and with a time limit of 300 secs. Since `torpacyc` is not able to prove relative termination, there are no tests using `torpacyc`.

One observation of the results is that again the transformation $split_{rel}$ leads to better results than the other transformations. A further observation is that the problem specific methods (the matrix interpretations to prove relative cycle termination and `cycnt` to disprove cycle termination) perform slightly better than the transformational approach.

Finally, the number of solved problems is quite small compared to the number of problems. One reason may be that the benchmark set contains only few small rewrite systems, and many large problems. Large problems are disadvantageous for the transformational approach, since the transformations (especially the transformation split) increases the size of the problem.

5.5. **Proving Cycle Termination by Relative String Termination.** To prove cycle termination of a string rewrite system S, we can use all the techniques for proving relative cycle termination by using all rules of S as strict rules (and thus there are no weak rules). This usually does not lead to an improvement by applying automated termination tools, since the problem of showing S being terminating relative to S is equal to showing that S is terminating. However, for the transformational approaches the setting is different. Let ψ be one of the sound and complete transformations and ψ_{rel} its variant for relative cycle termination. Since for an SRS S, the transformation $\psi_{rel}(S, S)$ results in (S', R') where this inclusion is strict, i.e. $S' \subset R'$, it makes sense to analyse whether transforming S into (S', R') and subsequently proving that S' is string terminating relative to R' enables further (non-)termination proofs compared to trying to prove string termination of $\psi(S)$.

For each problem in SRS_Standard-branch of the TPDB we applied each combination of a transformation and the two termination provers AProVE and T_TT_2. Table 3 summarizes the results of this analysis, where we used a time limit of 300 secs. The numbers of proved, disproved, and open problems are listed: first for the usual approach to prove string termination of $\psi(S)$, secondly for proving relative string termination of $\psi_{rel}(S, S)$, and as a third row the combined results are shown (in the rows labeled with "any").

The results show that indeed also the technique using relative string termination as target of the transformation works for several problems. However, they lead to new cycle (non-)termination proofs in very rare cases.

6. Conclusions

We presented techniques to prove termination and relative termination for cycle rewriting. The main approach is to apply a sound and complete transformation from cycle into string rewriting. We presented and analyzed three such transformations, both for termination and relative termination. Apart from that we provided a framework covering several variants of matrix interpretations serving for proving (relative) cycle termination. Our implementations and the corresponding experimental results show that both techniques are useful in the sense that they apply for several examples for which the earlier techniques failed.

Together with the sound and complete transformation ϕ in the reverse direction from [28], the existence of a sound and complete transformation like split implies that the problems of cycle termination and string termination of SRSs are equivalent in a strong sense.

SRS_Standard	AProVE			T_TT_2		
Time limit 300 secs.	YES	NO	MAYBE	YES	NO	MAYBE
split	41	324	950	46	180	1089
split_{rel}	26	179	1110	8	213	1094
any	41	324	950	46	216	1053
rotate	10	53	1252	9	0	1306
rotate_{rel}	10	1	1304	2	1	1312
any	10	53	1252	9	1	1305
shift	10	89	1216	8	1	1306
shift_{rel}	10	6	1299	4	13	1298
any	10	89	1216	8	13	1294

Table 3: Results for proving cycle termination of the 1315 problems in the SRS_Standard branch of the TPDB by transformation into string termination problems and relative string termination problems

For instance, it implies that they are in the same level of the arithmetic hierarchy, which is Π_2^0-complete along the lines of [6]. Alternatively, Π_2^0-completeness of cycle termination can be concluded from the sound and complete transformation ϕ combined with the observation that cycle termination is in Π_2^0.

For future research, there is a need for more sophisticated techniques to prove cycle non-termination, since we conjecture that e.g. several open problems in the benchmark sets are cycle non-terminating, but our tools are not able to find a corresponding proof. A promising non-termination technique may follow the ideas of [8], some first steps in this direction have been worked out in [27].

For a fair comparison of tools for (non-)termination of cycle rewriting, a set of benchmarks is needed with more focus on cycle rewriting, rather than the current set in which nearly all systems are copied from benchmarks for string rewriting.

Acknowledgments

We thank Johannes Waldmann for fruitful remarks, in particular for his suggestions leading to Section 4 on trace decreasing matrix interpretations. We also thank the anonymous reviewers of RTA 2015 for their valuable comments on the topic, and the anonymous reviewers of this journal version for their very careful reading and valuable suggestions and comments.

References

[1] Homepage of AProVE, 2016. `http://aprove.informatik.rwth-aachen.de`.

[2] Harrie Jan Sander Bruggink, Barbara König, Dennis Nolte, and Hans Zantema. Proving termination of graph transformation systems using weighted type graphs over semirings. In Francesco Parisi-Presicce and Bernhard Westfechtel, editors, *Graph Transformation - 8th International Conference, ICGT 2015, Held as Part of STAF 2015, L'Aquila, Italy, July 21-23, 2015. Proceedings*, volume 9151 of *Lecture Notes in Computer Science*, pages 52–68. Springer, 2015.

[3] Harrie Jan Sander Bruggink, Barbara König, and Hans Zantema. Termination analysis for graph transformation systems. In Josep Diaz, Ivan Lanese, and Davide Sangiorgi, editors, *Proc. 8th IFIP International Conference on Theoretical Computer Science*, volume 8705 of *Lecture Notes in Comput. Sci.*, pages 179–194. Springer, 2014.

[4] Michael Codish, Yoav Fekete, Carsten Fuhs, Jürgen Giesl, and Johannes Waldmann. Exotic semi-ring constraints. In Pascal Fontaine and Amit Goel, editors, *10th International Workshop on Satisfiability Modulo Theories (SMT 2012)*, volume 20 of *EPiC Series*, pages 88–97. EasyChair, 2012.

[5] Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Computer-Aided Verification (CAV'2014)*, volume 8559 of *Lecture Notes in Comput. Sci.*, pages 737–744. Springer, 2014.

[6] Jörg Endrullis, Herman Geuvers, Jakob Grue Simonsen, and Hans Zantema. Levels of undecidability in rewriting. *Inf. Comput.*, 209(2):227–245, 2011.

[7] Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *J. Autom. Reasoning*, 40(2-3):195–220, 2008.

[8] Jörg Endrullis and Hans Zantema. Proving non-termination by finite automata. In Maribel Fernández, editor, *26th International Conference on Rewriting Techniques and Applications (RTA 2015)*, volume 36 of *LIPIcs*, pages 160–176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[9] Alfons Geser. *Relative Termination*. Dissertation, Universität Passau, Germany, 1990.

[10] Jürgen Giesl, Marc Brockschmidt, Fabian Emmes, Florian Frohn, Carsten Fuhs, Carsten Otto, Martin Plücker, Peter Schneider-Kamp, Thomas Ströder, Stephanie Swiderski, and René Thiemann. Proving termination of programs automatically with AProVE. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Proc. 7th International Joint Conference on Automated Reasoning (IJCAR'14)*, volume 8562 of *Lecture Notes in Comput. Sci.*, pages 184–191. Springer, 2014.

[11] Jürgen Giesl, Frédéric Mesnard, Albert Rubio, René Thiemann, and Johannes Waldmann. Termination competition (termCOMP 2015). In Amy P. Felty and Aart Middeldorp, editors, *25th International Conference on Automated Deduction (CADE 2015)*, volume 9195 of *Lecture Notes in Comput. Sci.*, pages 105–108. Springer, 2015.

[12] Jürgen Giesl and Aart Middeldorp. Transformation techniques for context-sensitive rewrite systems. *J. Funct. Program.*, 14(4):379–427, 2004.

[13] Jürgen Giesl and Hans Zantema. Liveness in rewriting. In Robert Nieuwenhuis, editor, *Proc. 14th Conference on Rewriting Techniques and Applications (RTA)*, volume 2706 of *Lecture Notes in Comput. Sci.*, pages 321–336. Springer, 2003.

[14] Dieter Hofbauer and Johannes Waldmann. Termination of $\{aa\text{->}bc,\ bb\text{->}ac,\ cc\text{->}ab\}$. *Inf. Process. Lett.*, 98(4):156–158, 2006.

[15] Dieter Hofbauer and Johannes Waldmann. Termination of string rewriting with matrix interpretations. In Frank Pfenning, editor, *Proc. 17th Conference on Rewriting Techniques and Applications (RTA)*, volume 4098 of *Lecture Notes in Comput. Sci.*, pages 328–342. Springer, 2006.

[16] Adam Koprowski and Johannes Waldmann. Arctic termination ...below zero. In Andrei Voronkov, editor, *Proc. 19th Conference on Rewriting Techniques and Applications (RTA)*, volume 5117 of *Lecture Notes in Comput. Sci.*, pages 202–216. Springer, 2008.

[17] Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean termination tool 2. In Ralf Treinen, editor, *Proc. 20th Conference on Rewriting Techniques and Applications (RTA)*, volume 5595 of *Lecture Notes in Comput. Sci.*, pages 295–304. Springer, 2009.

[18] David Sabel and Hans Zantema. Transforming Cycle Rewriting into String Rewriting. In Maribel Fernández, editor, *26th International Conference on Rewriting Techniques and Applications (RTA 2015)*, volume 36 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 285–300, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[19] Imre Simon. Recognizable sets with multiplicities in the tropical semiring. In *Mathematical Foundations of Computer Science*, volume 324 of *LNCS*, pages 107–120. Springer, 1988.

[20] Starexec, 2016. `http://www.starexec.org`.

[21] Termination Competition 2015, 2015. `http://termination-portal.org/wiki/Termination_Competition_2015`.

[22] Termination and Complexity Competition 2016, 2016. `http://termination-portal.org/wiki/Termination_and_Complexity_Competition_2016`.

[23] The termination problem data base, 2015. `http://termination-portal.org/wiki/TPDB`.

[24] Homepage of $\mathrm{T_T T_2}$, 2016. `http://cl-informatik.uibk.ac.at/software/ttt2/`.

[25] Homepage of Yices, 2016. http://yices.csl.sri.com/.
[26] Hans Zantema. Termination of term rewriting: Interpretation and type elimination. *J. Symb. Comput.*, 17(1):23–50, 1994.
[27] Hans Zantema and Alexander Fedotov. Non-termination of string and cycle rewriting by automata. In Aart Middeldorp and René Thiemann, editors, *Proceedings of the 15th International Workshop on Termination*, pages 13:1–13:5, 2016. http://cl-informatik.uibk.ac.at/workspace/events/wst2016.pdf.
[28] Hans Zantema, Barbara König, and Harrie Jan Sander Bruggink. Termination of cycle rewriting. In Gilles Dowek, editor, *Proc. Joint 25th Conference on Rewriting Techniques and Applications and 12th Conference on Typed Lambda Calculi and Applications (RTATLCA)*, volume 8560 of *Lecture Notes in Comput. Sci.*, pages 476–490. Springer, 2014.