Simple Termination is Difficult

Aart Middeldorp

Institute of Information Sciences and Electronics
University of Tsukuba, Tsukuba 305, Japan
`ami@softlab.is.tsukuba.ac.jp`

Bernhard Gramlich

Fachbereich Informatik, Universität Kaiserlautern
Postfach 3049, D-6750 Kaiserlautern, Germany
`gramlich@informatik.uni-kl.de`

ABSTRACT

A terminating term rewriting system is called simply terminating if its termination can be shown by means of a simplification ordering, an ordering with the property that a term is always bigger than its proper subterms. Almost all methods for proving termination yield, when applicable, simple termination. We show that simple termination is an undecidable property, even for one-rule systems. This contradicts a result by Jouannaud and Kirchner. The proof is based on the ingenious construction of Dauchet who showed the undecidability of termination for one-rule systems.

1. Introduction

It is well-known that termination is an undecidable property of term rewriting systems. This result was obtained by Huet and Lankford [8] in 1978. They showed that every Turing machine can be coded as a string rewriting system—a term rewriting system with only unary function symbols—such that termination of the resulting string rewriting system is equivalent to the uniform halting problem for the originating Turing machine. The number of rules in their construction depends on the number of Turing machine instructions. Later, Dershowitz [3] showed that every Turing machine can be simulated by means of a two-rule term rewriting system. This result was improved by Dauchet [2], who showed that termination remains undecidable even if we restrict our attention to one-rule term rewriting systems that are orthogonal and variable preserving. His skillful construction will be explained in detail later in this paper. On the other hand, Caron [1] recently showed that termination is an undecidable property of length-preserving string rewriting systems—systems in which the left-hand side and the right-hand side of each rule have the same length—by a reduction to the uniform halting problem for linear bounded automata—a restricted kind of Turing machines.

From this last result one easily obtains the undecidability of *simple termination* for the same class of term rewriting systems. Simple termination is a stronger notion than termination. A term rewriting system is simply terminating if the addition of all rewrite rules of the form $f(x_1, \ldots, x_n) \to x_i$ results in a terminating system. Virtually all methods for proving termination yield, when applicable, simple termination. Simple termination is closely related to the *non-self-embedding* property, since every simply terminating term rewriting system is non-self-embedding. Plaisted [15] showed that the non-self-embedding property is undecidable. From this result we cannot infer the undecidability of simple termination, however. As a matter of fact, it is known that negative results for the class of non-self-embedding systems do not always carry over to the class of simply terminating systems, see [6]. In this paper we show the undecidability of simple termination for one-rule term rewriting systems. This contradicts a result of Jouannaud and Kirchner [9]. The undecidability proof is based on the ingenious construction of Dauchet. He showed in [2] that with every Turing machine M one can associate a term rewriting system $\mathcal{R}_M$ consisting of a single rewrite rule such that

$$\begin{gathered} M \text{ halts for all configurations} \\ \Longleftrightarrow \\ \mathcal{R}_M \text{ is terminating.} \end{gathered}$$

From this we cannot immediately infer the undecidability of simple termination for one-rule systems, since the implication "$\mathcal{R}_M$ is terminating $\Rightarrow$ $\mathcal{R}_M$ is simply terminating" does not hold for every Turing machine M. However, we will show that if we start the construction of Dauchet from a linear bounded automaton M instead of a Turing machine, termination and simple termination of $\mathcal{R}_M$ coincide.

The paper is organized as follows. The next section contains a brief introduction to term rewriting, including a discussion of the property simple termination. In Section 3 we define linear bounded automata. Section 4 describes Dauchet's construction. Actually, we present a somewhat simpler construction. We show that the equivalence

$$\begin{gathered} M \text{ halts for all configurations} \\ \Longleftrightarrow \\ \mathcal{R}_M \text{ is terminating} \end{gathered}$$

is easily obtained for all linear bounded automata M by using a recent result of Zantema [16] on type removal. In Section 5 we show that the equivalence

$$\mathcal{R}_M \text{ is terminating}$$
$$\Longleftrightarrow$$
$$\mathcal{R}_M \text{ is simply terminating}$$

follows for all linear bounded automata M by using a powerful method of Zantema [17] for establishing (simple) termination.

2. Simple Termination

We start with a brief introduction to term rewriting. Term rewriting is surveyed in Dershowitz and Jouannaud [4] and Klop [10].

A *signature* is a set $\mathcal{F}$ of *function symbols*. Associated with every $f \in \mathcal{F}$ is a natural number denoting its arity. Function symbols of arity 0 are called *constants*. Let $\mathcal{T}(\mathcal{F},\mathcal{V})$ be the set of all terms built from $\mathcal{F}$ and a countably infinite set $\mathcal{V}$ of *variables*, disjoint from $\mathcal{F}$. If t is a term then $\mathcal{V}ar(t)$ denotes the set of variables occurring in t. A term t is called *ground* if $\mathcal{V}ar(t) = ?$. The set of all ground terms is denoted by $\mathcal{T}(\mathcal{F})$. A term t is called *linear* if it does not contain multiple occurrences of the same variable. The *root symbol* of a term t is defined as follows: $\mathit{root}(t) = t$ if t is a variable and $\mathit{root}(t) = f$ if $t = f(t_1,\ldots,t_n)$. The *size* $|t|$ of a term t is the number of variables and function symbols occurring in t.

We introduce a fresh constant symbol Λ, named *hole*. A *context* C is a term in $\mathcal{T}(\mathcal{F}\cup\{\Lambda\},\mathcal{V})$. The designation *term* is restricted to members of $\mathcal{T}(\mathcal{F},\mathcal{V})$. A context may contain zero, one or more holes. If C is a context with n holes and $t_1,\ldots,t_n$ are terms then $C[t_1,\ldots,t_n]$ denotes the result of replacing from left to right the holes in C by $t_1,\ldots,t_n$. A term s is a *subterm* of a term t if there exists a context C such that $t = C[s]$. A subterm s of t is *proper*, denoted by $t \rhd s$, if $s \neq t$. A *substitution* is a map σ from $\mathcal{V}$ to $\mathcal{T}(\mathcal{F},\mathcal{V})$. If σ is a substitution and t a term then $t\sigma$ denotes the result of applying σ to t. We call $t\sigma$ an *instance* of t. A binary relation $\succ$ on terms is a *rewrite relation* if it is closed under contexts and substitutions, i.e. if $s \succ t$ then $C[s\sigma] \succ C[t\sigma]$ for all contexts C (with precisely one hole) and substitutions σ.

A *rewrite rule* is a pair (l,r) of terms such that the left-hand side l is not a variable and variables which occur in the right-hand side r occur also in l, i.e. $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. Rewrite rules (l,r) will henceforth be written as $l \to r$. A rewrite rule is *collapsing* if its right-hand side is a single variable. A rewrite rule is *duplicating* if its right-hand side contains more occurrences of some variable than its left-hand side. A rewrite rule is *left-linear* (*right-linear*) if its left-hand (right-hand) side is a linear term,

A *term rewriting system* (TRS for short) is a pair $(\mathcal{F},\mathcal{R})$ consisting of a signature $\mathcal{F}$ and a set $\mathcal{R}$ of rewrite rules between terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. We often present a TRS as a set of rewrite rules, without making explicit its signature, assuming that the signature consists of the function symbols occurring in the rewrite rules.

If $(\mathcal{F}, \mathcal{R})$ is a TRS then $\rightarrow_{\mathcal{R}}$ denotes the smallest rewrite relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ containing $\mathcal{R}$. So $s \rightarrow_{\mathcal{R}} t$ if there exists a rewrite rule $l \rightarrow r$ in $\mathcal{R}$, a substitution σ and a context C such that $s = C[l\sigma]$ and $t = C[r\sigma]$. The subterm $l\sigma$ of s is called a *redex* and we say that s rewrites to t by *contracting* redex $l\sigma$. We call $s \rightarrow_{\mathcal{R}} t$ a *rewrite* or *reduction step*. If $C = \Lambda$ then we speak of a *root* reduction. The transitive closure of $\rightarrow_{\mathcal{R}}$ is denoted by $\rightarrow_{\mathcal{R}}^{+}$ and $\rightarrow_{\mathcal{R}}^{*}$ denotes the transitive-reflexive closure of $\mathcal{R}$. If $s \rightarrow_{\mathcal{R}}^{*} t$ we say that s *reduces* to t. A TRS $(\mathcal{F}, \mathcal{R})$ is called *terminating* if there are no infinite reduction sequences $t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} t_3 \rightarrow_{\mathcal{R}} \cdots$ of terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$.

A rewrite relation that is also a (strict) partial order is called a *rewrite order*. A TRS $(\mathcal{F}, \mathcal{R})$ is *compatible* with a rewrite order $\succ$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ if $l \succ r$ for every rewrite rule $l \rightarrow r$ of $\mathcal{R}$. It is easy to show that a TRS is terminating if and only if it is compatible with a well-founded rewrite order.

Definition 2.1.

- A *simplification order* is a rewrite order $\succ$ with the *subterm property*, i.e. $C[t] \succ t$ for all contexts $C \neq \Lambda$ (with precisely one hole) and terms t.
- A TRS is called *simplifying* if it is compatible with a simplification order.
- A TRS is called *simply terminating* if it is compatible with a well-founded simplification order.

Clearly every simply terminating TRS is both simplifying and terminating. A simplifying TRS $(\mathcal{F}, \mathcal{R})$ with $\mathcal{F}$ or $\mathcal{R}$ finite is simply terminating, as a consequence of Kruskal's Tree Theorem. There exists (infinite) simplifying and terminating TRSs that are not simply terminating, see Ohlebusch [14]. This does not concern us too much as we will deal with decidability issues in the sequel, in which one considers only finite (both with respect to signature and set of rewrite rules) TRSs. Next we present a useful characterization of simple termination.

Definition 2.2. Let $\mathcal{F}$ be a signature. The TRS $\mathcal{E}mb\,(\mathcal{F})$ consists of all rewrite rules

$$f(x_1, \ldots, x_n) \quad \rightarrow \quad x_i$$

with $f \in \mathcal{F}$ a function symbol of arity n ¿ 1 and $i \in \{1, \ldots, n\}$. We write s . t for terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ if $t \rightarrow_{\mathcal{E}mb\,(\mathcal{F})}^{*} s$. The relation . is called *embedding*.

Lemma 2.3. *Let $(\mathcal{F}, \mathcal{R})$ be a TRS. The following statements are equivalent.*

- *The TRS $(\mathcal{F}, \mathcal{R})$ is simply terminating.*
- *The TRS $(\mathcal{F}, \mathcal{R}) \cup \mathcal{E}mb\,(\mathcal{F})$ is simply terminating.*
- *The TRS $(\mathcal{F}, \mathcal{R}) \cup \mathcal{E}mb\,(\mathcal{F})$ is terminating.*

Λ

The proof is not difficult. This lemma appeared for the first time in Zantema [17], although it is implicit in many earlier works on termination, see Dershowitz [3] for a survey. Kurihara and Ohuchi [11, 12] proved the related equivalence "a TRS $(\mathcal{F}, \mathcal{R})$ is simplifying $\Longleftrightarrow$ the transitive closure of the rewrite relation associated to the TRS $(\mathcal{F}, \mathcal{R}) \cup \mathcal{E}mb\,(\mathcal{F})$ is irreflexive".

The above lemma facilitates an easy proof of the undecidability of simple termination. For that matter we need some background on string rewriting systems. A *string rewriting system* (SRS) is a TRS $(\mathcal{F}, \mathcal{R})$ whose signature $\mathcal{F}$ contains only unary function symbols. A SRS $(\mathcal{F}, \mathcal{R})$ is called *non-length-increasing* if every rewrite rule $l \to r$ of $\mathcal{R}$ satisfies $|l|$ ¿ $|r|$. We call $\mathcal{R}$ *length-preserving* if $|l| = |r|$ for every rewrite rule $l \to r \in \mathcal{R}$. In the introduction we already mentioned that Caron [1] showed the undecidability of termination for length-preserving SRSs. Combining this result with Lemma 2.3 yields the undecidability of simple terminating for the same class of TRSs since it is very easy to show that a length-preserving SRS $(\mathcal{F}, \mathcal{R})$ is terminating if and only if the non-length-increasing SRS $(\mathcal{F}, \mathcal{R}) \cup \{f(x) \to x \mid f \in \mathcal{F}\}$ is terminating.

In the following sections we show that simple termination is an undecidable property of one-rule TRSs. This contradicts a result by Jouannaud and Kirchner [9]. They claimed that a one-rule TRS $\{l \to r\}$ is simply terminating if and only if l does not unify with any non-variable term embedded in r. This decision procedure is wrong as can be seen from the one-rule TRS $\mathcal{R} = \{f(a, b, x) \to f(x, x, x)\}$. The only non-variable term embedded in the right-hand side $f(x, x, x)$ is $f(x, x, x)$ itself, which clearly does not unify with the left-hand side $f(a, b, x)$. On the other hand, the term $f(a, b, f(a, b, b))$ has an infinite reduction with respect to the TRS

$$\left\{ \begin{array}{lcl} f(a,b,x) & \to & f(x,x,x) \\ f(x,y,z) & \to & x \\ f(x,y,z) & \to & y \\ f(x,y,z) & \to & z \end{array} \right\}$$

and hence $\mathcal{R}$ is not simply terminating, as a consequence of Lemma 2.3.

We would like to conclude this section with mentioning a (famous) open problem: the decidability of termination for one-rule SRSs. Partial results were obtained by Kurth [13]. He showed that termination is decidable in case the number of function symbols in the right-hand side of the single rewrite rule does not exceed six. Deciding the termination of one-rule non-length-increasing SRSs is much easier: a non-length-increasing SRS $\{l \to r\}$ is terminating if and only if $l \neq r$. Another open problem is whether termination is decidable for TRSs having only one left and right-linear rewrite rule (problem 21 in [5]).

3. Linear Bounded Automata

In this section we introduce linear bounded automata. Before presenting formal definitions, we give an intuitive description.

A linear bounded automaton consists of a tape which is divided into cells, a tape head that scans one cell at a time, and a finite control, see Figure 1(i). Each cell of the tape contains one symbol of a finite alphabet. A linear bounded

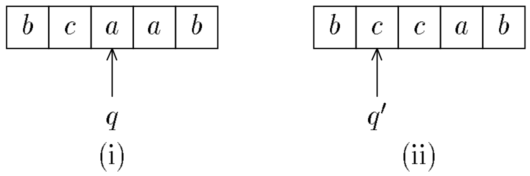

FIGURE 1.

automaton operates as follows. Depending on the state of the finite control and the symbol scanned by the tape head, a linear bounded automaton

- changes state,
- replaces the symbol scanned by the tape head by another symbol, and
- moves the tape head one cell to the left or to the right.

It is not required that the new state or the new tape symbol differ from the previous ones. On certain combinations of state and tape symbol, the linear bounded automaton stops operating. Moves to the left are not allowed if the tape head is positioned at the leftmost cell of the tape. Likewise, a right-move is forbidden if the tape head points to the rightmost cell of the tape.

DEFINITION 3.1.

- A (deterministic) *linear bounded automaton* (LBA for short) is a triple $M = (Q, \Gamma, \delta)$ consisting of a finite set Q of *states*, a finite set Γ of *tape symbols*, disjoint from Q, and a *transition function* δ, which is a partial mapping from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$.
- Let $M = (Q, \Gamma, \delta)$ be an LBA. A *configuration* is an element of $\Gamma^* Q \Gamma^+$, i.e. a string $w_1 q w_2$ with q a state, w_1 a string of tape symbols and w_2 a non-empty string of tape symbols. The idea is that the LBA scans the leftmost symbol of w_2. If $w_1 = \varepsilon$ then the tape head is positioned at the leftmost cell of the tape. The transition function δ determines a relation $\vdash_M$ on configurations as follows:

transition step	provided
$w_1 qabw_2 \vdash_M w_1 a' q' b w_2$	$\delta(q, a) = (q', a', R)$
$w_1 bqaw_2 \vdash_M w_1 q' b a' w_2$	$\delta(q, a) = (q', a', L)$

Here $q, q' \in Q$, $a, a', b, b' \in \Gamma$ and $w_1, w_2 \in \Gamma^*$. Observe that for every configuration α there is at most one configuration β such that $\alpha \vdash_M \beta$. In other words, the transition relation $\vdash_M$ is deterministic.

The situation of Figure 1(i) can be described by the configuration *bcqaab*. If $\delta(q, a) = (q', c, L)$ then $bcqaab \vdash bq'ccab$, i.e. the situation of Figure 1(ii) is obtained.

DEFINITION 3.2.

- Let M be an LBA and α a configuration of M. We say that M *halts* for α if there is no infinite sequence $\alpha \vdash_M \alpha' \vdash_M \alpha'' \vdash_M \cdots$

- The *halting problem* is the following decision problem: given an LBA M and a configuration α of M, does M halt for α? The *uniform halting problem* is the problem to decide whether a given LBA M halts for all its configurations.

Observe that the halting problem is decidable, since for any configuration α of an LBA M there are only finitely many different configurations α' reachable from α (i.e. $\alpha \vdash_M^* \alpha'$). Hence halting can be decided by enumerating the (unique) sequence $\alpha \vdash_M \alpha' \vdash_M \alpha'' \vdash_M \cdots$. If M does not halt for α then at some stage we will reach a configuration that occurred earlier in the sequence. The uniform halting problem is undecidable though.

THEOREM 3.3. *Let M be an arbitrary LBA. It is undecidable whether M halts for all its configurations.* Λ

A proof of this statement can be found in Caron [1], where a reduction to Post's Correspondence Problem is given. Caron ascribes the above result to Hooper [7], but she obtained it independently. Moreover, [7] is very hard to read—there is for instance no notion of LBA—and it is not clear at all whether we may assume the simple definition of LBA given above (in order to conclude the undecidability of the uniform halting problem). By coding every LBA as a length-preserving SRS, similar to the construction described in Huet and Lankford [8], Caron reduced the undecidability of termination for length-preserving SRSs to the uniform halting problem for LBAs.

We conclude this section with a concrete example of an LBA, which will be used to illustrate subsequent developments.

EXAMPLE 3.4. Consider the LBA $M = (Q, \Gamma, \delta)$ with $Q = \{p, q\}$, $\Gamma = \{a, b\}$ and δ defined by the following table:

	a	b
p	(p, b, R)	(q, a, L)
q		(p, a, R)

The LBA M halts for configuration pab since $pab \vdash_M bpb \vdash_M qba \vdash_M apa$ and there is no transition step possible from configuration apa since abp is not a configuration.

4. Dauchet's Construction

In this section we associate with every LBA M a one-rule TRS $\mathcal{R}_M$ such that M halts for all its configurations if and only if $\mathcal{R}_M$ is terminating. In the next section we show that simple termination of $\mathcal{R}_M$ coincides with termination. Our construction is somewhat simpler than the one by Dauchet [2]—we use for instance only five variables as opposed to the six used by Dauchet—but the essence is the same.

DEFINITION 4.1. Let $M = (Q, \Gamma, \delta)$ be an arbitrary LBA. Suppose $Q = \{q_1, \ldots, q_m\}$, $\Gamma = \{a_1, \ldots, a_n\}$ and the number of pairs in $Q \times \Gamma$ for which δ is defined equals p. (So M contains p instructions.) The signature $\mathcal{F}_M$ of $\mathcal{R}_M$ consists of the following symbols:

- constants $q_1, \ldots, q_m$ and $a_1, \ldots, a_n$,
- a binary function symbol c and two constants $\sharp$ and NIL,
- a function symbol L of arity $m + n + 3$ and a function symbol R of arity p.

The use of the same characters for function symbols on the one hand, and states and tape symbols on the other hand, will cause no confusion. Next we define the single rewrite $l_M \to r_M$ of the TRS $\mathcal{R}_M$. The left-hand side l_M is the term

$$L(\mathrm{c}(x_1, x_2), x_3, \mathrm{c}(x_4, x_5), q_1, \ldots, q_m, a_1, \ldots, a_n).$$

Here $x_1, \ldots, x_5$ are (pairwise different) variables. The right-hand side r_M is the term

$$R(r_1, \ldots, r_p)$$

with r_k (1 6 k 6 p) defined as follows:

$$r_k = L(\mathrm{c}(a', \mathrm{c}(x_1, x_2)), q', x_5, Q_1, \ldots, Q_m, A_1, \ldots, A_n)$$

if the k-th instruction of M is a right-moving instruction $\delta(q_i, a_j) = (q', a', R)$, and

$$r_k = L(x_2, q', \mathrm{c}(x_1, \mathrm{c}(a', x_5)), Q_1, \ldots, Q_m, A_1, \ldots, A_n)$$

if the k-th instruction of M is a left-moving instruction $\delta(q_i, a_j) = (q', a', L)$. Here the terms $Q_1, \ldots, Q_m, A_1, \ldots, A_n$ are defined by

$$Q_l = \begin{cases} x_3 & \text{if } i = l, \\ q_l & \text{if } i \neq l \end{cases}$$

for 1 6 l 6 m and

$$A_l = \begin{cases} x_4 & \text{if } j = l, \\ a_l & \text{if } j \neq l \end{cases}$$

for 1 6 l 6 n.

Let us try to explain the construction. The idea is that every configuration corresponds to an instance of the left-hand side. The first argument will contain the contents of the tape to the left of the tape head, the second argument will contain the state of the configuration, and the third argument will contain the contents of the tape cell scanned by the tape head as well as the contents of the tape to the right of the tape head. Tape parts are represented as terms by using the constructors c and NIL. For instance, aab will correspond to the term $\mathrm{c}(a, \mathrm{c}(a, \mathrm{c}(b, \text{NIL})))$. However, the contents of the tape to the left of the tape head should be represented in reverse order since the rightmost symbol will be accessed first. So the instance of the left-hand side representing configuration $abqab$ would have $\mathrm{C}(b, \mathrm{C}(a, \mathrm{NIL}))$, q and $\mathrm{C}(a, \mathrm{C}(b, \mathrm{NIL})))$ as first three arguments. There is only one problem with this approach: if the tape head is positioned at the leftmost tape cell then the term representing the empty tape part to its left would simply be NIL which is not an instance of $\mathrm{C}(x_1, x_2)$, the first argument of the left-hand side. For that reason we introduced the special constant $\sharp$, the trick being to represent a configuration like $abqab$ by the three terms $\mathrm{C}(b, \mathrm{C}(a, \mathrm{C}(\sharp, \mathrm{NIL})))$, q and $\mathrm{C}(a, \mathrm{C}(b, \mathrm{NIL})))$. So the configuration qa will be represented by $\mathrm{C}(\sharp, \mathrm{NIL})$, q and $\mathrm{C}(a, \mathrm{NIL})$. Observe that there is no need to add the symbol $\sharp$ to the third term since the string w_2 in a configuration $w_1 q w_2$ is always non-empty. After this informal discussion, the following definition is easy.

Definition 4.2. Let $M = (Q, \Gamma, \delta)$ be an LBA. We define two translations ϕ_1 and ϕ_2 from Γ^* to $\mathcal{T}(\Gamma \cup \{\mathrm{C}, \sharp, \mathrm{NIL}\})$ as follows:

$$\phi_1(w) = \begin{cases} \mathrm{C}(\sharp, \mathrm{NIL}) & \text{if } w = \varepsilon, \\ \mathrm{C}(a, \phi_1(w')) & \text{if } w = w'a \end{cases}$$

and

$$\phi_2(w) = \begin{cases} \mathrm{NIL} & \text{if } w = \varepsilon, \\ \mathrm{C}(a, \phi_2(w')) & \text{if } w = aw'. \end{cases}$$

These mappings are used to define a mapping ϕ from configurations of M to instances of the left-hand side of the single rewrite rule of $\mathcal{R}_M$ by means of the equation

$$\phi(w_1 q w_2) = L(\phi_1(w_1), q, \phi_2(w_2), q_1, \ldots, q_m, a_1, \ldots, a_n).$$

We still have to explain the remaining $m + n$ arguments of the L-terms occurring in the single rewrite rule, which really is the ingenious part of Dauchet's construction. This can best be done by means of a concrete example.

Example 4.3. Consider the LBA M of Example 3.4. Its associated TRS $\mathcal{R}_M$ has the rewrite rule

$$L(\mathrm{C}(x_1, x_2), x_3, \mathrm{C}(x_4, x_5), p, q, a, b)$$

$$\to R\begin{pmatrix} L(\mathrm{C}(b, \mathrm{C}(x_1, x_2)), p, x_5, x_3, q, x_4, b) \\ L(x_2, q, \mathrm{C}(x_1, \mathrm{C}(a, x_5)), x_3, q, a, x_4) \\ L(\mathrm{C}(a, \mathrm{C}(x_1, x_2)), p, x_5, p, x_3, a, x_4) \end{pmatrix}.$$

We have $pab \vdash_M bpb$. How is this transition step reflected at the rewrite level? In $\mathcal{R}_M$ we have the rewrite step

$$L(\textsc{c}(\sharp, \textsc{nil}), p, \textsc{c}(a, \textsc{c}(b, \textsc{nil})), p, q, a, b)$$

$$\rightarrow R\left(\begin{array}{l} L(\textsc{c}(b, \textsc{c}(\sharp, \textsc{nil})), p, \textsc{c}(b, \textsc{nil}), p, q, a, b) \\ L(\textsc{nil}, q, \textsc{c}(\sharp, \textsc{c}(a, \textsc{c}(b, \textsc{nil}))), p, q, a, a) \\ L(\textsc{c}(a, \textsc{c}(\sharp, \textsc{nil})), p, \textsc{c}(b, \textsc{nil}), p, p, a, a) \end{array}\right)$$

starting from $\phi(pab)$. The first argument

$$t_1 = L(\textsc{c}(b, \textsc{c}(\sharp, \textsc{nil})), p, \textsc{c}(b, \textsc{nil}), p, q, a, b)$$

of the resulting term corresponds to performing the instruction $\delta(p, a) = (p, b, R)$. This step is allowed since in configuration pab the state is p and the tape cell scanned by the tape head contains the symbol a. Notice that $t_1 = \phi(bpb)$. The second argument

$$t_2 = L(\textsc{nil}, q, \textsc{c}(\sharp, \textsc{c}(a, \textsc{c}(b, \textsc{nil}))), p, q, a, a)$$

corresponds to performing the instruction $\delta(p, b) = (q, a, L)$. This step is of course not allowed as the symbol scanned by the tape head in configuration pab is a, not b. Observe that t_2 is no longer reducible since its last argument is an a instead of a b. In addition, t_2 is not reducible because its first argument is NIL instead of an instance of $\textsc{c}(x_1, x_2)$, signaling the fact than an illegal left-move has been attempted. Finally, the third argument

$$t_3 = L(\textsc{c}(a, \textsc{c}(\sharp, \textsc{nil})), p, \textsc{c}(b, \textsc{nil}), p, p, a, a)$$

is not reducible since its last four arguments are p, p, a, a instead of p, q, a, b. This means that an instruction of the form "$\delta(q, b) = \ldots$" has been attempted where "$\delta(p, a) = \ldots$" was required.

The easy implication in the desired equivalence "an LBA M halts for all configurations if and only if the TRS $\mathcal{R}_M$ is terminating" is stated in the following lemma.

LEMMA 4.4. *Let M be a LBA. If M does not halt for configuration α then $\mathcal{R}_M$ has an infinite reduction starting from the term $\phi(\alpha)$.*

PROOF. By construction, every transition step $\alpha \vdash_M \beta$ translates to $\phi(\alpha) = l_M\sigma \rightarrow r_M\sigma$ with one of the arguments of the resulting term $r_M\sigma$ equal to $\phi(\beta)$. Thus $\phi(\alpha) \rightarrow_{\mathcal{R}_M} C[\phi(\beta)]$ for some context C. Hence an infinite transition sequence $\alpha \vdash_M \alpha' \vdash_M \alpha'' \vdash_M \cdots$ corresponds to an infinite rewrite sequence $\phi(\alpha) \rightarrow_{\mathcal{R}_M} C[\phi(\alpha')] \rightarrow_{\mathcal{R}_M} C[C'[\phi(\alpha'')]] \rightarrow_{\mathcal{R}_M} \cdots$ Λ

The validity of the implication "M halts for all configurations $\Longrightarrow \mathcal{R}_M$ is terminating" remains to be shown. This is less easy since there are many reducible terms in $\mathcal{R}_M$ that do not correspond to a configuration. However, since $\mathcal{R}_M$ contains no collapsing rules, we can use a recent result of Zantema. In [16] he showed that the termination behaviour of a TRS is not affected if we restrict our attention to well-typed terms according to some many-sorted type discipline, provided the system contains not both collapsing and duplicating rules. (See [16] for a precise formulation.) For $\mathcal{R}_M$ we take the following type discipline:

symbol	sort declaration
q_i $(1 \leqslant i \leqslant m)$, x_3	S_Q
a_i $(1 \leqslant i \leqslant n)$, $\sharp$, x_1, x_4	S_Γ
NIL, x_2, x_5	S_{LIST}
C	$S_\Gamma \times S_{\text{LIST}} \to S_{\text{LIST}}$
L	$S_{\text{LIST}} \times S_\Gamma \times S_{\text{LIST}} \times S_Q^m \times S_\Gamma^n \to S$
R	$S^p \to S$

Observe that both the left-hand side and right-hand side of the rewrite rules of $\mathcal{R}_M$ type-check and have the same sort S. Clearly only terms of sort S are reducible and hence the theorem of Zantema amounts to the equivalence of "$\mathcal{R}_M$ is terminating" and "$\mathcal{R}_M$ is terminating for all terms of sort S". It is not difficult to show that this last statement can be strengthened to "$\mathcal{R}_M$ is terminating for all *ground redexes* of sort S". So the problem remains how to extract an infinite transition sequence $\alpha_1 \vdash_M \alpha_2 \vdash_M \alpha_3 \vdash_M \cdots$ from an infinite rewrite sequence $t_1 \to_{\mathcal{R}_M} t_2 \to_{\mathcal{R}_M} t_3 \to_{\mathcal{R}_M} \cdots$ of ground terms of sort S with t_1 a redex.

DEFINITION 4.5. Let $M = (Q, \Gamma, \delta)$ be an LBA. Let $\Gamma_\sharp = \Gamma \cup \{\sharp\}$. We define two translations ψ_1 and ψ_2 from the set of ground terms of sort S_{LIST} to $\Gamma_\sharp^*$ as follows:

$$\psi_1(t) = \begin{cases} \varepsilon & \text{if } t = \text{NIL}, \\ \psi_1(t_2)\, t_1 & \text{if } t = \text{C}(t_1, t_2) \end{cases}$$

and

$$\psi_2(t) = \begin{cases} \varepsilon & \text{if } t = \text{NIL}, \\ t_1 \psi_2(t_2) & \text{if } t = \text{C}(t_1, t_2). \end{cases}$$

Observe that $\psi_2(t)$ is simply the reverse of $\psi_1(t)$. These mappings induce a mapping ψ from ground redexes of sort S to elements of $\Gamma_\sharp^* Q \Gamma_\sharp^+$ by means of the equation

$$\psi(L(t_1, t_2, t_3, q_1, \ldots, q_m, a_1, \ldots, a_n)) = \psi_1(t_1)\, t_2 \psi_2(t_3).$$

Because of the presence of $\sharp$, elements of $\Gamma_\sharp^* Q \Gamma_\sharp^+$ are not configurations in the sense of Definition 3.1. The transition relation $\vdash_M$ however easily extends to elements of $\Gamma_\sharp^* Q \Gamma_\sharp^+$ by relaxing $b \in \Gamma$ and $w_1, w_2 \in \Gamma^*$ in Definition 3.1 to $b \in \Gamma_\sharp$ and $w_1, w_2 \in \Gamma_\sharp^*$. In the proof of Lemma 4.7 below, we will extract an infinite $\vdash_M$-sequence of elements of $\Gamma_\sharp^* Q \Gamma_\sharp^+$ from a presupposed infinite reduction sequence (of ground terms of sort S) in $\mathcal{R}_M$. In order to obtain an infinite $\vdash_M$-sequence of configurations, we have to get rid of the $\sharp$'s. The next definition provides an easy solution.

DEFINITION 4.6. We define a mapping χ from $\Gamma_\sharp^*$ to Γ^* inductively as follows:

$$\chi(w) = \begin{cases} \varepsilon & \text{if } w = \varepsilon, \\ a\chi(w') & \text{if } w = aw' \text{ with } a \in \Gamma, \\ \chi(w') & \text{if } w = \sharp w'. \end{cases}$$

This mapping is extended to elements of $\Gamma_\sharp^* Q \Gamma_\sharp^+$ by putting

$$\chi(w_1 q w_2) = \chi(w_1) q \chi(w_2).$$

Observe that $\chi(w_1 q w_2)$ is not necessarily a configuration, since $\chi(w_2)$ may be the empty string. However, it is not difficult to see that if $\alpha_1 \vdash_M \alpha_2 \vdash_M \alpha_3$ with $\alpha_1, \alpha_2, \alpha_3 \in \Gamma_\sharp^* Q \Gamma_\sharp^+$, then $\chi(\alpha_1)$ and $\chi(\alpha_2)$ are configurations such that $\chi(\alpha_1) \vdash_M \chi(\alpha_2)$. This implies that an infinite $\vdash_M$-sequence of elements of $\Gamma_\sharp^* Q \Gamma_\sharp^+$ is transformed by χ into an infinite $\vdash_M$-sequence of configurations.

Lemma 4.7. *Let M be a LBA. If $\mathcal{R}_M$ is not terminating then M does not halt for all configurations.*

Proof. Suppose $\mathcal{R}_M$ is not terminating. From the preceding discussion we know that there exists an infinite reduction sequence $t_1 \to_{\mathcal{R}_M} t_2 \to_{\mathcal{R}_M} t_3 \to_{\mathcal{R}_M} \cdots$ of ground terms of sort S with t_1 a redex. Consider the first step $t_1 \to_{\mathcal{R}_M} t_2$. Because M is deterministic, at most one of the arguments of t_2 is a ground redex (of sort S). From the reducibility of t_2 we infer that precisely one of its arguments is reducible. Let us call this argument t_2'. We have $\psi(t_1) \vdash_M \psi(t_2')$ by construction of $\mathcal{R}_M$. Let C be the context such that $t_2 = C[t_2']$. There exist terms t_i' for i ¿ 3 such that $t_i = C[t_i']$ (i ¿ 3) and $t_2' \to_{\mathcal{R}_M} t_3' \to_{\mathcal{R}_M} \cdots$ is an infinite reduction sequence of ground terms of sort S with t_2' being a redex. Repeating the above argument yields an infinite sequence $\psi(t_1) \vdash_M \psi(t_2') \vdash_M \psi(t_3'') \to_{\mathcal{R}_M} \cdots$ of elements of $\Gamma_\sharp^* Q \Gamma_\sharp^+$. Applying the transformation χ to this sequence yields an infinite sequence of configurations $\chi(\psi(t_1)) \vdash_M \chi(\psi(t_2')) \vdash_M \chi(\psi(t_3'')) \to_{\mathcal{R}_M} \cdots$. Hence M does not halt for all configurations. Λ

5. Simple Termination is Undecidable for One-Rule Systems

Our main result follows if we can show that for the one-rule TRSs $\mathcal{R}_M$ introduced in the previous section, termination and simple termination coincide. It suffices to show that every terminating $\mathcal{R}_M$ is simply terminating. So consider an LBA M such that $\mathcal{R}_M$ is terminating. In the previous version of this paper we constructed a complicated well-founded order on $\mathcal{T}(\mathcal{F}_M)$ which extends the rewrite relation associated to the TRS $\mathcal{R}_M \cup \mathcal{E}mb(\mathcal{F}_M)$. This implies that $\mathcal{R}_M \cup \mathcal{E}mb(\mathcal{F}_M)$ terminates for all ground terms, which in turn implies the termination of $\mathcal{R}_M \cup \mathcal{E}mb(\mathcal{F}_M)$ (since every infinite reduction sequence can be transformed into an infinite reduction sequence involving only ground terms by simply substituting some constant for all variables). According to Lemma 2.3 this is equivalent to the simple termination of $\mathcal{R}_M$. In this paper we show that the powerful *distribution elimination* technique of Zantema [17] gives rise to a much simpler proof.

We start with a brief description of Zantema's technique, specialized to the present situation. Let $(\mathcal{F}, \mathcal{R})$ be a TRS and $f \in \mathcal{F}$ a function symbol of arity n ¿ 1 that does not occur in the left-hand sides of the rewrite rules in $\mathcal{R}$. We inductively define a mapping E_f that assigns to every term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ a subset of $\mathcal{T}(\mathcal{F}\backslash\{f\}, \mathcal{V})$ as follows:

$$E_f(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V}, \\ \bigcup_{i=1}^{n} E_f(t_i) & \text{if } t = f(t_1, \ldots, t_n), \\ \{g(u_1, \ldots, u_m) \mid \forall i\, u_i \in E_f(t_i)\} & \text{if } t = g(t_1, \ldots, t_m) \text{ and } f \neq g. \end{cases}$$

The set of rewrite rules $\{l \to u \mid l \to r \in \mathcal{R} \text{ and } u \in E_f(r)\}$ is denoted by $E_f(\mathcal{R})$.

THEOREM 5.1 (Zantema [17]). *If $E_f(\mathcal{R})$ is simply terminating and right-linear then $\mathcal{R}$ is simply terminating.* Λ

We would like to stress that Theorem 5.1 is only a very special case of the results in [17]. The idea is now to apply Theorem 5.1 to the TRS $\mathcal{R}_M$ with respect to the function symbol R, which only occurs in the right-hand side of the single rewrite rule of $\mathcal{R}_M$. If R happens to be a constant, i.e. if the LBA M contains no instructions, then $\mathcal{R}_M$ is immediately seen to be simply terminating. So we may assume that R is not a constant. Recall that the single rewrite rule of $\mathcal{R}_M$ has the form $l_M \to R(r_1, \ldots, r_p)$. One easily verifies that

$$E_R(\mathcal{R}_M) = \left\{ \begin{array}{ccc} l_M & \to & r_1 \\ & \vdots & \\ l_M & \to & r_p \end{array} \right\}.$$

Before we can apply Theorem 5.1 we have to check that $E_R(\mathcal{R}_M)$ is right-linear and simply terminating. Right-linearity is obvious. Simple termination follows from the termination of $\mathcal{R}_M$. First we show that $E_R(\mathcal{R}_M)$ is terminating.

LEMMA 5.2. *The TRS $E_R(\mathcal{R}_M)$ is terminating.*

PROOF. We use again the result of Zantema [16] on type removal. This is allowed since $E_R(\mathcal{R}_M)$ lacks collapsing rules. Consider the type discipline of Section 4 (page). If $E_R(\mathcal{R}_M)$ is not terminating then there exists an infinite reduction sequence in which all terms have sort S. This implies that such an infinite reduction sequence contains only root reductions. However, if $s \to_{E_R(\mathcal{R}_M)} t$ is a root reduction then there exists a context C such that $s \to_{\mathcal{R}_M} C[t]$, and hence any infinite $E_R(\mathcal{R}_M)$-reduction sequence containing only root reductions can trivially be embedded into an infinite $\mathcal{R}_M$-reduction sequence. This contradicts the termination of $\mathcal{R}_M$. Λ

Lemma 5.3. *The TRS $E_R(\mathcal{R}_M)$ is simply terminating.*

Proof. One easily verifies that $|s| = |t|$ whenever $s \rightarrow_{E_R(\mathcal{R}_M)} t$. Since clearly $|s| > |t|$ whenever $s \rightarrow_{\mathcal{E}mb\,(\mathcal{F}_M \backslash \{R\})} t$, termination of $E_R(\mathcal{R}_M) \cup \mathcal{E}mb\,(\mathcal{F}_M \backslash \{R\})$ follows from the termination of $E_R(\mathcal{R}_M)$ (Lemma 5.2). Lemma 2.3 yields the simple termination of $E_R(\mathcal{R}_M)$. Λ

It should be stressed that the above lemma does not hold if M is an arbitrary Turing machine instead of an LBA. Actually this is the only place where we use a property of $\mathcal{R}_M$ which does not hold for the single rewrite rule of Dauchet. Theorem 5.1 now yields the desired result.

Theorem 5.4. *The TRS $\mathcal{R}_M$ is simply terminating.* Λ

Corollary 5.5. *Simple termination is an undecidable property of one-rule TRSs.* Λ

Since every TRS $\mathcal{R}_M$ is *orthogonal* (left-linear and no critical pairs), *variable preserving* ($\mathcal{V}ar(l_M) = \mathcal{V}ar(r_M)$) and a *constructor system* (proper subterms of l_M do not contain the symbol $root(l_M)$), we can state that simple termination is an undecidable property of orthogonal, variable preserving, one-rule constructor systems.

References

1. A.-C. Caron, *Linear Bounded Automata and Rewrite Systems: Influence of Initial Configuration on Decision Properties*, Proceedings of the Colloquium on Trees in Algebra and Programming, Brighton, Lecture Notes in Computer Science **493**, pp. 74–89, 1991.

2. M. Dauchet, *Simulation of Turing Machines by a Regular Rewrite Rule*, Theoretical Computer Science **103**, pp. 409–420, 1992. Previous version in the Proceedings of the 3rd International Conference on Rewriting Techniques and Applications, Chapel Hill, Lecture Notes in Computer Science **355**, pp. 109–120, 1989.

3. N. Dershowitz, *Termination of Rewriting*, Journal of Symbolic Computation **3**(1), pp. 69–116, 1987.

4. N. Dershowitz and J.-P. Jouannaud, *Rewrite Systems*, in: Handbook of Theoretical Computer Science, Vol. B (ed. J. van Leeuwen), North-Holland, pp. 243–320, 1990.

5. N. Dershowitz, J.-P. Jouannaud, and J.W. Klop, *Open Problems in Rewriting*, Proceedings of the 4th International Conference on Rewriting Techniques and Applications, Como, Lecture Notes in Computer Science **488**, pp. 445–456, 1991.

6. B. Gramlich, *A Structural Analysis of Modular Termination of Term Rewriting Systems*, SEKI report SR-91-15, Universität Kaiserslautern, 1991.

7. P.K. Hooper, *The Undecidability of the Turing Machine Immortality Problem*, Journal of Symbolic Logic **31**(2), pp. 219–234, 1966.

8. G. Huet and D. Lankford, *On the Uniform Halting Problem for Term Rewriting Systems*, report 283, INRIA, 1978.

9. J.-P. Jouannaud and H. Kirchner, *Construction d'un Plus Petit Ordre de Simplification*, RAIRO Informatique Théorique **18**(3), pp. 191–207, 1984 (in French).

10. J.W. Klop, *Term Rewriting Systems*, in: Handbook of Logic in Computer Science, Vol. II (eds. S. Abramsky, D. Gabbay and T. Maibaum), Oxford University Press, pp. 1–112, 1992.

11. M. Kurihara and A. Ohuchi, *Modularity of Simple Termination of Term Rewriting Systems*, Journal of the Information Processing Society Japan **31**(5), pp. 633–642, 1990.

12. M. Kurihara and A. Ohuchi, *Modularity of Simple Termination of Term Rewriting Systems with Shared Constructors*, Theoretical Computer Science **103**, pp. 273–282, 1992.

13. W. Kurth, *Termination und Konfluenz von Semi-Thue-Systems mit nur einer Regel*, Ph.D. thesis, Technische Universität Clausthal, 1990 (in German).

14. E. Ohlebusch, *A Note on Simple Termination of Infinite Term Rewriting Systems*, report nr. 7, Universität Bielefeld, 1992.

15. D.A. Plaisted, *The Undecidability of Self-Embedding for Term Rewriting Systems*, Information Processing Letters **20**, pp. 61–64, 1985.

16. H. Zantema, *Type Removal in Term Rewriting*, Proceedings of the 3rd International Workshop on Conditional Term Rewriting Systems, Pont-à-Mousson, Lecture Notes in Computer Science **656**, pp. 148–154, 1993.

17. H. Zantema, *Termination of Term Rewriting by Interpretation*, Proceedings of the 3rd International Workshop on Conditional Term Rewriting Systems, Pont-à-Mousson, Lecture Notes in Computer Science **656**, pp. 155–167, 1993.