ALFONS GESER
HANS ZANTEMA

Non-looping string rewriting

Informatique théorique et applications, tome 33, nº 3 (1999), p. 279-301
<http://www.numdam.org/item?id=ITA_1999__33_3_279_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **33** (1999) 279–301

NON-LOOPING STRING REWRITING [*]

ALFONS GESER[1] AND HANS ZANTEMA[2]

Abstract. String rewriting reductions of the form $t \to_R^+ utv$, called *loops*, are the most frequent cause of infinite reductions (non-termination). Regarded as a model of computation, infinite reductions are unwanted whence their static detection is important. There are string rewriting systems which admit infinite reductions although they admit no loops. Their non-termination is particularly difficult to uncover. We present a few *necessary* conditions for the existence of loops, and thus establish a means to recognize the difficult case. We show in detail four relevant criteria: (i) the existence of loops is characterized by the existence of looping *forward closures*; (ii) *dummy elimination*, a non-termination preserving transformation method, also preserves the existence of loops; (iii) *dummy introduction*, a transformation method that supports subsequent dummy elimination, likewise preserves loops; (iv) *bordered systems* can be reduced to smaller systems on a larger alphabet, preserving the existence and the non-existence of loops. We illustrate the power of the four methods by giving a *two-rule string rewriting system* over a *two-letter alphabet* which admits an infinite reduction but no loop. So far, the least known such system had three rules.

AMS Subject Classification. 68Q42, 20M10.

Keywords and phrases: string rewriting, semi-thue system, termination, loop, term rewriting, transformation ordering, dummy introduction, dummy elimination, overlap closure, forward closure.

* *Partially supported by grant* Ku 966/3-1 *of the DFG within the Schwerpunkt Deduktion at the University of Tübingen.*

[1] Alfons Geser, Symbolisches Rechnen, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, 72076 Tübingen, Germany; e-mail: geser@informatik.uni-tuebingen.de

[2] Department of Computer Science, Universiteit Utrecht, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands; e-mail: hansz@cs.uu.nl

Résumé. Les réductions de mots de la forme $t \to_R^+ utv$, appelées *boucles*, constituent la cause la plus fréquente de chaînes de réduction infinies. En tant que modèle de calcul, ces chaînes infinies sont indésirables et il est donc d'autant plus important de les repérer statiquement. Il existe des systèmes de réécriture de mots qui admettent des chaînes infinies même s'ils n'admettent pas de boucles. La non-termination dans ce cas est particulièrement difficile à découvrir. Nous présentons quelques conditions nécessaires pour l'existence de boucles et nous établissons ainsi une méthode pour reconnaître le cas difficile. Nous présentons en détail quatre critères significatifs : (i) l'existence de boucles est caracterisée par l'existence de boucles obtenues par l'opération de *fermeture progressive* (*forward closure* en anglais) ; (ii) l'*élimination des symboles à occurrence unique*, une méthode de transformation qui préserve la non-termination ainsi que l'existence de boucles ; (iii) *l'introduction de symbole à occurrence unique*, une méthode de transformation qui autorise l'élimination des symboles à occurrence unique en même temps qu'elle préserve les boucles ; (iv) les *systèmes à bords* peuvent être ramenés à des systèmes plus petits sur un alphabet plus grand, tout en préservant l'existence et la non-existence de boucles. Nous illustrons la puissance des quatre méthodes en donnant un *système de réécriture des mots possèdant deux règles* sur un *alphabet de deux lettres* qui admet une chaîne de réduction infinie mais aucune boucle. Jusqu'à présent, le plus petit système connu ayant ces propriétés avait trois règles.

1. Introduction

A string rewriting system (SRS, for short) is a set of pairs of strings $\ell \to r$ by which one may replace in a directed fashion equals by equals: a string of the form $c\ell d$ may be replaced by crd which is considered as a simpler expression for the same object. SRSs are a convenient means to reason about semantic equality and algorithms. The study of SRSs is likely to attract practical interest in the near future since relevant aspects of DNA computation are modelled by string rewriting.

A SRS whose rewriting relation admits no infinite reductions is called *terminating*. Termination, without being too restrictive, entails that several important properties of an SRS become decidable, *e.g.* confluence. And inductive proofs can be done based on termination. Termination of SRSs is undecidable [9]. To learn more about termination it is useful to investigate shapes of infinite reductions.

Infinite reductions are often composed of cycles. A *cycle* is a non-empty reduction where a term is rewritten to the same term. More generally, a *loop* is a non-empty reduction where the starting term re-occurs surrounded by some left and right context. In other words, a loop is a reduction of the form $t \to_R^+ utv$. It is obvious that a loop can be composed infinitely, giving an infinite reduction. In fact, the usual way to deduce non-termination is to construct a loop.

However SRSs are known which admit no loops and still are not terminating. Such non-termination is of an inherently non-periodic nature and may be difficult to detect. As a first step towards the investigation of such SRSs we envisage a method to prove that an SRS admits no loops. That is, we seek *necessary conditions for the existence of loops.* Whereas construction of a loop trivially proves the existence of a loop, we have heard yet of no way to deduce non-existence of a loop unless the SRS is terminating.

We develop four tools to prove non-existence of loops.

- Our central result characterizes the existence of loops by the existence of *looping forward closures.* Forward closures [3, 14] are restricted forms of reductions which can often be captured intuitively. A looping forward closure is one of the form $t \to_R^+ utv$.
- As a second result we obtain that *dummy elimination* [6, 24] preserves the existence of loops. A dummy elimination splits a rewrite rule $l \to r\square r'$ where $\square$ is a dummy, *i.e.* a letter that does not occur at left hand sides, into two rules $l \to r, l \to r'$. Dummy elimination is known to preserve non-termination.
- Our third result says that *dummy introduction*, a method that replaces a dead string by a dummy symbol, also preserves the existence of loops.
- Our fourth result finally states that a bordered string rewriting system can be reduced to a string rewriting system with smaller right hand sides but a larger alphabet which admits loops if and only if the original system did.

We demonstrate the usefulness of the four results in concert by a proof that there exist non-terminating, non-looping SRSs having only *two rules.* We do so by giving a witness together with a complete proof. So far, the least known non-terminating, non-looping SRS had three rules [4, 12].

2. Basic notions

We assume that the reader is familiar with termination of string rewriting. For an introduction to string rewriting see [2, 10].

SRSs are also called *semi-Thue systems.*

A *cycle* is a reduction of the form $t \to_R^+ t$; a *loop* is a reduction of the form $t \to_R^+ utv$ where u, v are strings. Here tu denotes the *concatenation* of strings t and u. The string t is also called a *prefix*, u a *suffix* of tu. Any string utv is said to contain t as a *factor.* A loop can be *composed* with itself to form a larger loop: $t \to_R^+ utv \to_R^+ uutvv$. Composition can be iterated finitely and infinitely. A SRS R is said to *admit a loop* if a loop $t \to_R^+ utv$ exists.

We will occasionally treat SRSs as term rewriting systems where we identify letters with function symbols of arity one. This allows us to apply some of the term rewriting techniques to SRSs. Termination of term rewriting is summarized in Dershowitz [4].

3. A basic example

In this section we present a three-rule SRS to illustrate the typical aspects of non-terminating, non-looping rewriting.

Proposition 1. *The three-rule SRS R given by*

$$bc \rightarrow dc \tag{1}$$

$$bd \rightarrow db \tag{2}$$

$$ad \rightarrow abb \tag{3}$$

admits an infinite reduction, but no loop.

The same system already appears in Dershowitz's survey ([4] p. 111), however without a proof that it admits no loop. A slightly more complicated example, including a proof, of a three-rule SRS with the same property was given by Kurth [12].

Proof. For every $i > 0$ there is a reduction

$$ab^ic \rightarrow_R ab^{i-1}dc \rightarrow_R^{i-1} adb^{i-1}c \rightarrow_R ab^{i+1}c$$

yielding the infinite non-looping reduction

$$abc \rightarrow_R^+ ab^2c \rightarrow_R^+ ab^3c \rightarrow_R^+ \cdots$$

Now we prove that R is non-looping. Assume that $v \rightarrow_R^+ uvw$ was a loop. During this reduction, every rewrite rule had to be applied at least once; any two-rule subset of the SRS terminates, and therefore cannot form a loop. Particularly, we may conclude that v contains both letters a and c.

Then u and w contain no letters a or c since the number of a and c letters remain unchanged by rewrite steps. The length of the factor left from the first a remains unchanged by rewriting. So u is empty. The length of the factor right from the last c can at most cause each d to be replaced by two b-s. So w is empty. But this means that $v \rightarrow_R^+ v$, a contradiction to the fact that application of rule (3) increases the length of the string. Hence R is non-looping. □

4. Looping overlap closures

When speaking about reductions, one often resorts to special reductions, called *closures*. Several characterizations of termination by closures are known. In this section we will give a characterization of looping SRSs by means of overlap closures. In the next section we will strengthen this result towards the further restricted forward closures.

Definition 2 (Overlap Closure [8]). Let an SRS R be given. The set $\mathrm{OC}(R)$ of *overlap closures* is the least set of R-reductions such that

oc1. if $(l \to r) \in R$ then $(l \to_R r) \in \mathrm{OC}(R)$,
oc2. if $(s_1 \to_R^+ t_1'x) \in \mathrm{OC}(R)$ and $(xs_2' \to_R^+ t_2) \in \mathrm{OC}(R)$ such that $x \neq \varepsilon$ then $(s_1s_2' \to_R^+ t_1'xs_2' \to_R^+ t_1't_2) \in \mathrm{OC}(R)$,
oc3. if $(s_1 \to_R^+ t_1's_2t_1'') \in \mathrm{OC}(R)$ and $(s_2 \to_R^+ t_2) \in \mathrm{OC}(R)$ then $(s_1 \to_R^+ t_1's_2t_1'' \to_R^+ t_1't_2t_1'') \in \mathrm{OC}(R)$,
oc2′. if $(s_1 \to_R^+ xt_1') \in \mathrm{OC}(R)$ and $(s_2'x \to_R^+ t_2) \in \mathrm{OC}(R)$ such that $x \neq \varepsilon$ then $(s_2's_1 \to_R^+ s_2'xt_1' \to_R^+ t_2t_1') \in \mathrm{OC}(R)$,
oc3′. if $(s_1 \to_R^+ t_1) \in \mathrm{OC}(R)$ and $(s_2't_1s_2'' \to_R^+ t_2) \in \mathrm{OC}(R)$ then $(s_2's_1s_2'' \to_R^+ s_2't_1s_2'' \to_R^+ t_2) \in \mathrm{OC}(R)$.

The fairly technical definition of overlap closure becomes clearer as soon as one considers the fate of the positions in the terms during the reduction. To this end, we use Rosen's [20] notion of *residual.*

Intuitively, residual positions "correspond" to each other in a rewrite step. For the positions of the occurrence of the left hand side of the rule, there is no corresponding position at the right hand side: they are *touched.* It is apparent that only the touched positions are essential in a reduction.

A *position* is a nonnegative integer. A position p is a *position of a string* t if $0 \le p \le |t|$ where $|t|$ denotes the length of t. It is called an *inner* position of t if $0 < p < |t|$.

Let $t = cld \to crd = t'$ be a rewrite step using rewrite rule $l \to r$. We call a position p in t *touched by* this rewrite step if p is of the form $p = |c| + v$ where v is an inner position in l. A position p from t that is not touched has a unique residual p' in t' which is defined by $p' = p$ if $p \le |c|$ and $p' = p - |l| + |r|$ else (*i.e.* if $|c| + |l| \le p \le |t|$). The residual function is inductively extended to reductions: If p' is the residual in t' of p from t by the reduction $t \to_R^* t'$, and p'' is the residual in t'' of p' from t' by the reduction $t' \to_R^* t''$, then p'' is the residual in t'' of p from t by the composite reduction $t \to_R^* t' \to_R^* t''$. Now a position p in t may be called *touched during the reduction* $t \to_R^+ t'$ if the reduction is of the form $t \to_R^* t'' \to_R t''' \to_R^* t'$ and the residual p'' in t'' of p from t by $t \to_R^* t''$ is touched in the step $t'' \to_R t'''$.

Thus we can characterize overlap closures.

Lemma 3. *If R is an SRS, then the set $\mathrm{OC}(R)$ is exactly the set of reductions $t \to_R^+ t'$ where every inner position of t is touched during the reduction.*

Proof. The proof is in two parts. First we prove that every overlap closure touches all inner positions of its starting term, by structural induction along the definition of overlap closure. For Case (oc1) the claim is obvious.

Case (oc2): Let u be an inner position in s_1s_2'. If $u < |s_1|$ then the claim follows by inductive hypothesis for $s_1 \to_R^+ t_1 = t_1'x$. Otherwise, u has a residual $u' = u - |s_1| + |t_1|$ in $t_1\sigma$ by $s_1s_2' \to_R^+ t_1s_2'$, in which case $u' - |t_1'|$ is an inner position in $s_2 = xs_2'$. The claim follows by inductive hypothesis for $s_2 \to_R^+ t_2$.

Case (oc3): Follows immediately from inductive hypothesis for $s_1 \to_R^+ t_1's_2t_1''$.

Case (oc2′): Let u be an inner position in $s'_2 s_1$. If $u > |s'_2|$ then $u - |s'_2|$ is an inner position of s_1 which is touched during $s_1 \to_R^+ t_1 = xt'_1$ by inductive hypothesis. Else u has a residual u' in $s'_2 t_1$ by $s'_2 s_1 \to_R^* s'_2 t_1$. Then $u' - |s'_2|$ is touched during $s'_2 x \to_R^+ t_2$ by inductive hypothesis.

Case (oc3′): A position $|s'_2| < u < |s'_2 s_1|$ is touched during $s'_2 s_1 s''_2 \to_R^+ s'_2 t_1 s''_2$ because $u - |s'_2|$, an inner position of s_1, is touched during $s_1 \to_R^* t_1$ by inductive hypothesis. Every other inner position u of $s'_2 s_1 s''_2$ has its residual u' in $s'_2 t_1 s''_2$ by $s'_2 s_1 s''_2 \to_R^+ s'_2 t_1 s''_2$ which is touched during $s'_2 t_1 s''_2 \to_R^+ t_2$ by inductive hypothesis.

This finishes the first part.

In the second part, we prove that a non-empty reduction $t \to_R^+ t'$ that touches all inner positions of t, is an overlap closure. To this end we perform induction on the length n of $t \to_R^n t'$. If $n = 1$ then $t \to_R t'$ must be a rule in R, otherwise there remain untouched inner positions. Then $t \to_R t'$ is an overlap closure by Case (oc1).

Now let $n > 1$, and let the reduction be $t \to_R^{n-1} t'' = cld \to_R crd = t'$. Let $0 < p_1 < p_2 < \cdots < p_m < |t|$ be the enumeration of all inner positions of t untouched during $t \to_R^{n-1} t''$. For each p_i from t let p'_i denote its residual in t'' by $t \to_R^{n-1} t''$. The expression $t[p, q]$ is to denote the factor of t that starts at position p and ends at position q in t. Now the reduction $t \to_R^{n-1} t''$ can be split into $m+1$ reductions

$$t[p_i, p_{i+1}] \to_R^* t''[p'_i, p'_{i+1}] \tag{4}$$

where $0 \le i \le m$ with the agreements $p_0 = p'_0 = 0, p_{m+1} = p'_{m+1} = |t|$.

By construction, every inner position of $t[p_i, p_{i+1}]$ is touched during (4). Every non-empty reduction of these is an overlap closure by inductive hypothesis. At least one of these reductions is non-empty.

Together with the rule $l \to r$ we have a set of at least two overlap closures. Recall the premise that every inner position of t is touched during the whole reduction. Since p_i, $1 \le i \le m$ are not touched during $t \to_R^* t''$ their residuals p'_i in t'' must be touched by $t'' \to_R t'$. So the non-empty reductions (4) may be sticked together, one after the other, with $l \to r$ to yield an overlap closure by Cases (oc2), (oc3), or (oc2′). Case (oc3′) may occur in the border case $m = 0$. □

It is now natural to ask whether the existence of loops is reducible to the existence of loops during which every position is touched. We will call such a loop, which is by Lemma 3 an overlap closure of the form $t \to_R^+ utv$, a *looping overlap closure.* Call the number n of reduction steps of a loop $t \to_R^n utv$ its *length.*

Lemma 4. *An SRS admits a loop if and only if it has a looping overlap closure. Moreover if there is a loop of length n then there is a looping overlap closure of length at most n.*

Proof. "If" is trivial; we have to prove "only if".

Let R be the SRS. We prove that from a loop $t \to_R^+ utv$ an overlap closure $t' \to_R^+ u't'v'$ can be constructed, by induction on the number of positions in t

that are not touched during the given reduction. If this number is zero, then we are finished, thanks to Lemma 3. Otherwise we construct a loop with a smaller number of untouched positions in the starting term. To this loop the inductive hypothesis applies, which yields the claim.

Suppose that $t \to_R^+ utv$ is given during which a position p in t is not touched. Since no step touches p, every string in the given reduction contains a unique residual of p, and can therefore be split into two parts. Thus the entire reduction $t \to_R^+ utv$ can be split into two reductions,

$$t_1 \to_R^* t_1' \qquad \text{and} \qquad t_2 \to_R^* t_2' \tag{5}$$

such that

$$t = t_1 t_2 \qquad \text{and} \qquad utv = ut_1t_2v = t_1't_2'$$

holds. We perform a case analysis on the lengths of ut_1 and t_1'. Case 1: ut_1 is shorter than t_1'. Then there is v' such that $t_1' = ut_1v'$, and the reduction $t_1 \to_R^* t_1'$ is non-empty. So $t_1 \to_R^+ ut_1v'$ is another loop, with less untouched positions in t_1. Case 2: ut_1 is longer than t_1'. Then t_2v must be shorter than t_2', which is symmetric to Case 1. Case 3: ut_1 and t_1' have the same length. So $t_1' = ut_1$ and $t_2' = t_2v$ hold. One of the reductions (5) must be non-empty, as $t \to_R^+ utv$ is non-empty. This reduction may be used as the wanted loop.

By construction the length of the new loop does not exceed the length of the given loop. □

5. Looping forward closures

The set of overlap closures may be complex and difficult to determine. This suggests to go for further restricted reductions.

Definition 5 (Forward Closure [3,14]). The set of *forward closures* of an SRS R is the least set $\mathrm{FC}(R)$ of R-reductions such that

fc1. if $(l \to r) \in R$ then $(l \to_R r) \in \mathrm{FC}(R)$,
fc2. if $(s_1 \to_R^+ t_1'x) \in \mathrm{FC}(R)$ and $(xs_2' \to_R^+ t_2) \in \mathrm{FC}(R)$ such that $x \neq \varepsilon$ then $(s_1s_2' \to_R^+ t_1'xs_2' \to_R^+ t_1't_2) \in \mathrm{FC}(R)$,
fc3. if $(s_1 \to_R^+ t_1's_2t_1'') \in \mathrm{FC}(R)$ and $(s_2 \to_R^+ t_2) \in \mathrm{FC}(R)$ then $(s_1 \to_R^+ t_1's_2t_1'' \to_R^+ t_1't_2t_1'') \in \mathrm{FC}(R)$.

Forward closures apply naturally as by a result of Dershowitz [3], termination of SRSs can be reduced to finiteness of reductions initiated by right hand sides of forward closures.

Theorem 6. *[3] An SRS is non-terminating if and only if the right hand side of some forward closure initiates an infinite reduction.*

We are going to strengthen Lemma 4 towards looping *forward* closures. To this end we show that one can get rid of compositions of overlap closures of types oc2′ and oc3′.

For the proofs below it turns out useful to imagine every overlap closure accompanied by its composition tree. The *composition tree* for an overlap closure c is defined to be a binary tree where the nodes are overlap closures, the root is c, and arrows point from an overlap closure to the two overlap closures that formed it. Each overlap closure is labelled by its type. Let us use a term notation for such trees: *e.g.* 1 denotes an overlap closure of type oc1; $2(1,1)$ an overlap closure of type oc2 that is formed by two overlap closures of type oc1; $3(1,2(1,1))$ is obtained by oc3 from the previous two; and so on.

For such composition trees we define a size measure: Let

$$\begin{aligned}\rho(1) &= 2,\\ \rho(2(c_1,c_2)) = \rho(3(c_1,c_2)) &= \rho(c_1)\rho(c_2),\\ \rho(2'(c_1,c_2)) = \rho(3'(c_1,c_2)) &= \rho(c_1)\rho(c_2)+1.\end{aligned}$$

Let us say that an overlap closure c_1 is smaller than an overlap closure c_2 if $\rho(c_1) < \rho(c_2)$ holds.

Lemma 7. *If there is a looping overlap closure of type oc3′ then there is also a smaller looping overlap closure having at most the same length.*

Proof. Let $u_2'u_1 \to_R^+ v_2v_1'$ be an overlap closure of type oc3′, derived from $(u_1 \to_R^+ v_1) \in \mathrm{OC}(R)$ and $(u_2'v_1u_2'' = u_2 \to_R^+ v_2) \in \mathrm{OC}(R)$. Moreover let this overlap closure be looping, *i.e.* $u_2'u_1u_2''$ is a factor of v_2, *i.e.* there are strings y, z such that $v_2 = yu_2'u_1u_2''z$.

Then $(u_2 \to_R^+ v_2) \in \mathrm{OC}(R)$ and $(u_1 \to_R^+ v_1) \in \mathrm{OC}(R)$ form a looping overlap closure $u_2 \to_R^+ yu_2'u_2u_2''z$ of type oc3. The new overlap closure is smaller. For, if c_1 and c_2 denote the composition trees of the overlap closures $u_1 \to_R^+ v_1$ and $u_2'v_1u_2'' = u_2 \to_R^+ v_2$, respectively, then $\rho(3(c_2,c_1)) = \rho(c_2)\rho(c_1) < \rho(c_1)\rho(c_2)+1 = \rho(3'(c_1,c_2))$.

By construction, the length of the new loop does not exceed the length of the given loop. □

Lemma 8. *If there is a looping overlap closure of type oc2′ then there is also a smaller looping overlap closure having at most the same length.*

Proof. Let $u_2'u_1 \to_R^+ v_2v_1'$ be an overlap closure of type oc2′, derived from $(u_1 \to_R^+ v_1 = xv_1') \in \mathrm{OC}(R)$ and $(u_2'x = u_2 \to_R^+ v_2) \in \mathrm{OC}(R)$. Let c_1, c_2 denote the composition trees of the two overlap closures. The size of the composed overlap closure is $\rho(2'(c_1,c_2)) = \rho(c_1)\rho(c_2)+1$. Moreover let the composed overlap closure be looping, *i.e.* $u_2'u_1$ is a factor of v_2v_1', *i.e.* there are strings y, z such that $v_2v_1' = yu_2'u_1z$.

Distinguish cases whether $|u_1z| \le |v_1'|$ or $|z| < |v_1'| < |u_1z|$ or $|v_1'| \le |z|$ holds.

In the first case u_1 is a factor of v_1' whence $u_1 \to_R^+ xv_1'$ is a smaller looping overlap closure, for its size is only $\rho(c_1)$.

In the second case $u_1 = wu'_1$, $v_2 = yu'_2w$, and $v'_1 = u'_1z$ for non-empty strings w, u'_1. Define $v'_2 = yu'_2$. Then $(u_2 \to_R^+ v'_2w) \in \mathrm{OC}(R)$ and $(wu'_1 \to_R^+ v_1) \in \mathrm{OC}(R)$ form an overlap closure $u_2u'_1 \to_R^+ v'_2v_1$ of type oc2. The new overlap closure is looping: $u_2u'_1 = u'_2xu'_1$ is a factor of $v'_2v_1 = v'_2xv'_1 = yu'_2xu'_1z$. Its size is $\rho(2(c_2, c_1)) = \rho(c_2)\rho(c_1)$ which is less than $\rho(2'(c_1, c_2))$.

In the third case u'_2u_1 is a factor of $v_2 = yu'_2u_1z'$ where string z' is given by $z = z'v'_1$. Then $(u_2 \to_R^+ yu'_2u_1z') \in \mathrm{OC}(R)$ and $(u_1 \to v_1) \in \mathrm{OC}(R)$ form an overlap closure $u_2 \to_R^+ yu'_2v_1z'$ of type oc3. The new overlap closure is looping: $u_2 = u'_2x$ is a factor of $yu'_2v_1z' = yu'_2xv'_1z'$. Its size is $\rho(3(c_2, c_1)) = \rho(c_2)\rho(c_1)$ which is less than $\rho(2'(c_1, c_2))$.

By construction, the length of the new loop does not exceed the length of the given loop. □

Theorem 9. *A SRS admits a loop if and only if it has a looping* forward *closure. Moreover if there is a loop of length n then there is a looping forward closure of length at most n.*

Proof. Suppose that R is a string rewriting system that admits a loop. By Lemma 4 there is a looping overlap closure, $u \to_R^+ v$. We have to show that if this is not a forward closure then there is still a smaller overlap closure. By induction on ρ then the claim follows.

Suppose that $u \to_R^+ v$ is not a forward closure, *i.e.* in its composition tree some label $2'$ or $3'$ occurs. If the outermost occurrence of a label $2'$ or $3'$ is at the root then Lemmas 7 and 8, respectively, take care of a smaller overlap closure.

Otherwise the composition tree contains a pattern $f(f'(c_1, c_2), c_3)$ or a pattern $f(c_1, f'(c_2, c_3))$ where $f \in \{2, 3\}$, $f' \in \{2', 3'\}$. We show that then the composition tree can be rearranged such that the size decreases. The rearrangement is expressed by rewriting rules listed in Table 1. It is tedious but easy to verify that, given an overlap closure together with its composition tree, in each case one of the rules describes a valid rearrangement of the composition tree such that the resulting overlap closure remains the same. Some rules have an "a" and a "b" variant; the "a" variant applies when c_2 overlaps with c_1 or c_3 (depending on the rule), the "b" variant otherwise.

It remains to show that every such rearrangement decreases the size of the composition tree.

To prove that each instance of a rearrangement rule strictly decreases ρ, observe that for left hand sides $\rho(f(f'(c_1, c_2), c_3)) = \rho(c_1)\rho(c_2)\rho(c_3) + \rho(c_3)$ and $\rho(f(c_1, f'(c_2, c_3))) = \rho(c_1)\rho(c_2)\rho(c_3) + \rho(c_1)$, respectively, and that for all right hand sides ρ yields $\rho(c_1)\rho(c_2)\rho(c_3) + 1$. Strict decrease follows from $\rho(c) \geq 2$ for every composition tree c.

ρ is strictly monotonic, so strict decrease holds also for rearrangement at a non-root node of the composition tree.

By construction, the length of the new loop does not exceed the length of the given loop. □

TABLE 1. Rearrangement rules for composition trees.

$$2(2'(c_1,c_2),c_3) \to 2'(c_1,2(c_2,c_3)) \tag{1a}$$
$$2(2'(c_1,c_2),c_3) \to 2'(2(c_1,c_3),c_2) \tag{1b}$$
$$2(c_1,2'(c_2,c_3)) \to 2'(2(c_1,c_2),c_3) \tag{2a}$$
$$2(c_1,2'(c_2,c_3)) \to 2'(c_2,2(c_1,c_3)) \tag{2b}$$
$$3(2'(c_1,c_2),c_3) \to 2'(c_1,2(c_2,c_3)) \tag{3a}$$
$$3(2'(c_1,c_2),c_3) \to 2'(3(c_1,c_3),c_2) \tag{3b}$$
$$3(c_1,2'(c_2,c_3)) \to 2'(3(c_1,c_2),c_3) \tag{4}$$
$$2(3'(c_1,c_2),c_3) \to 3'(c_1,2(c_2,c_3)) \tag{5}$$
$$2(c_1,3'(c_2,c_3)) \to 3'(2(c_1,c_2),c_3) \tag{6a}$$
$$2(c_1,3'(c_2,c_3)) \to 3'(c_2,2(c_1,c_3)) \tag{6b}$$
$$3(3'(c_1,c_2),c_3) \to 3'(c_1,3(c_2,c_3)) \tag{7}$$
$$3(c_1,3'(c_2,c_3)) \to 3'(3(c_1,c_2),c_3) \tag{8}$$

Example 1. We can now give an alternative proof that the three-rule SRS R in Proposition 1 admits no loops, by showing that it has no looping forward closures. Forward closures are at most of the forms

$$bd^{n+1} \to_R^+ d^{n+1}b, \tag{6}$$
$$bd^m c \to_R^+ d^{m+1}c, \tag{7}$$
$$ad^{n+1} \to_R^+ awb, \tag{8}$$
$$ad^{n+1}c \to_R^+ aw'c \tag{9}$$

where $m,n \geq 0$ and $w,w' \in \{b,d\}^*$, $|w|_b + 2|w|_d = 2n+1$, $|w'| > n+1$. To see that this set of reductions contains all forward closures check that it contains each rule in R and is closed under operations oc2 and oc3.

Not all of these need be forward closures, but this is not essential for our argument. One can show that none of these reductions is of the form $t \to_R^+ utv$. Reductions of forms (6, 7), or (8) can also be generated by a terminating subset of R each, hence cannot be loops. Reductions of form (9) cannot be cycling by a length argument, and cannot be properly looping because uv non-empty must contain one of a, c. So there is no looping forward closure. Hence R is non-looping by Theorem 9.

Example 2. Let S be as R above where c has been identified with a.

$$\begin{aligned} ba &\to da \\ bd &\to db \\ ad &\to abb. \end{aligned}$$

It has no loops either, as we are going to show. Forward closures in S are at most of the forms

$$bd^{n+1} \to_S^+ d^{n+1}b, \tag{10}$$
$$bd^m a \to_S^+ d^{m+1}a, \tag{11}$$
$$ad^{n+1} \to_S^+ awb, \tag{12}$$
$$bd^m ad^{n+1} \to_S^+ d^{m+1}awb, \tag{13}$$
$$u \to_S^+ u', \tag{14}$$
$$bd^m u \to_S^+ d^{m+1}u', \tag{15}$$
$$ud^{n+1} \to_S^+ u'wb, \tag{16}$$
$$bd^m ud^{n+1} \to_S^+ d^{m+1}u'wb \tag{17}$$

where $m, n \geq 0$, $w \in \{b, d\}^*$, $|w|_b + 2|w|_d = 2n + 1$, and

$$\begin{aligned} u &= ad^{q_1}ad^{q_2} \cdots ad^{q_k}a, \\ u' &= au_1'au_2' \cdots au_k'a, \end{aligned}$$

for some $k \geq 1$ and $|u_i'| > q_i \geq 1$ and $u_i' \in \{b, d\}^*$ for all $1 \leq i \leq k$.

None of these reductions can be looping. Reductions of forms (10),...,(12) cannot be looping because each is formed by a terminating subset of the rewrite rules. If a reduction of the form (13) were looping then so were a reduction of the forms (10),...,(12).

Because the number of a symbols is not changed by rewriting, and $u_i \neq u_i'$ holds for all $1 \leq i \leq k$, none of the reductions of forms (14),...,(17) can be looping either. So S admits no looping forward closures. By Theorem 9 S admits no loops.

Example 3. The four-rule SRS R'

$$\begin{aligned} bc &\to dc \\ be &\to eeb \\ ed &\to de \\ ade &\to aeeb \end{aligned}$$

over the alphabet $\mathcal{A} = \{a, b, c, d, e\}$ admits infinite, but not looping reductions as we are going to show. The set of forward closures is contained in

$$be^n \to^+_{R'} e^{2n}b, \tag{18}$$

$$ed^m \to^+_{R'} d^m e, \tag{19}$$

$$be^n c \to^+_{R'} wc, \tag{20}$$

$$ade^{n+1} \to^+_{R'} ae^{2n+2}b, \tag{21}$$

$$ade^{n+1}c \to^+_{R'} aw'c \tag{22}$$

where $w \in \{b, e\}^*$, $|w| = n + 1$, $|w|_d = 1$, $w' \in \{b, d, e\}^*$, $|w'| > n + 2$, $|w'|_{be} = 1$. Reductions of form (18) obviously cannot be loops; neither can reductions of form (22) be. By Theorem 9 the SRS admits no loop. It admits, however, for every $i \geq 1$ a reduction of the form

$$ade^i c \to_{R'} aeebe^{i-1}c \to^{i-1}_{R'} ae^{2i}bc \to_{R'} ae^{2i}dc \to^{2i}_{R'} ade^{2i}c,$$

and so the infinite reduction

$$ade^1c \to^+_{R'} ade^2c \to^+_{R'} ade^4c \to^+_{R'} ade^8c \to^+_{R'} \cdots$$

The set of forward closures of length bounded by n is finite and can effectively be computed. So there is a decision procedure for the existence of loops of length bounded by n.

Corollary 10. *For every $n > 0$ the following is decidable: given a finite SRS, does it admit a loop of length $\leq n$?*

In the next few sections we investigate which transformations preserve looping SRSs.

6. Dummy elimination

Dummy elimination is a useful method to split right hand sides of rewrite rules at symbols ("dummies") that do not occur at left hand sides. Dummy elimination is known to preserve non-termination. It preserves loopingness, too, as we show next.

For our presentation we fix an alphabet $\mathcal{A}$ including a dummy symbol $\Box$.

Definition 11. For each string of the form $s = r_1\Box r_2 \cdots \Box r_n$ where $r_i \in (\mathcal{A} \setminus \{\Box\})^*$ for all $i \in \{1, \dots, n\}$, let $\mathcal{E}(s) =_{\text{def}} \{r_1, \dots, r_n\}$.

Definition 12. [24] For an SRS R on the alphabet $\mathcal{A}$ where the symbol $\Box \in \mathcal{A}$ does not occur on left hand sides of R, let

$$E(R) =_{\text{def}} \{l \to u \mid (l \to r) \in R \wedge u \in \mathcal{E}(r)\}.$$

Theorem 13. *[24] Let R be an SRS. If $\rightarrow_{E(R)}$ terminates then $\rightarrow_R$ terminates.*

Theorem 14. *Let R be an SRS. If $\rightarrow_R$ admits loops then $\rightarrow_{E(R)}$ admits loops.*

In the proof we utilize the following characterization of the existence of loops.

Definition 15. For an SRS R, a relation $>_R$ on strings is defined by $v >_R w$ if there exist q, q' such that $v \rightarrow_R^+ qwq'$.

Proposition 16. *The relation $>_R$ is transitive. The SRS R admits no looping reductions if and only if $>_R$ is irreflexive.*

Proof of Theorem 14. Let R be an SRS, and let $S = E(R)$.

We claim that $s \rightarrow_R t$ implies $\mathcal{E}(s) >_S^{mult} \mathcal{E}(t)$. Suppose $s \rightarrow_R t$ using rule $l \rightarrow r$ in R, which means that s is of the form $s = s_1 l s_2$. Let us first assume that s_1, s_2 do not contain $\square$, whence $\mathcal{E}(s) = \{s_1 l s_2\}$. If r does not contain $\square$, then $\mathcal{E}(t) = \{s_1 r s_2\}$, and the claim follows by $s_1 l s_2 \rightarrow_S s_1 r s_2$, as rule $l \rightarrow r$ is also in S. Else, suppose that $r = r_1 \square r_2 \square \cdots \square r_n$ with $n > 1$. Then $\mathcal{E}(t) = \{s_1 r_1, r_2, r_3, \ldots, r_{n-1}, r_n s_2\}$. Now by definition $s_1 l s_2$ is greater than every element of $\mathcal{E}(t)$, hence again the claim follows. By closure under multiset union, this reasoning carries over to the case where s_1 or s_2 contain dummy symbols and the claim has been proved.

Now it is easy to show that this extends to: $s >_R t$ implies $\mathcal{E}(s) >_S^{mult} \mathcal{E}(t)$. The following chain of implications finishes the proof. Suppose S admits no loop; then $>_S$ is irreflexive; then $>_S^{mult}$ is irreflexive; then $>_R$ is irreflexive; so R admits no loop. $\square$

The following example shows that dummy elimination preserves neither termination nor non-loopingness.

Example 4. Let R be the one-rule SRS $gf \rightarrow gg\square fff$. The system $E(R) = \{gf \rightarrow gg, gf \rightarrow fff\}$ derived by dummy elimination, admits the following loop.

$$gff \rightarrow_{E(R)} ggf \rightarrow_{E(R)} gfff.$$

In contrast, R terminates as its only forward closure is $gf \rightarrow gg\square fff$, the right hand side $gg\square fff$ of which is unable to initiate an infinite reduction.

Example 5. The two-rule SRS

$$R = \left\{ \begin{array}{lcl} bad & \rightarrow & dadcbabb \\ bd & \rightarrow & db \end{array} \right.$$

over the alphabet $\mathcal{A} = \{a, b, c, d\}$ admits an infinite reduction but admits no loop. Non-termination follows from the existence of reductions $bab^i ad \rightarrow_R^+ ubab^{i+1}adv$ with suitable strings u, v for all $i > 0$. For the proof that R admits no loop, we apply dummy elimination to R with c the dummy, yielding the system $E(R)$ as follows.

$$E(R) = \begin{cases} bad & \to dad \\ bad & \to babb \\ bd & \to db. \end{cases}$$

Every $E(R)$-reduction is also a S-reduction where S is the SRS of Example 2. Since S admits no loop, $E(R)$ admits no loop either. By Theorem 14 then, R admits no loop either.

7. Dummy introduction for loops

A dead factor in a string, *i.e.* a factor that will never be touched by any rewrite step, may be replaced by a dummy symbol. In view of applicability of dummy elimination (described in the previous section) such a replacement is a valuable preprocessing step, which is therefore called *dummy introduction.* Dummy introduction preserves not only non-termination but also the existence of loops.

Definition 17. A position p is called *dead* (for R) in a string t if no R-derivation starting from t touches p. Likewise a string d is called a *dead factor* (for R) in context (v, w) if d is R-irreducible and the positions $|sv|$ and $|sv| + |d|$ are dead in the string $svdwu$ for all strings s, u.

Technically, a dummy introduction is given by an SRS of the form

$$T = \{vdw \to v\Box w \mid v \in V, w \in W\}$$

with the intended meaning that the (potentially empty) factor d is dead within each left context $v \in V$ and right context $w \in W$. We next give a technical criterion, *shieldedness*, by which this claim can be checked effectively.

Definition 18 (Left Overlap). A string l is said to *left overlap* a string r if there are strings l', r' and x where $l = l'x$, $r = xr'$, and x and $l'r'$ are non-empty. Likewise, r is called to *right overlap* l in this case. Two strings are said to overlap if one is a factor of the other or one left overlaps the other.

Definition 19 (Shielded [23]). Let R be an SRS. Let d be a string and let V, W be non-empty sets of strings over $\mathcal{A}$ such that d is non-empty if the empty string is in V and in W. Then d is called *shielded* by V, W if moreover

1. every overlap of a left hand side l of a rewrite rule $l \to r$ in R with a string vdw, $v \in V, w \in W$ already is either a left overlap with v or a factor of v or a right overlap with w or a factor of w; and
2. for each left overlap of $l = l'x$, $v = xv'$ of l with v, a suffix of rv' is in V,
3. for each $v = v'lv''$, a suffix of $v'rv''$ is in V,
4. for each right overlap of $l = yl''$, $w = w''y$ of l with w, a prefix of $w''r$ is in W, and
5. for each $w = w'lw''$, a prefix of $w'rw''$ is in W.

$$\begin{tikzcd} t = vdw \arrow[r, "l\to r"'] \arrow[d, "T"] & t' = v'dw' \arrow[d, "T"] \\ u = v\Box w \arrow[r, "l\to r"'] & v'\Box w' \end{tikzcd}$$

FIGURE 1. The situation in Lemma 24. Either $v = v'$ and $w \to_{l\to r} w'$ or $v \to_{l\to r} v'$ and $w = w'$.

Lemma 20. *[23] Condition* (1) *in Definition 19 is equivalent to: for all strings s, u, t' and strings $v \in V, w \in W$, the indicated occurrence of d is not touched in the step $svdwu \to_R t'$.*

Lemma 21. *[23] Conditions* (2) *and* (3) *in Definition 19 are equivalent to: for all strings $v \to_R v'$ where a suffix of v is in V, also a suffix of v' is in V.*

Lemma 22. *[23] Conditions* (4) *and* (5) *in Definition 19 are equivalent to: for all strings $w \to_R w'$ where a prefix of w is in W, also a prefix of w' is in W.*

Lemma 23. *[23] If d is shielded by sets V, W of strings then d is dead in context (v, w) for every $v \in V, w \in W$. Conversely, if d is dead in some context (v, w) then there exist sets V, W of strings such that $v \in V, w \in W$ and d is shielded by V, W.*

The sets V, W may be infinite as in the following example[1].

Example 6. Let $v = a$, $d = b$, $w = \varepsilon$, and let $R = \{xa \to ax, yx \to \varepsilon, yb \to \varepsilon\}$. Here we get $V = ax^*$, $W = \varepsilon$.

It is routine to prove the following commutation property.

Lemma 24. *Let $T = \{vdw \to v\Box w \mid v \in V, w \in W\}$ be a dummy introduction such that d is shielded by V, W for the rule $(l \to r)$. Then for all strings t, t', u there is a string u' such that $u \leftarrow_T t \to_{l\to r} t'$ implies $u \to_{l\to r} u' \leftarrow_T t'$.*

The situation is illustrated in Figure 1.

Lemma 25. *Let $T = \{vdw \to v\Box w \mid v \in V, w \in W\}$ be a dummy introduction such that d is shielded by V, W for the SRS R. Then d is shielded by V, W also for the SRS*

$$\{s \to t \mid (s \to_R^+ t) \in \mathrm{OC}(R)\}.$$

Proof. Let the premises hold. We check against the conditions for shieldedness of d by V, W for the rule $s \to t$ where $s \to_R^n t$ is a overlap closure.

Condition (1): Let s overlap with vdw. No position in the box in $v\boxed{d}w$ (including the borders) is touched during $s \to_R^n t$, a fact that can be proven easily by induction on n. On the other hand, every inner position of s is touched,

[1]Suggested by an anonymous referee

by Lemma 3. Hence every overlap of s with vdw is either a left overlap with v, or a factor of v, or a right overlap with w, or a factor of w.

Condition (2) and (3): We use Lemma 21 for convenience. Suppose that $v \to_{s\to t} v'$ where a suffix of v is in V. By definition of $s \to t$ then $v \to_R^n v'$ for some $n > 0$. By induction on n, using Lemma 21 for R, we get that a suffix of v' is in V. Using Lemma 21 for $s \to t$ finishes the proof.

Conditions (4) and (5) are proven dually. □

Dummy introduction is a special case of the following non-termination preserving transformation.

Theorem 26. *[24] Let R, S, and T be string rewriting systems. If*

1. *S terminates,*
2. *$R \subseteq \to_S^+ \leftarrow_T^*$,*
3. *$CP(T, R) \subseteq \to_R^+ \leftarrow_T^*$,*

then R terminates.

Since the transformation $R \mapsto S$ preserves non-termination, provided that Conditions (2) and (3) hold, it is natural to ask whether it also preserves the existence of loops. In other words, if Conditions (2) and (3) hold, and R admits loops, does S admit loops, too? The answer is negative as the following counterexample shows.

Example 7. Let R, S, and T be the SRSs

$$R = \begin{cases} bc \to dc \\ bd \to db \\ ad \to pab \end{cases} \qquad S = \begin{cases} bc \to dc \\ bd \to db \\ ad \to abb \end{cases} \qquad T = \{pa \to ab\},$$

respectively. Condition (2) of Theorem 26 holds by $ad \to_S abb \leftarrow_T pab$. Condition (3) holds since $abd \leftarrow_T pad \to_R ppab$ satisfies $abd \to_R adb \to_R pabb \leftarrow_T ppab$. R has a loop

$$adc \to pabc \to padc$$

but S has no loop by Proposition 1. Hence the transformation $R \mapsto S$ does not preserve the existence of loops.

Surprisingly, dummy introduction does preserve the existence of loops.

Theorem 27. *Let R be an SRS and let $T = \{vdw \to v\Box w \mid v \in V, w \in W\}$ be a dummy introduction for it such that d is shielded by V, W. Let S be an SRS such that $R \subseteq \to_S^+ \leftarrow_T^*$. Then S admits a loop if R admits a loop.*

Proof. By straightforward induction, from Lemma 24 one can prove that

$$\leftarrow_T^m \to_R^n \subseteq \to_R^n \leftarrow_T^m \tag{23}$$

for all m and n. Next,

$$\to_R^n \subseteq \to_S^+ \leftarrow_T^* \tag{24}$$

holds for all n. This is proven by induction on n as follows. For $n = 1$ the claim follows from the premise $R \subseteq \to_S^+ \leftarrow_T^*$ by the definition of $\to_R$. For $n > 1$, we have the reasoning

$$\to_R^n \underset{\text{premise}}{\subseteq} \to_S^+ \leftarrow_T^* \to_R^{n-1} \underset{(23)}{\subseteq} \to_S^+ \to_R^{n-1} \leftarrow_T^* \underset{\text{IH},n-1}{\subseteq} \to_S^+ \to_S^+ \leftarrow_T^* \leftarrow_T^* .$$

Assume now that R admits a loop $t \to_R^+ ptq$ where $t, p, q \in \mathcal{A}^*$. According to Lemma 4, we may assume that this is an overlap closure. By (24) there is u such that $t \to_S^+ u \leftarrow_T^k ptq$ for some k. We prove by induction on k that every string in the reduction $ptq \to_T^k u$ contains t as a factor.

To prove the claim, we show that if $t' \to_T t''$ and t is a factor of t', then t is a factor of t'' as well. The property that t is a factor of t' may equivalently be expressed by the existence of a string u' such that $t' \to_{t \to ptq} u'$. We use the fact that $(t \to_R^+ ptq) \in \mathrm{OC}(R)$ whence by Lemma 25, d is shielded by V, W for the SRS $\{s \to t \mid (s \to_R^+ t) \in \mathrm{OC}(R)\}$. With that, Lemma 24 yields the commuting diagram

$$\begin{tikzcd}
t' \arrow[r, "t \to ptq"'] \arrow[d, "T"] & u' \arrow[d, "T"] \\
t'' \arrow[r, "t \to ptq"'] & u''
\end{tikzcd}$$

Since t'' admits a rewrite step for rule $t \to ptq$, it follows that t'' contains t as a factor.

So every term in the reduction $ptq \to_T^k u$ contains t as a factor. We conclude that u has t as a factor, and so $t \to_S^+ u$ is a loop for S. □

Example 8. The SRS

$$bc \to dc \tag{1}$$
$$bd \to db \tag{2}$$
$$ad \to abbbabb \tag{3}$$

contains the dead part $abbb$ in rule (3). To prove that it does not loop, choose $T = \{abbba \to \Box a\}$. It is easily verified that $d = abbb$ is shielded by $V = \{\varepsilon\}, W = \{a\}$.

Example 9. The two-rule SRS

$$bad \to dadbabb$$
$$bd \to db$$

over the alphabet $\mathcal{A} = \{a, b, d\}$ admits an infinite reduction but admits no loop. Let R be the two-rule system. Non-termination follows from the existence of reductions $bab^iad \to_R^+ ubab^{i+1}adv$ for all $i > 0$. For the proof that R admits no loop, we transform it to the system from Example 5 of which we already derived non-loopingness. The transformation is done by the dummy transformation defined by

$$T = \{vw \to v\Box w \mid v \in \{ad, abb\}, w \in \{ba, da\}\} \cdot$$

Check that the only overlaps of left hand sides of R with vw are of the form $b\,ad\,w$ and $v\,ba\,d$ (the overlapping region is enclosed by spaces). It is straightforward to verify that the empty string is shielded by $V = \{ad, abb\}, W = \{ba, da\}$.

By Theorem 27 then R admits no loop either.

8. Bordered string rewriting systems

Shikishima-Tsuji *et al.* [21] have reduced the termination problem of confluent one-rule SRSs to that of non-overlapping one-rule SRSs. Their encoding idea was introduced by Adjan and Oganesjan [1] to attack the word problem for one-rule string rewriting. One step of the reduction applies under less restrictive premises.

Definition 28. A string s is called *bordered* if there is a non-empty string z such that z is both a prefix and a suffix of s. A SRS R is called *bordered* if there is a non-empty string z such that every left hand side of R and every right hand side of R each has prefix and suffix z. The shortest such z is called the *border* of the string, SRS, respectively.

By Wrathall's characterization [22], a length increasing one-rule SRS is confluent if and only if every self-overlap of ℓ is also a self-overlap of r. So particularly every confluent, overlapping one-rule SRS is bordered.

It is obvious that a border has no self-overlaps. We will denote the border string by the symbol $\diamond$ from now on. Let us assume that the alphabet Σ has at least two letters. Let $C = \Sigma^* \setminus \Sigma^* \diamond \Sigma^*$ denote the set of strings that do not contain the border as factor. The set $\diamond C$ forms a *code*, in other words, $\diamond C$ is a free generator of the submonoid of Σ^* generated by $\diamond C$.

Proposition 29. *For a string $\diamond$ that has no self-overlaps, every string $s \in \Sigma^*$ has a unique decomposition $s = s_0 \diamond s_1 \diamond \ldots \diamond s_n$ where $s_0, s_1, \ldots, s_n \in C$.*

Now let Ω be an (infinite) alphabet bijective to C via the function $f : \Omega \to C$. Then Ω^* and $(\diamond C)^*$ are isomorphic via the isomorphism $f^* : \Omega^* \to (\diamond C)^*$ defined by $f^*(\varepsilon) = \varepsilon$, $f^*(\omega) = \diamond f(\omega)$, $\omega \in \Omega$, $f^*(xy) = f^*(y)f^*(y)$, $x, y \in \Omega^+$.

Let $\phi : \Omega^* \to (\diamond C)^* \diamond$ be defined by $\phi(x) = f^*(x)\diamond$. Then ϕ is bijective with inverse

$$\phi^{-1}(\diamond s_1 \diamond s_2 \diamond \ldots \diamond s_n \diamond) = f^{-1}(s_1)f^{-1}(s_2)\ldots f^{-1}(s_n)$$

for all $s_1, s_2, \ldots, s_n \in C$. By Proposition 29 ϕ^{-1} is well-defined and satisfies $\phi^{-1}(s \diamond t) = \phi^{-1}(s \diamond)\phi^{-1}(\diamond t)$ for all $s \in (\diamond C)^*$, $t \in (C \diamond)^*$.

Let R be a bordered SRS over alphabet Σ with border $\diamond$. Then $\ell, r \in (\diamond C)^* \diamond$ for every rewrite rule $\ell \to r$ in R. Define $\phi^{-1}(R) = \{\phi^{-1}(\ell) \to \phi^{-1}(r) \mid (\ell \to r) \in R\}$. Conversely, if S is a SRS over (a subset of) alphabet Ω then $\phi(S) = \{\phi(\ell) \to \phi(r)\}$ is bordered with border $\diamond$. Moreover $\phi(\phi^{-1}(R)) = R$ and $\phi^{-1}(\phi(S)) = S$.

Lemma 30 ([21] Lem. 4). *Let R be a bordered SRS with border $\diamond$. Then the following hold for all $s, t \in \Sigma^*$, $x, y \in \Omega^*$.*

1. *If $x \to_{\phi^{-1}(R)} y$ then $\phi(x) \to_R \phi(y)$.*
2. *If $s \to_R t$ then there are unique $u, u' \in C^*$ and $s', t' \in (\diamond C)^* \diamond$ such that $s = us'u'$, $t = ut'u'$ and $\phi^{-1}(s') \to_{\phi^{-1}(R)} \phi^{-1}(t')$.*
3. *x is a factor of y if and only if $\phi(x)$ is a factor of $\phi(y)$.*

Proof. By Proposition 29 a bordered rule $\ell \to r$ with border $\diamond$ can be decomposed uniquely into

$$\diamond \ell_1 \diamond \ell_2 \diamond \ldots \diamond \ell_m \diamond \to \diamond r_1 \diamond r_2 \diamond \ldots \diamond r_n \diamond$$

where $\ell_1, \ell_2, \ldots, \ell_m, r_1, r_2, \ldots, r_n \in C$.

For the first claim, let $x = z\phi^{-1}(\ell)z' \to_{\phi^{-1}(R)} z\phi^{-1}(r)z' = y$. Then $\phi(x) = u\ell u' \to_R uru' = \phi(y)$ where $u, u' \in \Sigma^*$ are given by $u\diamond = \phi(z)$ and $\diamond u' = \phi(z')$.

Now for the second claim let $s = u\ell u'$ where $u \in (\diamond C)^*$ and $u' \in (C \diamond)^*$. By definition of rewrite step, $y = uru'$, so $t \in (\diamond C)^* \diamond$. Moreover,

$$\phi^{-1}(s) = \phi^{-1}(s' \diamond)\phi^{-1}(\ell)\phi^{-1}(\diamond u') \to_{\phi^{-1}(R)} \phi^{-1}(u \diamond)\phi^{-1}(r)\phi^{-1}(\diamond u') = \phi^{-1}(t).$$

This proves the second claim.

The third claim is an immediate consequence of the first and second because x is a factor of y if and only if y is reducible by some rewrite rule $x \to x'$ for an arbitrary $x' \in \Omega^*$, and a similar rule for factors over Σ^*. □

Shikishima-Tsuji *et al.* prove the following result.

Theorem 31. *[21] For every confluent one-rule SRS $R = \{\ell \to r\}$ there is effectively a non-overlapping one-rule SRS $S = \{\ell' \to r'\}$, $|r'| < |r|$ over another alphabet such that R terminates if and only if S terminates.*

In their proof they show that every overlapping one-rule SRS can be reduced such that the number of self-overlaps of the left hand side decreases (by 1). By induction on the number of self-overlaps of the left hand side they obtain finally a one-rule SRS which is non-overlapping.

A close observation shows that in order to do one such reduction step, it suffices that $\{\ell \to r\}$ is bordered. The reduction readily carries over to the case of arbitrarily many rules.

Theorem 32 ([7] for $|R| = 1$). *A bordered SRS R terminates if and only if $\phi^{-1}(R)$ terminates. Moreover $|r| > |\phi^{-1}(r)|$ holds for every rewrite rule $\ell \to r$ in R.*

Proof. The inequality $|r| > |\phi^{-1}(r)|$ is immediate by $|r| \geq |r|_\diamond \geq |\phi^{-1}(r)| + 1$, where $|r|_\diamond$ denotes the number of occurrences of $\diamond$ as a factor in r.

We show that the existence of an infinite reduction in the one system entails the existence of an infinite reduction in the other. If there is an infinite reduction $s^0 \rightarrow_{\phi^{-1}(R)} s^1 \rightarrow_{\phi^{-1}(R)} \cdots$ then by Lemma 30 there is a corresponding infinite reduction $\phi(s^0) \rightarrow_R \phi(s^1) \rightarrow_R \cdots$. Conversely, if there is an infinite reduction $z^0 \rightarrow_R z^1 \rightarrow_R \cdots$ then there is an infinite reduction $x^0 \rightarrow_R x^1 \rightarrow_R \cdots$ where $x^0 \in (\diamond\,\Sigma^*)^* \diamond$ is such that $z^0 = z'x^0z''$, $z', z'' \in \Sigma^* \backslash \Sigma^* \diamond \Sigma^*$. The z', z'' are nowhere used during the reduction whence they may be stripped. Now by Lemma 30 there is a corresponding infinite reduction $\phi^{-1}(x^0) \rightarrow_{\phi^{-1}(R)} \phi^{-1}(x^1) \rightarrow_{\phi^{-1}(R)} \cdots$ □

For example, $R = \{ababa \rightarrow aaabbba\}$ is bordered with border a, but not confluent: The self-overlap aba of $\ell = ababa$ is not a self-overlap of $r = aaabbba$. With Ω the set of nonnegative integers and $f(b^n) = n$, one gets $\phi^{-1}(R) = \{11 \rightarrow 003\}$ which is obviously terminating by counting 1 symbols. So R is terminating as well.

The following is particularly interesting if one is interested in loops.

Theorem 33. *A bordered SRS R has a loop of length n if and only if $\phi^{-1}(R)$ has a loop of length n.*

Proof. Let R have a loop of length $n > 0$, say $s \rightarrow_R^n t$. Then by applying Lemma 30 n times one gets $\phi^{-1}(s') \rightarrow^n_{\phi^{-1}(R)} \phi^{-1}(t')$. Now since $s = us'u'$ is a factor of $t = ut'u'$, s' is a factor of t', and by Lemma 30 $\phi^{-1}(s')$ is a factor of $\phi^{-1}(t')$. Hence $\phi^{-1}(s') \rightarrow^n_{\phi^{-1}(R)} \phi^{-1}(t')$ is a loop of length n for $\phi^{-1}(R)$.

Similarly in the opposite direction. □

9. A new example

In this section we present a *two-rule SRS* over a *two-letter alphabet* that has an infinite reduction but no looping reduction.

Proposition 34. *The two-rule SRS*

$$\begin{aligned} 1100101 &\rightarrow 10100101100111 \\ 1101 &\rightarrow 1011 \end{aligned}$$

over the alphabet $\Sigma = \{0, 1\}$ admits an infinite reduction but admits no loop.

Proof. Let R be the two-rule system. R is a bordered SRS with border 1; then $C = 0^*$. Let $\Omega = \{\omega_0, \omega_1, \ldots\}$ and let $f(\omega_i) = 0^i$ for all nonnegative integers i. For $\phi^{-1}(R)$ we get

$$\begin{aligned} \omega_0\omega_2\omega_1 &\rightarrow \omega_1\omega_2\omega_1\omega_0\omega_2\omega_0\omega_0 \\ \omega_0\omega_1 &\rightarrow \omega_1\omega_0 \end{aligned}$$

which is nothing but a renamed version of the SRS from Example 9 by the renaming $\{a \mapsto \omega_2, b \mapsto \omega_0, d \mapsto \omega_1\}$. So we gather that $\phi^{-1}(R)$ does not terminate but admits no loop. By Theorems 32 and 33, respectively, it follows that R, too, does not terminate and admits no loop. □

10. Related work

Our characterization result (Th. 9) is a close relative of the following two results about termination, which were originally stated for restricted term rewriting systems. Guttag *et al.* gave a characterization of cycling. A SRS R is called *quasi-terminating* if every infinite R-reduction enters a cycle, *i.e.* is of the form $t \rightarrow_R^+ t' \rightarrow_R^+ t' \rightarrow_R \cdots$

Theorem 35. *[8] A quasi-terminating SRS R terminates (i.e., admits no cycle) if and only if there is no overlap closure of the form $t \rightarrow_R^+ t$.*

Dershowitz has characterized termination of right-linear term rewriting systems by reductions starting from forward closures (Th. 6 is its string rewriting version). A collection of results about forward closures and termination is presented in Dershowitz and Hoot [5].

Dummy elimination for term rewriting has been introduced by Ferreira and Zantema [6]. They show that dummy introduction preserves non-termination. Zantema and Geser [24] present a technically simpler version for string rewriting. McNaughton [17, 18] calls SRSs where *every* right hand side contains a dummy symbol *inhibitor systems*; he shows that their termination is decidable and their non-termination entails the existence of loops.

For the decidability results below all SRSs are assumed to be finite. It is well-known that termination of SRSs is an undecidable property [9], even for three-rule SRSs [15]. The problem whether termination is undecidable for two-rule SRSs or for one-rule SRSs is still open.

It is easily seen that it is semi-decidable whether a string initiates a loop with respect to some SRS. On the other hand it is co-semi-decidable whether a string initiates an infinite derivation with respect to some SRS. As a consequence, for classes of SRSs in which every infinite derivation contains a loop, for instance non-length-increasing SRSs and inhibitor SRSs, it is decidable whether a string initiates an infinite derivation or not.

Otto [19] has shown that the existence of *proper* loops (*i.e.* loops that are not cycles) is undecidable. Kurth [13] has given a decision procedure for the problem whether a one-rule SRS admits a loop of length 1, 2, or 3. The study of special forms of reductions so far shows only one-rule SRSs that terminate or admit a loop [11, 16, 17]. If there were non-terminating, non-looping one-rule SRSs they were not easy to find: None of the $6.7 * 10^9$ candidates $l \rightarrow r$ where $|r| \leq 9$ is both non-terminating and non-looping [7].

McNaughton [17], having proved that well-behaved SRSs either terminate or admit a loop, conjectures that no non-terminating, non-looping one-rule SRS exists at all. A few related results witness that one-rule SRSs are indeed special:

local confluence and confluence, properties undecidable for SRSs, are decidable for one-rule SRSs [12,22]. To several other problems, such as termination or the word problem, their decidability status for one-rule SRSs is unknown. A proof or a disproof of McNaughton's conjecture were of great help in solving these open questions. In every attempt to falsify McNaughton's conjecture, the central step: the proof that the counterexample does not admit loops, can be done using our method. In particular, as a consequence of our Theorem 33, McNaughton's conjecture is true if and only it is true for all one-rule SRSs that are not bordered.

11. Conclusion

The most general known form of reduction that proves non-termination, *i.e.* the existence of infinite reductions, in a string rewriting system (SRS) is a *loop* $t \to_R^+ utv$.

To obtain proof methods for the non-existence of loops in an SRS, we have examined various known proof methods for termination. We found that it suffices to check the set of *forward closures* against loops. Moreover *dummy elimination* and *dummy introduction* each not only preserves non-termination but also existence of loops of SRSs. A reduction of bordered systems turned out to even preserve existence of loops as well as non-existence of loops.

We illustrated our four new methods in concert at a proof that a *two-rule SRS* over a *two-letter alphabet* exists that admits infinite reductions but no loops.

References

[1] S. Adjan and G. Oganesjan, Problems of equality and divisibility in semigroups with a single defining relation. *Mat. Zametki* **41** (1987) 412–421.

[2] R. Book and F. Otto, *String-rewriting systems.* Texts and Monographs in Computer Science. Springer, New York (1993).

[3] N. Dershowitz, Termination of linear rewriting systems. In *Proc. of the 8th International Colloquium on Automata, Languages and Programming (ICALP81)*, Springer, *Lecture Notes in Computer Science* **115** (1981) 448–458.

[4] N. Dershowitz, Termination of rewriting. *J. Symb. Comput.* **3** (1987) 69–115; *Corrigendum* **4** (1987) 409-410.

[5] N. Dershowitz and C. Hoot, Natural termination. *Theoret. Comput. Sci.* **142** (1995) 179–207.

[6] M. Ferreira and H. Zantema, Dummy elimination: Making termination easier. In *Proc. 10th Conf. Fundamentals of Computation Theory*, H. Reichel, Ed., Springer, *Lecture Notes in Computer Science* **965** (1995) 243–252.

[7] A. Geser, Termination of one-rule string rewriting systems $\ell \to r$ where $|r| \leq 9$. Tech. Rep., Universität Tübingen, D (Jan. 1998).

[8] J.V. Guttag, D. Kapur and D.R. Musser, On proving uniform termination and restricted termination of rewriting systems. *SIAM J. Comput.* **12** (1983) 189–214.

[9] G. Huet and D.S. Lankford, On the uniform halting problem for term rewriting systems. Rapport Laboria 283, INRIA (1978).

[10] M. Jantzen, *Confluent string rewriting*, Vol. 14 of *EATCS Monographs on Theoretical Computer Science.* Springer, Berlin (1988).

[11] Y. Kobayashi, M. Katsura and K. Shikishima-Tsuji, Termination and derivational complexity of confluent one-rule string rewriting systems. Tech. Rep., Dept. of Computer Science, Toho University, JP (1997).

[12] W. Kurth, *Termination und Konfluenz von Semi-Thue-Systemen mit nur einer Regel.* Dissertation, Technische Universität Clausthal, Germany (1990).

[13] W. Kurth, One-rule semi-Thue systems with loops of length one, two, or three. *RAIRO Theoret. Informatics Appl.* **30** (1995) 415–429.

[14] D.S. Lankford and D.R. Musser, A finite termination criterion. Tech. Rep., Information Sciences Institute, Univ. of Southern California, Marina-del-Rey, CA (1978).

[15] Y. Matiyasevitch and G. Sénizergues, Decision problems for semi-Thue systems with a few rules. In *IEEE Symp. Logic in Computer Science'96* (1996).

[16] R. McNaughton, The uniform halting problem for one-rule Semi-Thue Systems. Tech. Rep. 94-18, Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, Aug. 1994. See also "Correction to The Uniform Halting Problem for One-rule Semi-Thue Systems", personal communication (Aug. 1996).

[17] R. McNaughton, Well-behaved derivations in one-rule Semi-Thue Systems. Tech. Rep. 95-15, Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY (Nov. 1995).

[18] R. McNaughton, Semi-Thue Systems with an inhibitor. Tech. Rep. 97-5, Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY (1997).

[19] F. Otto, The undecidability of self-embedding for finite semi-Thue and Thue systems. *Theoret. Comput. Sci.* **47** (1986) 225–232.

[20] B.K. Rosen, Tree-manipulating systems and Church-Rosser Theorems. *J. ACM* **20** (1973) 160–187.

[21] K. Shikishima-Tsuji, M. Katsura and Y. Kobayashi, On termination of confluent one-rule string rewriting systems. *Inform. Process. Lett.* **61** (1997) 91–96.

[22] C. Wrathall, Confluence of one-rule Thue systems. In *Word Equations and Related Topics*, K.U. Schulz, Ed., Springer, *Lecture Notes in Computer Science* **572** (1991).

[23] H. Zantema and A. Geser, A complete characterization of termination of $0^p1^q \to 1^r0^s$. *Applicable Algebra in Engineering, Communication, and Computing*. In print.

[24] H. Zantema and A. Geser, A complete characterization of termination of $0^p1^q \to 1^r0^s$. In *Proc. of the 6th Conference on Rewriting Techniques and Applications*, J. Hsiang, Ed., Springer, *Lecture Notes in Computer Science* **914** (1995) 41–55.

Communicated by Ch. Choffrut.
Received October, 1998. Accepted Juin, 1999.