Non-Looping Rewriting

Hans ZANTEMA[1] and Alfons GESER[2]

[1] Dept. of Computer Science, Universiteit Utrecht, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands, E-mail: hansz@cs.ruu.nl

[2] Lehrstuhl für Programmiersysteme, Universität Passau,
94030 Passau, Germany, E-mail: geser@fmi.uni-passau.de

Abstract

In this paper we present a number of *necessary* conditions for the existence of *loops*, i.e. reductions of the form $t \to_R^+ c[t\sigma]$. We investigate which of the known termination preserving transformation methods also preserve the non-existence of loops. We characterize the existence of loops by overlap closures. We illustrate these methods at new examples of a *one-rule term rewriting system* and a *two-rule string rewriting system* which admit a non-terminating reduction but no loop.

1 Introduction

Term rewriting systems (TRS, for short) are a convenient means for reasoning about equations and algorithms. A TRS whose rewriting relation admits no infinite reductions is called a *terminating* TRS. Terminating TRSs enjoy a number of favourable properties without being too restrictive.

Infinite reductions are often composed of cycles. A *cycle* is a reduction where a term is rewritten to the same term. More generally, a *loop* is a reduction where the starting term re-occurs, with its variables substituted, in a context. 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 non-terminating, non-looping reductions do exist. Apparently, Dershowitz [7] was the first to distinguish looping from non-termination.

Although the notion of loop is fairly common, we have heard of no sufficient conditions to deduce that a TRS is non-terminating, non-looping. This is strange since we can fancy a number of applications.

For instance in the Knuth/Bendix completion procedure, it is favourable to exclude non-terminating TRSs from consideration. It is common practice to do it by restricting to provably terminating TRSs. But then, the procedure may overlook terminating candidates as soon as the termination ordering is too weak. Purdom [23] prefers to exclude looping rewrite systems. Then no terminating candidate is lost, and so the procedure is less likely to fail or diverge. This is paid, however, with an enormous increase in complexity, caused by branching and the absence of normalization. In this respect, means to exclude as many non-terminating systems as possible are invaluable.

Another application is the question how expressive one-rule string rewriting systems (SRS) are. It is unknown whether one-rule SRSs can simulate Turing machines, and whether their termination is undecidable. McNaughton [18] conjectures that no non-terminating, non-looping one-rule SRS exists. If the conjecture were true, it would be a clear indication against Turing power of one-rule SRSs.

A *termination preserving transformation* is a binary relation $\hookrightarrow$ on TRSs such that $R \hookrightarrow R'$ and R terminating implies R' terminating. It is natural to ask whether "terminating" may be replaced by "non-looping". We answer this question for the following transformations.

1. transformation ordering [2, 1, 3], and specifically, dummy introduction [26],

2. dummy elimination [10],

3. semantic labelling [25],

Moreover we characterize the existence of loops by the existence of looping overlap closures.

We illustrate some of the methods at new examples. Specifically, we give witnesses for the facts that *two-rule SRSs* and *one-rule TRSs* exist that are non-terminating but admit no loop. We conclude the paper with an overview of decidability results connected with looping.

2 Basic Notions

We assume that the reader is familiar with termination of term rewriting [7]. We will use notation as in Dershowitz/Jouannaud [9].

A *position* in a term is the path, given in Dewey decimal notation, from the top of the term downwards. The subterm of t at position p is denoted $t|_p$. If $t|_p$ is defined then p is called a position of t. The term t, with its subterm at p replaced by term t' is denoted $t[t']_p$.

A *cycle* is a reduction of the form $t \to_R^+ t$; a *loop* is a reduction of the form $t \to_R^+ c[t\sigma]_p$ where c is a term, p a position in c, and σ a substitution. A loop can be *composed* with itself to form a larger loop: $t \to_R^+ c[t\sigma]_p \to_R^+ c[c[t\sigma]_p\sigma]_p = c'[t\sigma']_{p'}$ where $c' = c[c\sigma]_p$, $\sigma' = \sigma\sigma$, and $p' = p.p$. Composition can be iterated finitely or infinitely.

We say that a term t *cycles* if there is a reduction $t \to_R^* t' \to_R^+ t'$ ending in a cycle. Likewise, t *loops* if a reduction ending in a loop, $t \to_R^* t' \to_R^+ c[t'\sigma]_p$, exists. A TRS R is said to *admit a loop* if a looping t exists.

We will treat SRSs as TRSs where we identify letters with function symbols of arity one. SRSs are also called *semi-Thue systems*. A loop in an SRS is then of the form $t \to_R^+ utv$ where t, 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*.

3 A Basic Example

In this section we present a three-rule SRS to illustrate the typical aspects of non-terminating, non-looping rewriting. In Section 8 a one-rule TRS and a two-rule SRS with the same property will be given.

Example 1. The three-rule SRS R given by

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

admits an infinite reduction, but no loop.

The same system already appears in Dershowitz's survey [7], p. 111, however without a proof that it admits no loop. A slightly more complicated example of a three-rule SRS with the same property was given by Kurth [15]. 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 TRS 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 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.

In the next few sections we investigate which transformations preserve non-looping TRSs.

4 Transformation Ordering

The idea of *transformation order* arose from the observation that monotonic interpretations of terms as (other) terms may be conveniently modelled as unique normal forms w.r.t. a confluent, terminating TRS T. To ensure that the defined order is closed under substitution and contexts, a commutation property is required which can be localized to a property called *local cooperation.*

Theorem 1 (Transformation order, [2, 1, 3]). *Let $\rightarrow_S$ and $\rightarrow_T$ be relations on a given set. If*

1. $\rightarrow_S \cup \rightarrow_T$ *terminates,*
2. $\rightarrow_T$ *is confluent,*

3. $\to_T$ locally cooperates *with* $\to_S$*, i.e.* $\leftarrow_T \to_S \subseteq (\to_S / \to_T)^+ \leftarrow_T^*$*,*

then $> =_{\mathrm{def}} (\to_S / \leftrightarrow_T)^+ \cup \to_T^+$ *terminates.*

For TRSs S and T, such that $\to_S$ and $\to_T$ are their respective rewrite relations, Condition (2) follows by local confluence of critical pairs, and Condition (3) follows from

3i. T is non-erasing and left-linear, and

3ii. $CP(T, S) \subseteq (\to_S / \to_T)^+ \leftarrow_T^*$,

which are effectively checkable. Moreover, termination of $\to_R$ for a TRS R follows from $R \subseteq >$. We have developed another version which disposes of confluence and termination of T.

Theorem 2 ([26]). *Let* $\to_S$*,* $\to_T$*, and* $\to_R$ *be binary relations on a given set. If*

1. $\to_S$ *terminates,*

2. $\to_R \subseteq \to_S^+ \leftarrow_T^*$*,*

3. $\leftarrow_T \to_R \subseteq \to_R^+ \leftarrow_T^*$*,*

then $\to_R$ *terminates.*

If R, S, and T are TRSs, such that $\to_R$, $\to_S$ and $\to_T$ are their respective rewrite relations, for Condition (2) it is sufficient that $R \subseteq \to_S^+ \leftarrow_T^*$, and for Condition (3) that

3i. R is left-linear,

3ii. T left-linear and non-erasing,

3iii. $CP(T, R) \subseteq \to_R^+ \leftarrow_T^*$.

Thm. 2, akin to a commutation result of Bachmair and Dershowitz [1], is particularly suited to SRSs which are always left-linear, right-linear, and non-erasing.

Both kinds of transformation order fail to to preserve non-looping reductions as the following counterexample demonstrates.

Example 2. Let R, S, and T be the TRSs

$$\begin{array}{lll} bc \to dc & bc \to dc & \\ bd \to db & bd \to db & pa \to ab \\ ad \to pab & ad \to abb & \end{array}$$

respectively. R has a loop

$$adc \to abc \to adc$$

but S has no loop (see Ex. 1). Condition (2) of Thm. 2 holds by $ad \to_S abb \leftarrow_T pab$. Condition (3iii) holds because the only overlap $abd \leftarrow_T pad \to_R ppab$ satisfies $abd \to_R adb \to_R pabb \leftarrow_T ppab$. Hence Thm. 2 is not correctly applicable to infer non-loopingness.

By a simple form of completion, an infinite system $R =_{\mathrm{def}} \{ab^n d \to pab^{n+1} \mid n > 0\}$ can be constructed such that $\leftarrow_T \to_R \subseteq \to_R \leftarrow_T$ holds although T is terminating and confluent. At present, we do not know whether a *finite* counterexample with these properties exists.

4.1 Dummy Introduction

Dummy introduction is a special case of our form of transformation order. A *dummy* is a symbol $\Box$ that occurs only at right hand sides, whence it acts as a barrier for rewrite redexes. To introduce a dummy means to restrict rewriting. A dummy introduction rule is of the form

$$vdw \to v\Box w$$

with the intended meaning that within the context v and w the (potentially empty) substring d is *dead*; none of its positions will ever be touched in a reduction. So it may just as well be replaced by the dummy. If unrestricted reductions can be transformed to restricted ones, then termination of the original system follows from termination of the restricted system. Dead substrings are indeed never rewritten.

Definition 3 (Left Overlap). A string l is said to *left overlap* a string r is there are strings l', r' and x where $l = l'x$, $r = xr'$, and x is nonempty. Likewise, r is called to *right overlap* l in this case.

Definition 4 (Dummy Introduction). Let $\mathcal{A}$ be a alphabet where $\Box \notin \mathcal{A}$ and let R be a SRS over $\mathcal{A}$. Let d be a string and let C, D be nonempty sets of strings over $\mathcal{A}$ such that d is nonempty if the empty string is in C and in D. Let T be the SRS

$$T =_{\mathrm{def}} \{udv \to u\Box v \mid u \in C \wedge v \in D\} \ .$$

Then T is called a *dummy introduction* for R if

1. every overlap of a left hand side l of a rewrite rule $l \to r$ in R with a left hand side udv of a rule in T already is an overlap of l with $u\Box v$, i.e. is either a left overlap with u or a factor of u or a right overlap with v or a factor of v; and

2. for each left overlap of $l = l'x$, $u = xu'$ of l with u, a suffix of ru' is in C,

3. for each $u = u'lu''$, a suffix of $u'ru''$ is in C, and symmetrically,

4. for each right overlap of $l = yl''$, $v = v''y$ of l with v, a prefix of $v''r$ is in D,

5. for each $v = v'lv''$, a prefix of $v'rv''$ is in D.

Particularly it follows that the left hand sides of R are in T-normal form, and that T terminates.[1] Let $T(R)$ be the T-normal SRS

$$T(R) =_{\mathrm{def}} \{l \to r' \mid (l \to r) \in R \wedge r \to^!_T r'\}$$

Recall that T need not be confluent; so one has to consider all possible T-normal forms r' of r. It is routine to prove the following technical lemma.

Lemma 5. *If T is a dummy introduction for R, a rule $(l \to r) \in R$, a suffix of u is in C, a prefix of v is in D, then*

$$\begin{tikzcd} udv \arrow[r, "l\to r"'] \arrow[d, "T"] & u'dv' \arrow[d, "T"] \\ u\Box v \arrow[r, "l\to r"'] & u'\Box v' \end{tikzcd}$$

holds where a suffix of u' is in C and a prefix of v' is in D.

[1] If d is empty then use dummy elimination (Thm. 11) for T to prove its termination.

Moreover, one observes that each position within the box in $u\boxed{d}v$ is not touched; it has its unique residual in $u'\boxed{d}v'$. Another observation is that l is a factor of u or of v. We will need these, and the following result below.

Lemma 6. *Let T be a dummy introduction for the SRS R. Then T is a dummy introduction for the SRS*

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

Proof. Let T be a dummy introduction for R. We check against the conditions for dummy introduction of T for the overlap closure $s \to_R^n t$.

Condition (1): Let s overlap with udv. No position in the box in $u\boxed{d}v$ (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, by Lemma 19. Hence every overlap of t with udv is either a left overlap with u, or a factor of u, or a right overlap with v, or a factor of v.

Condition (2), (3), (4) and (5): By induction on n from Lemma 5. □

Dummy introduction is a special case of transformation order. This is implicit in [26].

Proposition 7. *Let R be an SRS, let T be a dummy introduction for R, and let $S = T(R)$. Then R, S, T satisfy the conditions of transformation order.*

Proof. We have to establish Conditions (2) and (3) of Thm. 2. From the definition of $T(R)$ it follows immediately that $R \subseteq \to_{T(R)}^+ \leftarrow_T^*$ holds. The commutation $\leftarrow_T \to_R \subseteq \to_R \leftarrow_T$ is stated in Lemma 5. □

Surprisingly, dummy introduction preserves non-looping SRSs.

Theorem 8. *Let R be an SRS and let T be a dummy introduction for it. Then $\to_{T(R)}$ admits a loop if $\to_R$ admits a loop.*

Proof. By straightforward induction, one can prove that

$$\leftarrow_T^m \to_R^n \subseteq \to_R^n \leftarrow_T^m \tag{4}$$

for all m and n. By induction on n, one can then establish that

$$\to_R^n \subseteq \to_S^+ \leftarrow_T^* \tag{5}$$

holds for all n. For $n = 1$ this follows from the premise $R \subseteq \to_S^+ \leftarrow_T^*$ by the definition of $\to_R$. For $n > 1$, we have the reasoning

$$\to_R^n \underset{\text{premise}}{\subseteq} \to_S^+ \leftarrow_T^* \to_R^{n-1} \underset{(4)}{\subseteq} \to_S^+ \to_R^{n-1} \leftarrow_T^* \underset{\text{IH},n-1}{\subseteq} \to_S^+ \to_S^+ \leftarrow_T^* \leftarrow_T^* \tag{6}$$

Assume now that R admits a loop $t \to_R^+ ptq$ where $t, p, q \in \mathcal{A}^*$. According to Thm. 22, we may assume that this is an overlap closure. By (5) 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. By Lemma 6 and Lemma 5, we have a commuting diagram

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

Since t'' admits a rewrite step for rule $t \to ptq$ in the TRS formed by overlap closures, it follows that t'' contains t as a factor.

So every term in the reduction $ptq \to_T^k$ 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. $\square$

Example 3. The SRS

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

contains the dead part $abbb$ in rule (3). To prove that it does not loop, choose T the one-rule system $abbba \to \square a$. It is easily verified that T is a dummy introduction.

5 Dummy Elimination

Dummy elimination is somewhat the counterpart to dummy introduction. The two steps together form a method to split right hand sides of rules, the intention being that the pieces are easier to handle. In the introduction step one puts a mark for the dead part in a right hand side of a rule; in the elimination step the right hand side is broken up at the marked position. Ferreira and Zantema [10] introduce a version for term rewriting; we restrict ourselves here to a technically simpler version for string rewriting, as in our RTA-95 paper [26].

Definition 9. 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 $i \in \{1, \ldots, n\}$, let $\mathcal{E}(s) =_{\text{def}} \{r_1, \ldots, r_n\}$.

Definition 10 ([26]). Let R be an SRS on the alphabet $\mathcal{A}$ where the symbol $\Box \in \mathcal{A}$ does not occur on left hand sides of R. Let

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

Dummy elimination preserves non-terminating SRSs.

Theorem 11 ([26]). *Let R be a SRS. If $\to_{E(R)}$ terminates then $\to_R$ terminates.*

It preserves looping as well, as we show next. We conjecture that the result extends to proper TRSs R.

Theorem 12. *Let R be a SRS. If $\to_R$ admits loops then $\to_{E(R)}$ admits loops.*

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

Definition 13. For an SRS R, a relation $>_R$ on strings is defined by $v >_R w$ if there exist q, q' such that $v \to_R^+ qwq'$.

Proposition 14. *$>_R$ is transitive. R admits no looping reductions if and only if $>_R$ is irreflexive.*

of Thm. 12. Let R be an SRS, and let $S = E(R)$.

We claim that $s \to_R t$ implies $\mathcal{E}(s) >_S^{mult} \mathcal{E}(t)$. Suppose $s \to_R t$ using rule $l \to r$ 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 $\Box$, whence $\mathcal{E}(s) = \{s_1 l s_2\}$. If r does not contain $\Box$, then $\mathcal{E}(t) = \{s_1 r s_2\}$, and the claim follows by $s_1 l s_2 \to_S s_1 r s_2$, as rule $l \to r$ is also in S. Else, suppose that $r = r_1 \Box r_2 \Box \cdots \Box r_n$ 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 defintion $s_1 l s_2$ 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 $>_S$ irreflexive; then $>_S^{mult}$ is irreflexive; then $>_R$ is irreflexive; so R admits no loop. $\Box$

The following example shows a subtle property of dummy elimination, by which the reversed directions in Thm. 11 and Thm. 12 do not hold in general.

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

$$gff \to_{E(R)} ggf \to_{E(R)} gfff$$

In contrast, R terminates, a fact that can be proven by transformation order. Choose the one-rule system $T = \{gg\Box fff \to \Box'\}$ where $\Box'$ is a new dummy symbol. There are no critical pairs between T and R. The transformed system $gf \to \Box'$ trivially terminates.

6 Semantic Labelling

The purpose of *semantic labelling* [25] is to collect global information of a term, and to deposit it as a label where it can be accessed locally. Local termination proof methods such as precedence-based path orders or monotonic interpretations may greatly profit from such a step.

Let $\mathcal{F}$ be a set of function symbols, each having fixed arity ≥ 0. A $\mathcal{F}$-algebra $\mathcal{M}$ is a set M, together with for every $f \in \mathcal{F}$ of arity n a function $[f] : M^n \to M$. It defines an evaluation homomorphism $[\_] : \mathcal{T}(\mathcal{F}, \mathcal{X}) \times (\mathcal{X} \to M) \to M$ inductively by

$$[x](\beta) = \beta(x),$$
$$[f(t_1, \ldots, t_n)](\beta) = [f]([t_1](\beta), \ldots, [t_n](\beta))$$

where $\beta : \mathcal{X} \to M$, $x \in \mathcal{X}$, $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. $\mathcal{M}$ is called a *model* of a TRS R if $[l](\beta) = [r](\beta)$ holds for all $\beta : \mathcal{X} \to M$ and $l \to r$ in R.

A *labelling* $\mathcal{L}$ is defined as follows. Choose for each function symbol f a set S_f of labels. A new set $\bar{\mathcal{F}}$ of labelled function symbols is defined by

$$\bar{\mathcal{F}} = \{f_s \mid f \in \mathcal{F} \wedge s \in S_f\}$$

If $|S_f| = 1$ then f_s and f may be confused, and f_s may be called unlabelled. Choose for every $f \in \mathcal{F}$ of arity n a map $\pi_f : M^n \to S_f$. This induces a *labelling function* $\mathsf{lab} : \mathcal{T}(\mathcal{F}, \mathcal{X}) \times (\mathcal{X} \to M) \to \mathcal{T}(\bar{\mathcal{F}}, \mathcal{X})$, by

$$\begin{aligned} \mathsf{lab}(x, \beta) &= x, \\ \mathsf{lab}(f(t_1, \dots, t_n), \beta) &= f_d(\mathsf{lab}(t_1, \beta), \dots, \mathsf{lab}(t_n, \beta)) \end{aligned}$$

where $\beta : \mathcal{X} \to M$, $x \in \mathcal{X}$, $t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and $d = \pi_f([t_1](\beta), \dots, [t_n](\beta))$. The labelled TRS $L(R)$ over $\mathcal{T}(\bar{\mathcal{F}}, \mathcal{X})$ is defined as follows.

$$L(R) = \{\mathsf{lab}(l, \beta) \to \mathsf{lab}(r, \beta) \mid \beta : \mathcal{X} \to M \wedge (l \to r) \in R\}$$

Theorem 15 ([25]). *Let $\mathcal{M}$ be a model for a TRS R over $\mathcal{F}$, and let $\mathcal{L}$ be a labelling for $\mathcal{F}$. Then $\to_R$ terminates if and only if $\to_{L(R)}$ terminates.*

Removal of labels trivially preserves loops. The next question whether an according preservation holds for non-looping, is answered negatively by the following example.

Example 5. The one-rule SRS $1 \to 10$ is trivially looping. Choose as a model the nonnegative integers with $[1](x) = 0$ and $[0](x) = x + 1$. Label 1 by $\pi_1(x) = x$. The 0 symbol remains unlabelled. The labelled system $L(R)$ consists of rules

$$1_i \to 1_{i+1}0$$

for each nonnegative integer i. The labelled system does not admit a loop, as we show next. For the purpose of contradiction, assume $\bar{t} \to^+_{L(R)} \bar{u}\bar{t}\bar{v}$ were a loop where $\bar{t}$, $\bar{u}$, and $\bar{v}$ are terms in the labelled signature. Since the number of labelled 1 symbols never changes during reduction, no labelled 1 symbol occurs in $\bar{u}$ or $\bar{v}$. So the sum of all labels in $\bar{u}\bar{t}\bar{v}$ is the same as in $\bar{t}$. On the other hand, the sum of all labels of a string increases by one, along each reduction step. Contradiction.

For the example to work it was essential that the set of labels is infinite. Indeed, we have:

Theorem 16. *Let $\mathcal{M}$ be a* finite *model for a TRS R over $\mathcal{F}$, and let $\mathcal{L}$ be a labelling for $\mathcal{F}$. Then $\to_{L(R)}$ admits a loop if and only if $\to_R$ admits a loop.*

Proof. Given a loop $t \to_R^+ c[t\sigma]_p$, we have to construct a loop in the labelled system. There are only finitely many possible configurations of labels for the symbols of t. By the pigeonhole principle there is n so large that in a succession of n configurations the same configuration appears at least twice. We compose the given loop n times, and label this reduction. The section of the reduction between two equal configurations forms a loop. □

7 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 by means of overlap closures.

Definition 17 (Overlap Closure, [12]). Let a TRS R be given. The set $\mathrm{OC}(R)$ of *overlap closures* is the smallest set of R-reductions, closed under bijective renaming of variables, such that

oc1. if $(l \to r) \in R$ then $(l \to_R r) \in \mathrm{OC}(R)$,

oc2. if $(s_1 \to_R^+ t_1) \in \mathrm{OC}(R)$ and $(s_2 \to_R^+ t_2) \in \mathrm{OC}(R)$ have disjoint sets of variables, and there is a position p in t_1 such that $t_1|_p$ is not a variable, and $t_1|_p$, s_2 are unifiable, then

$$s_1\sigma \to_R^+ t_1\sigma \to_R^+ t_1[t_2]_p\sigma$$

is in $\mathrm{OC}(R)$, where σ is the most general unifier of $t_1|_p$ and s_2.

oc3. if $(s_1 \to_R^+ t_1) \in \mathrm{OC}(R)$ and $(s_2 \to_R^+ t_2) \in \mathrm{OC}(R)$ have disjoint sets of variables, and there is a position p in s_2 such that $s_2|_p$ is not a variable, and t_1, $s_2|_p$ are unifiable, then

$$s_2[s_1]_p\sigma \to_R^+ s_2\sigma \to_R^+ t_2\sigma$$

is in $\mathrm{OC}(R)$, where σ is the most general unifier of t_1 and $s_2|_p$.

Lemma 18 ([12]). *If R is left-linear then* $\mathrm{OC}(R)$ *is left-linear.*

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 [24] 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.

Call a position p in a term t *touched by the rewrite step* $t \xrightarrow[l\to r]{u} t'$ if p is of the form $p = u.v$ where v is an inner position in l. A position p' in t' is called a *residual* of p in t by the rewrite step $t \xrightarrow[l\to r]{u} t'$ if $p = p'$ is at or above u, or $p = u.v.w$, $p' = u.v'.w$ where $l|_v = r|_{v'}$ is a variable. The residual relation is inductively extended to reductions: A position p' in t' is called a *residual* of p in t by the reduction $t \to_R^n t'$ if $n = 0$ and $p = p'$ or $t \to_R t'' \to_R^{n-1} t'$ and there is a residual p'' in t'' of p such that p' in t' is a residual of $t'' \to_R^{n-1} t'$. Now a position p 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 a residual p'' in t'' of p by $t \to_R^* t''$ is touched in the step $t'' \to_R t'''$.

Thus we can characterize overlap closures. We call a position an *inner* position of t if it is not the top and not a variable position in t.

Lemma 19. *If R is a left-linear TRS, 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. 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 $s_1\sigma$. Either u is a non-variable position in s_1, or it has a residual $p.u'$ in $t_1\sigma$ by $s_1\sigma \to_R^+ t_1\sigma$, in which case u' is a non-variable position in s_2. The claim follows by inductive hypothesis for $s_1 \to_R^+ t_1$ in the first case, and for $s_2 \to_R^+ t_2$ in the latter case.

Case (oc3): Let u be an inner position in $s_2[s_1]_p\sigma$. If u is of the form $p.u'$ where u' is a inner position of s_1, then u is touched during $s_1 \to_R^+ t_1$ by inductive hypothesis. If u is of the form $p.u'$ where u' is a position in the substitution part of $s_1\sigma$, then u has a residual $p.u''$ in $s_2[t_1]_p\sigma = s_2\sigma$ which is a non-variable position in s_2. Likewise if u is not of the form $p.u'$ for non-empty u' then u has residual u by $s_2[s_1]_p\sigma \rightarrow^+_R s_2\sigma$, and u is a non-variable position in s_2. In both cases the claim follows by inductive hypothesis for $s_2 \rightarrow^+_R t_2$. This finishes the first part.

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

Now let $n > 1$, and let the reduction be $t \rightarrow^{n-1}_R t'' \xrightarrow[l\to r]{u} t'$. Let P be the set of inner positions of t untouched during $t \rightarrow^{n-1}_R t''$. Assign each $p \in P$ a unique variable y_p not used in the reduction. Now the reduction $t \rightarrow^{n-1}_R t''$ can be split into $|P| + 1$ reductions

$$(t|_p)[y_{p_1}]_{p_1}[y_{p_2}]_{p_2} \cdots [y_{p_m}]_{p_m} \rightarrow^*_R (t''|_{p'})[y_{p_1}]_{p'_1}[y_{p_2}]_{p'_2} \cdots [y_{p_m}]_{p'_m} \qquad (4)$$

where $p \in P$ or p is the top position, and $p_1, \dots, p_m \in P$ are the topmost positions below p. For each p the (nonempty!) set of its residuals by $t \rightarrow^{n-1}_R t''$ is denoted by p'. The expression $t[y]_{p'}$ is to denote simultaneous replacement of each subterm at a position in p' by y. The expression $t''|_{p'}$ is to denote t'' at some position in p'; this is well-defined since the term t'' has the same subterm at each position in p'.

By construction, every inner position of $(t|_p)[y_{p_1}]_{p_1}[y_{p_2}]_{p_2} \cdots [y_{p_m}]_{p_m}$ is touched during (4). Every nonempty reduction of these is an overlap closure by inductive hypothesis. At least one of these reductions is non-empty.

Together with the rule $l \rightarrow r$ we have a set of at least two overlap closures. Recall that each position in P has at least one residual that is touched by $t \xrightarrow[l\to r]{u} t'$. So the nonempty reductions (4) may be sticked together, one after the other, with $l \rightarrow r$ to yield an overlap closure by Case (oc3). □

Let us review related work. Geupel characterized termination of left-linear TRSs.

Theorem 20 ([11]). *If a left-linear TRS does not terminate, then some right hand side of an overlap closure issues an infinite reduction.*

Guttag, Kapur, and Musser gave a characterization of cycling. A TRS is called *quasi-terminating* if every reduction is either finite or ends in a cycle.

Theorem 21 ([12]). *A left-linear, quasi-terminating TRS terminates (i.e., admits no cycle), if and only if there is no reflexive overlap closure.*

As Zhang [27] demonstrates, the left-linearity condition in both theorems is essential. It is now natural to seek a similar characterization for looping. We can offer a characterization result for SRSs. We will call an overlap closure of the form $t \to_R^+ ptq$ a *looping overlap closure.*

Theorem 22. *An SRS admits a loop if and only if it has a looping overlap closure.*

Proof. Let R be an SRS. "If" is trivial; we have to prove "only if".

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 19. 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,

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

such that

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

holds. Case 1: t_1' is longer than ut_1. Then there is v' such that $t_1' = ut_1v'$, and the reduction $t_1 \to_R^* t_1'$ is nonempty. So $t_1 \to_R^+ ut_1v'$ is another loop, with less untouched positions in t_1. Case 2: t_1' is shorter than ut_1. Then t_2' must be longer than t_2v, which is solved by a symmetric reasoning. Case 3: $t_1' = ut_1$ and $t_2' = t_2v$ hold. One of the reductions must be nonempty, as $t \to_R^+ utv$ is nonempty. This reduction may be used as the wanted loop. □

Example 6. We can give an alternative proof that the three-rule SRS R in Ex. 1 admits no loops, by showing that it has no looping overlap closures.

Overlap closures are of the following forms, where $w, w' \in \{b, d\}^*$.

$$\begin{array}{rcl} bwd \to_R^+ dw'b & \text{if} & w \to_R^* w' \\ bwc \to_R^+ dw'c & \text{if} & wc \to_R^* w'c \\ awd \to_R^+ aw'b & \text{if} & abw \to_R^* aw' \end{array}$$

and some overlap closures of the form $awc \to_R^+ aw'c$ where w is strictly shorter than w'. Not all of the latter are overlap closures, but this is not essential for our argument. It is easy to show that none of these reductions can be looping, as the reductions in the "if" part can be done with a terminating subset of the rules. So there is no looping overlap closure. By Thm. 22 it follows that R admits no loop.

Example 7. Let S be as R above where c has been identified with a.

$$ba \to da \tag{5}$$

$$bd \to db \tag{6}$$

$$ad \to abb \tag{7}$$

It has no looping overlap closures either, as we are going to show. By the identification, the set of overlap closures of R (with a instead of c) extend by some reductions of the form

$$u \to_R^+ u' \qquad bwu \to_R^+ dw'u', \qquad uw''d \to_R^+ u'w'''b, \qquad bwuw''d \to_R^+ dw'u'w'''b$$

where $w, w', w'', w''', w_1, w_2, \ldots \in \{b, d\}^*$, and

$$u = aw_1aw_2 \cdots aw_na,$$
$$u' = aw_1'aw_2' \cdots aw_n'a$$

for some n[2] are such that for every i

$$bwa \to_R^* dw'a, \qquad aw''d \to_R^* aw'''b, \qquad aw_ia \to_R^* aw_i'a \tag{8}$$

are overlap closures or empty.

Because the number of a symbols is not changed by rewriting, any looping overlap closure must break down to looping overlap closures of the form (8). As we have argued in the previous example these are non-looping. So there cannot be looping overlap closures in S either; so S admits no loops.

[2]including the case $n = 0$ where $u = a = u'$

It is an open question whether in Thm. 22 “SRS” may be replaced by “left-linear TRS”, or whether “overlap closure” may be replaced by “forward closure”. A collection of results about forward closures and termination is presented in Dershowitz and Hoot [8].

8 New Examples

In this section we present a *one-rule TRS* and a *two-rule SRS*, each having an infinite reduction but no looping reduction.

Example 8. The one-rule TRS R given by

$$f(c, a(x), y) \;\to\; g(f(c, x, a(y)), f(x, y, a(a(c))))$$

admits an infinite reduction, but no loop.

Let S consist of the two rules

$$f(c, a(x), y) \to f(c, x, a(y)) \qquad (9)$$
$$f(c, a(x), y) \to f(x, y, a(a(c))), \qquad (10)$$

then S admits for every $i > 0$ a reduction

$$f(c, a(c), a^i(c)) \to_S f(c, a^i(c), a(a(c))) \to_S^{i-1} f(c, a(c), a^{i+1}(c))$$

yielding the infinite non-looping reduction

$$f(c, a(c), a(c)) \to_S^+ f(c, a(c), a(a(c))) \to_S^+ f(c, a(c), a(a(a(c)))) \to_S^+ \cdots$$

Since in this infinite S-reduction all reduction steps are in the root, it easily extends to an infinite R-reduction.

For proving non-loopingness of R we first prove non-loopingness of S. Assume S admits a loop $t \to_S^+ c[t\sigma]$. Since no other symbols than a can be created during reduction, the context c only consists of a's; since no a can be created outside the outermost f-symbol we conclude $t \to_S^+ t\sigma$. Taking a reduction of this shape of minimal depth we may assume that $t = f(t_1, t_2, t_3)$ and that the reduction contains steps at the root level.

If only Rule (9) is applied in a root reduction step then the decrease of the second argument of f leads to a contradiction. Hence the reduction contains at least one step by Rule (10). We may even assume that it contains at least two; else, we compose the given loop with itself. Hence our reduction is of the shape

$$\underbrace{f(t_1,t_2,t_3) \to_S^* f(c,a(u),v)}_{\text{no root reduction steps of Rule (10)}} \to_S \underbrace{f(u,v,a(a(c))) \to_S^* f(c,a(r),s)}_{\text{no root reduction steps of Rule (10)}} \to_S$$
$$f(r,s,a(a(c))) \to_S^* f(t_1\sigma,t_2\sigma,t_3\sigma) \;.$$

We claim that then t_1, t_2, and t_3 are ground. From the first part of this reduction follows $t_1 \to_S^* c$, hence $t_1 = c$. By applicability of an S-rewrite step to $f(u,v,a(a(c)))$ we have $u = c$. Looking at the second argument gives $t_2 \to_S^* a^i(u) = a^i(c)$ for some $i > 0$, hence $t_2 = a^i(c)$. Finally either the reduction $f(r,s,a(a(c))) \to_S^* f(t_1\sigma,t_2\sigma,t_3\sigma)$ contains no steps at the top; in this case $r \to_S^* t_1\sigma = c$ and so $r = c$ holds. Or $r = c$ by applicability of a top S-rewrite step in this reduction. In both cases $v = a^{j+1}(r) = a^{j+1}(c)$ follows. By $a^i(t_3) \to_S^* a(v) = a^{j+2}(c)$ one gets $t_3 = a^{j+2-i}(c)$. So t_1, t_2, t_3, and hence also t, are ground. This way the whole reduction only contains ground terms, hence $t \to_S^+ t$. Since Rule (9) is terminating and preserves size and Rule (10) increases size, this is a contradiction. Hence S is non-looping.

Non-loopingness of R now follows by the following proposition.

Proposition 23. *Let l, r, r' be linear terms not containing the symbol g. If the one-rule TRS $l \to g(r,r')$ admits a loop, then the two-rule TRS $l \to r$, $l \to r'$ admits a loop, too.*

Proof. Let R and S be the one-rule and two-rule TRS, respectively, and let a loop $t \to_R^+ c[t\sigma]_p = u$ be given. We may assume that t does not contain the symbol g.

For, assume t contains g at a position p, i.e. $t = t[g(t_1,t_2)]_q$. Since g symbols are not touched during the derivation, every term in the derivation may be split in the same way at the respective residual position of q. Thus the derivation may be decomposed into three parts

$$t[z]_q \to_R^* u[z]_{q'}, \qquad t_1 \to_R^* u_1, \qquad t_2 \to_R^* u_2$$

where z is any variable, and $u[g(u_1,u_2)]_{q'} = c[t\sigma]_p$, and q' is the residual of q. Either q' is at or below $p.q.1$, in which case $t_1 \to_R^* u_1$ is a loop. Or q' is at or below $p.q.2$, in which case $t_2 \to_R^+ u_2$ is a loop. Or $q' = p.q$ in which case any nonempty of the three derivations forms a loop. Else $t[c]_q \to_R^+ u[c]_{q'}$ is a loop. In each case, the starting term has a smaller number of g symbols.

By induction on the number of g symbols in the starting term, we conclude that we get a loop with no g symbol in the starting term. We show next how to read off an S-reduction. Let p_i denote the position in the i-th term of the derivation whose residual in u is p. If p_{i+1} is at or below $p_i.1$ then we use $l \to r$ to simulate the R-step by an S-step. If p_{i+1} is at or below $p_i.2$ then we use $l \to r'$ accordingly. Else, we may ignore the step since it falls below or beside t. The constructed S-reduction is nonempty and reduces t to a term where t re-occurs: so it forms a loop. $\square$

We feel that a more elegant proof could be given, if dummy elimination and characterization by overlap closures were available for left-linear, right-linear, nonerasing TRSs.

We are in a much better situation with SRSs.

Example 9. The two-rule SRS

$$\begin{aligned} bad &\to dad\square babb \\ bd &\to db \end{aligned}$$

over the alphabet $\Sigma = \{a, b, d, \square\}$ is non-terminating but admits no loop.

Let R be the two-rule system. Non-termination is easy to show: The string *babad* issues an infinite R-reduction like in the three-rule example. For the proof that R admits no loop, we proceed in two steps. First, we apply dummy elimination to R, yielding the system $E(R)$ as follows.

$$bad \to dad \tag{11}$$
$$bad \to babb \tag{12}$$
$$bd \to db \tag{13}$$

Every $E(R)$-reduction is also a S-reduction where S is the SRS of Ex. 7. Since S admits no loop, $E(R)$ admits no loop either. By Thm. 12 then, R admits no loop either.

9 Decidability

In this section all signatures and TRSs are assumed to be finite. It is well-known that termination of TRSs is an undecidable property, even for SRSs

[13] and for one-rule TRSs [4, 17]. The problem whether termination is undecidable for one-rule SRSs is still open.

In this section we consider the question of *decidability of loopingness*. It turns out that, like termination, the existence of loops is undecidable even for one-rule TRSs. However, the existence of loops is semi-decidable, while termination is neither semi-decidable nor co-semi-decidable. Before presenting the results we recall some standard definitions and results.

A problem upon a fixed kind of data with a logical answer 'yes' or 'no' is called *semi-decidable* if there exists an algorithm with the same kind of data as input with the following properties:

- if the right answer for the problem on the input is 'yes', then the algorithm delivers 'true' in a finite amount of time;
- if the right answer for the problem on the input is 'no', then the algorithm either delivers 'false' or does not terminate.

A problem is called *co-semi-decidable* if its complement ('yes' and 'no' exchanged) is semi-decidable. It is known that a semi-decidable, co-semi-decidable problem is decidable, and that semi-decidability coincides with recursive enumerability.

In case of the halting problem for Turing machines the input consists of a Turing machine and an initial configuration, and the problem is whether the Turing machine halts starting from that particular initial configuration. A Turing machine is called uniformly halting if it halts for every initial configuration.

Proposition 24. *Termination is neither semi-decidable nor co-semi-decidable, both for SRSs and for one rule TRSs.*

Proof. Turing machines M can be transformed to SRSs R_M such that R_M is terminating if and only if M is uniformly halting [13]. In a similar way, Turing machines can be transformed to a one-rule TRS [4]. Now the proposition follows from the known fact that unifom halting of Turing machines is neither semi-decidable nor co-semi-decidable (for a proof see e.g. [5], p. 224). □

Now we arrive at the results on looping.

Proposition 25. *It is semi-decidable whether a TRS admits a loop.*

Proof. Since the signature is finite, there exists an enumeration $t_1, t_2, t_3, \dots$ of all terms. Since the TRS is finite its rewrite relation is finitely branching and computable. For a fixed t and fixed positive integer k all finitely many u satisfying $t \to^k u$ can be computed, and it can be checked whether $u = c[t\sigma]_p$ for some c, p, σ or not. Now the semi-decision procedure consists of steps $1, 2, 3, \dots$, where in step n it is checked whether $i \leq n$ and $k \leq n$, and c, σ exist satisfying $t_i \to^k c[t_i\sigma]_p$. □

Proposition 26. *It is not co-semi-decidable whether a one rule TRS admits a loop.*

Proof. Due to proposition 25 it is sufficient to prove that non-loopingness is undecidable. For arbitrary TRSs this has been proved by Plaisted [22]. The stronger result for one rule TRSs has been proved by Middeldorp and Gramlich [19, 20] and Lescanne [17]. □

For ground TRSs termination has been proved to be decidable by Huet and Lankford [13]; this result easily extends to right-ground TRSs [6]. It is easy to see that for right-ground TRSs the notions of termination and non-loopingness coincide, hence non-loopingness is decidable for right-ground TRSs.

As in the proof of proposition 25 it is easily seen that it is semi-decidable whether a term initiates a loop with respect to some TRS. On the other hand it is co-semi-decidable whether a term initiates an infinite derivation with respect to some TRS. As a consequence, for classes of TRSs in which every infinite derivation contains a loop, for instance non-length-increasing TRSs and right-ground TRSs, it is decidable whether a term initiates an infinite derivation or not.

For SRSs, Otto [21] has shown that the existence of *proper* loops (i.e. loops that are not cycles) is undecidable. Kurth [16] has given a decision procedure for the problem whether a one-rule SRS admits a loop of length 1, 2, or 3. In a recent technical report [18], McNaughton has given a decidable property, namely the existence of infinite, "well-behaved" derivations, which is sufficient for the existence of loops. We take this as a heavy argument in favour of decidability of the existence of loops for one-rule SRSs.

Finally, a remarkable connection to simple termination is worth mentioning. As Kurihara and Ohuchi [14] have shown, a finite TRS R over signature $\mathcal{F}$ is simply terminating if and only if, the transitive closure of the rewrite relation of the TRS $R \cup \mathcal{E}mb(\mathcal{F})$ is irreflexive. Here

$$\mathcal{E}mb(\mathcal{F}) =_{\text{def}} \{f(x_1, \ldots, x_n) \to x_i \mid f \in \mathcal{F}\ n\text{-ary}, x_1, \ldots, x_n \in \mathcal{X}, 1 \leq i \leq n\}$$

By our characterization of non-looping by irreflexivity (Prop. 14), we can derive the following little result.

Proposition 27. *A finite TRS R over signature $\mathcal{F}$ is simply terminating if and only if, $R \cup \mathcal{E}mb(\mathcal{F})$ is non-looping.*

Proof.

$$\begin{aligned} &R \text{ simplifying} \iff \to^{+}_{R \cup \mathcal{E}mb(\mathcal{F})} \text{ irreflexive} \\ &\quad \iff >_{R \cup \mathcal{E}mb(\mathcal{F})} \text{ irreflexive} \iff R \cup \mathcal{E}mb(\mathcal{F}) \text{ non-looping} . \end{aligned}$$

□

So our methods may be useful to prove that a finite TRS is simply terminating, without using a simplification order for this purpose.

As an immediate consequence by Prop. 25 we get:

Corollary. *Simple termination is co-semi-decidable for TRSs.*

This is in contrast to termination which is neither semi-decidable nor co-semi-decidable.

10 Conclusion

We have checked four methods which preserve termination of TRSs, whether they preserve non-looping, too. It turns out that *transformation order* does not preserve non-looping, even under fairly hard restrictions. A practically important special case of transformation order, *dummy introduction*, however does. *Dummy elimination* preserves non-looping SRSs. *Semantic labelling* generally does not preserve non-looping; it does if the underlying model is finite.

The methods which preserve non-looping can be used to infer that a given TRS is non-looping, as exhibited in the two-rule example. The other methods are applicable to detect non-termination in the non-looping case, as they may map a non-looping infinite reduction to a looping one.

References

[1] Bachmair, L., and Dershowitz, N. Commutation, transformation, and termination. In *Proc. 8th Int. Conf. Automated Deduction* (July 1986), J. Siekmann, Ed., Springer LNCS 230, Springer, pp. 5–20.

[2] Bellegarde, F., and Lescanne, P. Transformation ordering. In *2nd TAPSOFT* (1987), Springer LNCS 249, pp. 69–80.

[3] Bellegarde, F., and Lescanne, P. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing 1*, 2 (1990), 79–96.

[4] Dauchet, M. Simulation of Turing machines by a regular rewrite rule. *Theoretical Computer Science 103*, 2 (1992), 409–420. Appeared before in Proceedings of RTA89, Lecture Notes in Computer Science 355, Springer, 1989.

[5] Davis, M. D., Sigal, R., and Weyuker, E. J. *Computability, complexity, and languages – fundamentals of theoretical computer science*, 2nd ed. Computer Science and Scientific Computing. Academic Press, 1994.

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

[7] Dershowitz, N. Termination of rewriting. *J. Symb. Comput. 3*, 1&2 (Feb./April 1987), 69–115. Corrigendum: 4, 3, Dec. 1987, 409-410.

[8] Dershowitz, N., and Hoot, C. Natural termination. *Theoretical Comput. Sci.* (1994). submitted.

[9] Dershowitz, N., and Jouannaud, J.-P. Notations for rewriting. *Bull. EATCS 43* (1991), 162–172.

[10] Ferreira, M. C. F., and Zantema, H. Dummy elimination: Making termination easier. In *Proc. 10th Cpnf. Fundamentals of Computation Theory* (1995), H. Reichel, Ed., Springer LNCS965, Springer, pp. 243–252.

[11] Geupel, O. Overlap closures and termination of term rewriting systems. Tech. Rep. MIP-8922, Universität Passau, Germany, 1989.

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

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

[14] Kurihara, M., and Ohuchi, A. Modularity of simple termination of term rewriting systems. *Journal of IPS Japan 31*, 5 (1990), 633–642.

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

[16] Kurth, W. One-rule semi-thue systems with loops of length one, two, or three. *RAIRO Inform. Théor.* (1995). submitted.

[17] Lescanne, P. On termination of one rule rewrite systems. *Theoretical Computer Science 132* (1994), 395–401.

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

[19] Middeldorp, A., and Gramlich, B. Simple termination is difficult. In *Proceedings of the 5th Conference on Rewriting Techniques and Applications* (1993), C. Kirchner, Ed., vol. 690 of *Lecture Notes in Computer Science*, Springer.

[20] Middeldorp, A., and Gramlich, B. Simple termination is difficult. *Applicable Algebra in Engineering, Communication and Computing 6*, 2 (1995), 115–128.

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

[22] Plaisted, D. The undecidability of self-embedding for term rewriting systems. *Information Processing Letters 20* (1985), 61–64.

[23] Purdom, Jr, P. W. Detecting looping simplifications. In *Proc. 2nd Int. Conf. Rewriting Techniques and Applications* (May 1987), P. Lescanne, Ed., Springer LNCS 256, Springer, pp. 54–61.

[24] Rosen, B. Tree-manipulating systems and church-rosser theorems. *J. ACM 20*, 1 (Jan. 1973), 160–187.

[25] Zantema, H. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae 24* (1995), 89–105.

[26] Zantema, H., and Geser, A. A complete characterization of termination of $0^p1^q \to 1^r0^s$. In *Proceedings of the 6th Conference on Rewriting Techniques and Applications* (1995), J. Hsiang, Ed., vol. 914 of *Lecture Notes in Computer Science*, Springer, pp. 41–55. Appeared as report UU-CS-1994-44, Utrecht University.

[27] Zhang, X. Overlap closures do not suffice for termination of general term rewriting systems. *Inf. Process. Lett. 37*, 1 (1991), 9–11.