Acta Cybernetica 14 (1999) 83-98.

On One-Pass Term Rewriting

Zoltán Fülöp * Eija Jurvanen † Magnus Steinby ‡
Sándor Vágvölgyi§

Dedicated to Professor Ferenc Gécseg on the occasion of his 60th birthday.

Abstract

Two restricted ways to apply a term rewriting system (TRS) to a tree are considered. When the *one-pass root-started* strategy is followed, rewriting starts from the root and continues stepwise towards the leaves without ever rewriting a part of the current tree produced in a previous rewrite step. *One-pass leaf-started rewriting* is defined similarly, but rewriting begins from the leaves. In the *sentential form inclusion problem* one asks whether all trees which can be obtained with a given TRS from the trees of some regular tree language T belong to another given regular tree language U, and in the *normal form inclusion problem* the same question is asked about the normal forms of T. We show that for a left-linear TRS these problems can be decided for both of our one-pass strategies. In all four cases the decision algorithm involves the construction of a suitable tree recognizer.

1 Introduction

In general, reducing a term with a term rewriting system (TRS) is a highly nondeterministic process in which many choices have to made, and usually no bound for the lengths of the possible reduction sequences can be given in advance. In this paper we consider two very restrictive strategies of term rewriting, *one-pass root-started rewriting* and *one-pass leaf-started rewriting*. When the former strategy is followed, rewriting starts at the root of the given term t and proceeds continuously towards the leaves without ever rewriting any part of the current term which has

*József Attila University, Department of Computer Science, H-6701 Szeged, P. O. Box 652, Hungary, Email: fulop@inf. u-szeged.hu

†Turku Centre for Computer Science, DataCity, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland, Email: jurvanen@utu.fi

‡Turku Centre for Computer Science, and Department of Mathematics, University of Turku, FIN-20014 Turku, Finland, Email: steinby@utu.fi

§József Attila University, Department of Applied Informatics, H-6701 Szeged, P. O. Box 652, Hungary, Email: vagvolgy@inf.u-szeged.hu

been produced in a previous rewrite step. When no more rewriting is possible, a *one-pass root-started normal form* of the original term t has been reached. Of course, such a normal form is not necessarily irreducible in the usual sense since a rewrite rule may apply either in the part already rewritten or to a subtree rooted at a position strictly below the nodes affected by the last rewriting steps. The leaf-started version is similar, but the rewriting is initiated at the leaves and proceeds towards the root. The requirement that rewriting should always concern positions immediately adjacent to parts of the term rewritten in previous steps distinguishes our rewriting strategies from the IO and OI rewriting schemes considered in [ES77, ES78] or [DDC87]. It also implies that the top-down and bottom-up cases are different even for a linear TRS.

Both of the one-pass modes of rewriting are defined formally by associating with any given TRS an auxiliary TRS in which a new separator mark restricts rewriting in the intended way.

Let us now describe the problems concerning one-pass rewriting considered in this paper. Since the problems involve regular tree languages, we find it convenient to use the terminology of the theory of tree languages. Let $\mathcal{R} = (\Sigma, R)$ be a TRS over a ranked alphabet Σ. For any Σ-tree language T ($\subseteq T_\Sigma$), we denote the sets of one-pass root-started sentential forms, one-pass root-started normal forms, one-pass leaf-started sentential forms and one-pass leaf-started normal forms of trees in T by $1r\mathrm{S}_\mathcal{R}(T)$, $1r\mathrm{N}_\mathcal{R}(T)$, $1\ell\mathrm{S}_\mathcal{R}(T)$ and $1\ell\mathrm{N}_\mathcal{R}(T)$, respectively. We show that all of the following inclusion problems, in which the input consists of a left-linear TRS $\mathcal{R} = (\Sigma, R)$ and two regular Σ-tree languages T_1 and T_2 (effectively given by tree recognizers, for example), are decidable.

The one-pass root-started sentential form inclusion problem: $1r\mathrm{S}_\mathcal{R}(T_1) \subseteq T_2$?

The one-pass root-started normal form inclusion problem: $1r\mathrm{N}_\mathcal{R}(T_1) \subseteq T_2$?

The one-pass leaf-started sentential form inclusion problem: $1\ell\mathrm{S}_\mathcal{R}(T_1) \subseteq T_2$?

The one-pass leaf-started normal form inclusion problem: $1\ell\mathrm{N}_\mathcal{R}(T_1) \subseteq T_2$?

In [GT95] the sentential form inclusion problem for ordinary sentential forms is called the second-order reachability problem and the problem is shown to be decidable for a TRS $\mathcal{R}$ which preserves recognizability, i.e. if the set $\mathrm{S}_\mathcal{R}(T)$ of sentential forms of the trees of any recognizable tree language T is also recognizable.

Many questions concerning term rewriting systems have been studied, and solved, using tree automata and tree languages; cf. [DDC87, DG89, Gil91, GT95, GV98, HH94, KT95, VG92], for example. Also here tree automata are used: in all four cases the decidability of the problem is proved by showing how one may construct from $\mathcal{R}$ and the given tree recognizers of T_1 and T_2 a new tree recognizer $\mathcal{C}$ such that the question can be answered by checking whether the tree language $T(\mathcal{C})$ recognized by $\mathcal{C}$ is empty. To simplify these constructions we introduce generalized top-down and generalized bottom-up tree recognizers. It is easy to see that both of these new types of recognizers recognize exactly the regular tree languages and that their emptiness problems are decidable.

The paper is essentially self-contained since all special concepts used, as well as many general notions, are completely defined. However, for further information about term rewriting systems and tree automata, we refer the reader to [Ave95], [DJ90], [Hue80], [GS84] and [GS97].

A conference version of this paper has appeared as [FJSV98]. This research was supported by the exchange program of the University of Turku and the József Attila University, and by the grants MKM 665/96 and FKFP 0095/97.

2 Preliminaries

Throughout this paper Σ is a *ranked alphabet*, i.e. a finite set of operation symbols. For each $m \geq 0$, the set of m-ary symbols in Σ is denoted by Σ_m. We say that Σ is *unary* if $\Sigma = \Sigma_1$, i.e., if every symbol $f \in \Sigma$ has rank 1. If Y is an alphabet disjoint with Σ, then the set $T_\Sigma(Y)$ of *Σ-terms with variables* in Y is the smallest set U of strings such that

(1) $Y \cup \Sigma_0 \subseteq U$ and

(2) $f(t_1, \ldots, t_m) \in U$ whenever $m \geq 1$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in U$.

Sometimes we consider terms $f(t_1, \ldots, t_m)$ with $m \geq 0$. For $m = 0$, this is interpreted as f. The set $T_\Sigma(\emptyset)$ of *ground Σ-terms* is denoted by T_Σ. Terms are viewed in the usual way as formal representations of trees, and we also call them trees. In particular, ground Σ-terms and subsets of T_Σ are called *Σ-trees* and *Σ-tree languages*, respectively.

The *height* $\mathrm{hg}(t)$ of a tree $t \in T_\Sigma(Y)$ is defined so that $\mathrm{hg}(t) = 0$ for $t \in Y \cup \Sigma_0$, and $\mathrm{hg}(t) = \max\{\mathrm{hg}(t_1), \ldots, \mathrm{hg}(t_m)\} + 1$ for $t = f(t_1, \ldots, t_m)$. The set $\mathrm{var}(t)$ ($\subseteq Y$) of variables appearing in t is also defined as usual (cf. [GS97], for example).

Let $X = \{x_1, x_2, \ldots\}$ be a countably infinite set of variables. For every $n \geq 0$, we put $X_n = \{x_1, \ldots, x_n\}$ and abbreviate $T_\Sigma(X_n)$ to $T_{\Sigma,n}$. A tree $t \in T_{\Sigma,n}$ is called *linear* if each x_i, $1 \leq i \leq n$, appears at most once in t.

We introduce a subset $\widetilde{T}_{\Sigma,n}$ of $T_{\Sigma,n}$ as follows. A tree $t \in T_{\Sigma,n}$ belongs to $\widetilde{T}_{\Sigma,n}$ if and only if each of $x_1, \ldots, x_n$ occurs in t exactly once and their left-to-right order is $x_1, \ldots, x_n$. Hence the elements of $\widetilde{T}_{\Sigma,n}$ are special linear trees. In addition, let $\widetilde{T}_{\Sigma,X} = \bigcup_{n=0}^{\infty} \widetilde{T}_{\Sigma,n}$. If $f \in \Sigma_m$, $m \geq 1$ and $t_1, \ldots, t_m \in \widetilde{T}_{\Sigma,X}$, then $\|f(t_1, \ldots, t_m)\|$ is the unique tree in $\widetilde{T}_{\Sigma,X}$ obtained from $f(t_1, \ldots, t_m)$ by renaming the variables.

A *substitution* $\sigma : X \to T_\Sigma(X)$ is extended to a mapping $\sigma : T_\Sigma(X) \to T_\Sigma(X)$ so that $\sigma(t) = f(\sigma(t_1), \ldots, \sigma(t_m))$ for any $t = f(t_1, \ldots, t_m)$. Hence, for any tree $t \in T_\Sigma(X)$, $\sigma(t)$ is the tree obtained from t when every occurrence of each variable $x \in \mathrm{var}(t)$ is replaced by the corresponding tree $\sigma(x)$. If, in particular, $t \in T_{\Sigma,n}$ and $\sigma(x_i) = t_i$ $(i = 1, 2, \ldots, n)$, we write $\sigma(t) = t[t_1, \ldots, t_n]$.

Let Σ be a ranked alphabet. A *term rewriting system* (TRS, for short) over Σ is a system $\mathcal{R} = (\Sigma, R)$, where R is a finite subset of $T_\Sigma(X) \times T_\Sigma(X)$ such that $\mathrm{var}(r) \subseteq \mathrm{var}(p)$ and $p \notin X$ for each $(p, r) \in R$. Note that since R is finite, $R \subseteq T_\Sigma(X_n) \times T_\Sigma(X_n)$ for some $n \geq 0$. The elements of R are called *(rewrite)*

rules and written in the form $p(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n)$ or just as $p \to r$. A rule $p \to r$ of a TRS $\mathcal{R}$ is called *ground* if both p and r are ground trees.

A TRS $\mathcal{R}$ induces a binary relation $\Rightarrow_{\mathcal{R}}$ in T_Σ defined as follows: for any t, $u \in T_\Sigma$, $t \Rightarrow_{\mathcal{R}} u$ if u is obtained from t by replacing an occurrence of a subtree of t of the form $p[t_1, \ldots, t_n]$ by $r[t_1, \ldots, t_n]$, where $p \to r \in R$ and $t_1, \ldots, t_n \in T_\Sigma$. The reflexive, transitive closure of $\Rightarrow_{\mathcal{R}}$ is denoted by $\Rightarrow^*_{\mathcal{R}}$. Hence $s \Rightarrow^*_{\mathcal{R}} t$ if and only if there exists a *reduction sequence*

$$t_0 \Rightarrow_{\mathcal{R}} t_1 \Rightarrow_{\mathcal{R}} \cdots \Rightarrow_{\mathcal{R}} t_n$$

in $\mathcal{R}$ such that $n \geq 0$, $t_0 = s$ and $t_n = t$.

A TRS $\mathcal{R}$ is *left-linear* if, for every rule $p \to r$ in R, p is a linear tree. A TRS is *in standard form* if for every rule $p \to r$ in it, $p \in \widetilde{T}_{\Sigma,n}$ for some $n \geq 0$. Obviously one can construct for every left-linear TRS $\mathcal{R}$ a standard form TRS $\mathcal{R}'$ such that $\Rightarrow_{\mathcal{R}} = \Rightarrow_{\mathcal{R}'}$.

We define the set of *left-hand sides* of a TRS $\mathcal{R} = (\Sigma, R)$ as $\mathrm{lhs}(\mathcal{R}) = \{ p \mid (\exists r)\, p \to r \in R \}$.

An element $s \in T_\Sigma$ is *irreducible* with respect to $\mathcal{R}$ if there exists no u such that $s \Rightarrow_{\mathcal{R}} u$. A Σ-tree s is a *normal form* of a Σ-tree t if $t \Rightarrow^*_{\mathcal{R}} s$ and s is irreducible. The set of all normal forms of a Σ-tree t is denoted by $\mathrm{N}_{\mathcal{R}}(t)$. The set of *sentential forms* of t is defined by

$$\mathrm{S}_{\mathcal{R}}(t) = \{ s \mid t \Rightarrow^*_{\mathcal{R}} s \}.$$

Moreover, for a tree language $T \subseteq T_\Sigma$, we put

$$\mathrm{S}_{\mathcal{R}}(T) = \bigcup_{t \in T} \mathrm{S}_{\mathcal{R}}(t),$$

$$\mathrm{N}_{\mathcal{R}}(T) = \bigcup_{t \in T} \mathrm{N}_{\mathcal{R}}(t).$$

A TRS $\mathcal{R}$ over Σ is *terminating* if there are no infinite reduction sequences $t_1 \Rightarrow_{\mathcal{R}} t_2 \Rightarrow_{\mathcal{R}} \cdots$, cf. [Hue80], [DJ90] and [Ave95].

Let us recall the two basic types of tree recognizers which both define the same family of recognizable tree languages. To facilitate our proofs, we also introduce certain generalized versions of both types. All of these recognizers can be defined conveniently as special term rewriting systems.

In a *top-down* Σ*-recognizer* $\mathcal{A} = (A, \Sigma, P, a_0)$

(1) A is a (finite) unary ranked alphabet of *states* such that $A \cap \Sigma = \emptyset$,

(2) P is a finite set of rewrite rules, the *transition rules*, each of the form

   (a) $a(f(x_1, \ldots, x_m)) \to f(a_1(x_1), \ldots, a_m(x_m))$, also written simply $a(f) \to f(a_1, \ldots, a_m)$, where $m > 0$, $f \in \Sigma_m$ and $a, a_1, \ldots, a_m \in A$, or of the form

   (b) $a(c) \to c$, where $c \in \Sigma_0$ and $a \in A$, and

(3) $a_0 \in A$ is the *initial state.*

We treat $\mathcal{A}$ as the TRS $(\Sigma \cup A, P)$ and the rewrite relation $\Rightarrow_{\mathcal{A}} \subseteq T_{\Sigma \cup A} \times T_{\Sigma \cup A}$ is defined accordingly. For each $a \in A$, let $T(\mathcal{A}, a) = \{ t \in T_\Sigma \mid a(t) \Rightarrow_{\mathcal{A}}^* t \}$. In particular, the tree language *recognized* by $\mathcal{A}$ is the set $T(\mathcal{A}) = T(\mathcal{A}, a_0)$.

A tree language $T \subseteq T_\Sigma$ is *recognizable* or *regular* if there exists a top-down Σ-recognizer $\mathcal{A}$ such that $T(\mathcal{A}) = T$. The class of all recognizable Σ-tree languages is denoted by REC_Σ.

In a *generalized top-down Σ-recognizer* $\mathcal{A} = (A, \Sigma, P, a_0)$

(1) A is a (finite) unary ranked alphabet of *states* such that $A \cap \Sigma = \emptyset$,

(2) P is a finite set of rewrite rules, the *transition rules* of $\mathcal{A}$, of the form

$$a(t(x_1, \ldots, x_n)) \to t[a_1(x_1), \ldots, a_n(x_n)],$$

where $n \geq 0$, $a, a_1, \ldots, a_n \in A$, and $t \in \widetilde{T}_{\Sigma,n}$, and

(3) $a_0 \in A$ is the *initial state.*

The rewrite relation $\Rightarrow_{\mathcal{A}} \subseteq T_{\Sigma \cup A} \times T_{\Sigma \cup A}$ and its reflexive, transitive closure $\Rightarrow_{\mathcal{A}}^*$ are defined in the natural way. The tree language *recognized* by $\mathcal{A}$ is the set

$$T(\mathcal{A}) = \{ t \in T_\Sigma \mid a_0(t) \Rightarrow_{\mathcal{A}}^* t \}.$$

It is clear that also the generalized top-down Σ-recognizers recognize exactly the regular Σ-tree languages.

Next we define tree recognizers which read their inputs from the leaves to the root.

A *bottom-up Σ-recognizer* is a quadruple $\mathcal{A} = (A, \Sigma, P, A_f)$, where

(1) A is a finite set of *states* of rank 0, $\Sigma \cap A = \emptyset$,

(2) P is a finite set of rewrite rules of the form

$$f(a_1, \ldots, a_m) \to a$$

with $m \geq 0$, $f \in \Sigma_m$, $a_1, \ldots, a_m$, $a \in A$, and

(3) A_f $(\subseteq A)$ is the set of *final states.*

We say that $\mathcal{A}$ is *total deterministic* if for all $f \in \Sigma_m$, $m \geq 0$, $a_1, \ldots, a_m \in A$, there is exactly one rule of the form $f(a_1, \ldots, a_m) \to a$.

When we treat $\mathcal{A}$ as the rewriting system $(\Sigma \cup A, P)$, the tree language recognized by it can be defined as the set

$$T(\mathcal{A}) = \{ t \in T_\Sigma \mid (\exists\, a \in A_f)\ t \Rightarrow_{\mathcal{A}}^* a \}.$$

For any bottom-up Σ-recognizer $\mathcal{A}$, one can effectively construct a total deterministic bottom-up Σ-recognizer $\mathcal{B}$ such that $T(\mathcal{A}) = T(\mathcal{B})$. If $\mathcal{A}$ is total deterministic, there is for each Σ-tree t exactly one state $a \in A$ such that $t \Rightarrow_{\mathcal{A}}^* a$.

A *generalized bottom-up* Σ-*recognizer* is a system $\mathcal{A} = (A, \Sigma, P, A_f)$, where A, Σ and A_f are as in the case of a bottom-up Σ-recognizer, but P is now a finite set of rewrite rules of the form

$$t[a_1, \ldots, a_n] \to a,$$

where $n \geq 0$, $t \in \widetilde{T}_{\Sigma,n}$ and $a_1, \ldots, a_n$, $a \in A$. Hence there may be rules of the form $a \to b$ in P, where $a, b \in A$. The tree language recognized by $\mathcal{A}$ is $T(\mathcal{A}) = \{\, t \in T_\Sigma \mid (\exists\, a \in A_f)\ t \Rightarrow_{\mathcal{A}}^* a \,\}$.

It is not hard to see that the class of Σ-tree languages recognized by generalized bottom-up Σ-recognizers is REC_Σ. Moreover, the emptiness problem "$T(\mathcal{A}) = \emptyset$?" is obviously decidable for both generalized top-down and generalized bottom-up recognizers.

3 One-pass Term Rewriting

The first of our two modes of one-pass rewriting may be described as follows.

Let $\mathcal{R} = (\Sigma, R)$ be a TRS and t the Σ-tree to be rewritten. The rewriting starts at the root of t and the portion first rewritten should include the root. Rewriting then proceeds as far as possible towards the leaves so that each rewrite step applies to a root segment of some maximal unprocessed subtree but never involves any part of the tree produced by a previous rewrite step. For the formal definition we associate with $\mathcal{R}$ a TRS in which a new special symbol forces this mode of rewriting.

Definition 3.1. The *one-pass root-started TRS* associated with a given TRS $\mathcal{R} = (\Sigma, R)$ is the TRS $\mathcal{R}_\# = (\Sigma \cup \{\#\}, R_\#)$, where $\#$ is a new unary symbol, the *separator mark*, and $R_\#$ is the set of all rewrite rules

$$\#(p(x_1, \ldots, x_n)) \to r[\#(x_1), \ldots, \#(x_n)]$$

obtained from a rule $p(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n)$ in R by adding $\#$ to the root of the left-hand side and above the variables in the right-hand side.

Example 3.2. Suppose that Σ consists of the binary symbol f, the unary symbol g and the nullary symbol c. If $\mathcal{R} = (\Sigma, R)$ is the TRS, where

$$R = \{\, f(g(x_1), x_2) \to f(x_1, g(x_2)),\ g(x_1) \to f(c, x_1),\ g(x_1) \to g(c) \,\},$$

then the rule set of the associated one-pass root-started TRS $\mathcal{R}_\#$ is

$$\begin{aligned} R_\# = \{\, &\#(f(g(x_1), x_2)) \to f(\#(x_1), g(\#(x_2))), \\ &\#(g(x_1)) \to f(c, \#(x_1)),\ \#(g(x_1)) \to g(c) \,\}. \end{aligned}$$

For any TRS $\mathcal{R}$, the associated one-pass root-started TRS $\mathcal{R}_\#$ is terminating. For recovering the one-pass root-started reduction sequences of $\mathcal{R}$ from the reduction sequences of $\mathcal{R}_\#$, we introduce the tree homomorphism $\delta : T_{\Sigma\cup\{\#\}} \to T_\Sigma$ which erases the separator marks:

(1) $\delta(c) = c$ for any $c \in \Sigma_0$;

(2) $\delta(\#(t)) = \delta(t)$ for any $t \in T_{\Sigma\cup\{\#\}}$;

(3) $\delta(t) = f(\delta(t_1), \ldots, \delta(t_m))$ for $t = f(t_1, \ldots, t_m)$, where $m > 0$, $f \in \Sigma_m$ and $t_i \in T_{\Sigma\cup\{\#\}}$.

If $t \in T_\Sigma$ and

$$\#(t) \Rightarrow_{\mathcal{R}_\#} t_1 \Rightarrow_{\mathcal{R}_\#} t_2 \Rightarrow_{\mathcal{R}_\#} \ldots \Rightarrow_{\mathcal{R}_\#} t_k$$

is a reduction sequence with $\mathcal{R}_\#$, then

$$t \Rightarrow_{\mathcal{R}} \delta(t_1) \Rightarrow_{\mathcal{R}} \delta(t_2) \Rightarrow_{\mathcal{R}} \ldots \Rightarrow_{\mathcal{R}} \delta(t_k)$$

is a *one-pass root-started reduction sequence* with $\mathcal{R}$. The terms $t, \delta(t_1), \ldots, \delta(t_k)$ are called *one-pass root-started sentential forms* of t in $\mathcal{R}$. If t_k is irreducible in $\mathcal{R}_\#$, then $\delta(t_k)$ is a *one-pass root-started normal form* of t in $\mathcal{R}$. The sets of all one-pass root-started sentential forms and normal forms of a Σ-tree t are denoted by $1r\mathrm{S}_{\mathcal{R}}(t)$ and $1r\mathrm{N}_{\mathcal{R}}(t)$, respectively. This notation is extended to sets of Σ-trees in the natural way.

Note that for any TRS $\mathcal{R} = (\Sigma, R)$ and any $t \in T_\Sigma$, the sets $1r\mathrm{S}_{\mathcal{R}}(t)$ and $1r\mathrm{N}_{\mathcal{R}}(t)$ are finite and effectively computable but that $1r\mathrm{S}_{\mathcal{R}}(T)$ and $1r\mathrm{N}_{\mathcal{R}}(T)$ are not necessarily regular even for a regular Σ-tree language T.

The mode of one-pass rewriting which starts from the leaves is also formalized by associating with the given TRS a special one-pass TRS. This TRS is constructed in two stages. First we add to the original TRS all rules obtained by instantiating any variables in the original rules as constants. In the second stage the extended TRS is turned into a one-pass TRS by introducing a separator mark and by labelling the symbols of the right-hand sides so that the rewriting cannot be restarted from leaves which have already been processed.

Definition 3.3. Let $\mathcal{R} = (\Sigma, R)$ be a TRS. First we extend R to the set R_e of all rules

$$p[y_1, \ldots, y_n] \to q[y_1, \ldots, y_n]$$

such that $p(x_1, \ldots, x_n) \to q(x_1, \ldots, x_n) \in R$, with $p, q \in T_{\Sigma,n}$, and for each i, $1 \leq i \leq n$, either $y_i \in X$ or $y_i \in \Sigma_0$, and $p[y_1, \ldots, y_n] \in \tilde{T}_{\Sigma,X}$. Now let $\Sigma' = \{ f' \mid f \in \Sigma \}$ be a disjoint copy of Σ such that for any $f \in \Sigma$, f and f' have the same rank. The *one-pass leaf-started TRS* associated with $\mathcal{R}$ is the TRS $\mathcal{R}^\# = (\Sigma \cup \Sigma' \cup \{\#\}, R^\#)$, where $\#$ is a new unary symbol, the *separator mark*, and $R^\#$ consists of all rules

$$p[\#(x_1), \ldots, \#(x_n)] \to \#(r'(x_1, \ldots, x_n)),$$

where $p \to r \in R_e$, with $p, r \in T_{\Sigma,n}$, and r' is obtained from r by replacing every symbol $f \in \Sigma$ by the corresponding symbol f' in Σ'.

For every left-linear TRS $\mathcal{R}$, the one-pass TRS $\mathcal{R}^\#$ is in standard form.

Example 3.4. Let $\Sigma = \{f, g, c\}$, where f is binary, g unary and c nullary, and let $\mathcal{R} = (\Sigma, R)$ be the TRS with

$$R = \{\, f(g(x_1), x_2) \to f(x_1, g(x_2)),\ g(c) \to f(c, c) \,\}.$$

Then $\Sigma' = \{f', g', c'\}$ and the one-pass leaf-started TRS associated with R is the TRS $\mathcal{R}^\# = (\Sigma \cup \Sigma' \cup \{\#\}, R^\#)$ where $R^\#$ is

$$\begin{aligned} R^\# = \{\, & f(g(\#(x_1)), \#(x_2)) \to \#(f'(x_1, g'(x_2))), \\ & f(g(c), \#(x_1)) \to \#(f'(c', g'(x_1))), \\ & f(g(\#(x_1)), c) \to \#(f'(x_1, g'(c'))), \\ & f(g(c), c) \to \#(f'(c', g'(c'))),\ g(c) \to \#(f'(c', c')) \,\}. \end{aligned}$$

It is clear that the TRS $\mathcal{R}_e = (\Sigma, R_e)$ is always equivalent to the original TRS $\mathcal{R}$ in the sense that $\Rightarrow_{\mathcal{R}_e} = \Rightarrow_{\mathcal{R}}$. The connection between $\mathcal{R}$ and $\mathcal{R}^\#$ is the following. The reduction sequences of $\mathcal{R}^\#$ represent such reduction sequences of $\mathcal{R}$ which start at the leaves of a term and proceed towards the root of it in such a way that symbols introduced by a previous rewrite step never form a part of the left-hand side of the rule applied next. At the same time the rewrite places are joined together by the separator mark. Moreover, $\mathcal{R}^\#$ can make only one pass over the term because the left-hand sides and the right-hand sides of its rules are over disjoint ranked alphabets. The one-pass reduction sequence of $\mathcal{R}$ represented by a reduction sequence of $\mathcal{R}^\#$ is recovered by applying a tree homomorphism which erases the #-marks and the primes from the symbols $f' \in \Sigma'$. Formally we define $\delta\colon T_{\Sigma \cup \Sigma' \cup \{\#\}} \to T_\Sigma$ so that

(1) $\delta(c') = \delta(c) = c$ for every $c \in \Sigma_0$;

(2) $\delta(\#(t)) = \delta(t)$ for every $t \in T_{\Sigma \cup \Sigma' \cup \{\#\}}$;

(3) $\delta(f(t_1, \ldots, t_m)) = \delta(f'(t_1, \ldots, t_m)) = f(\delta(t_1), \ldots, \delta(t_m))$ for any $f \in \Sigma_m$, $m \geq 1$, and $t_1, \ldots, t_m \in T_{\Sigma \cup \Sigma' \cup \{\#\}}$.

Then each reduction sequence

$$t \Rightarrow_{\mathcal{R}^\#} t_1 \Rightarrow_{\mathcal{R}^\#} t_2 \Rightarrow_{\mathcal{R}^\#} \ldots \Rightarrow_{\mathcal{R}^\#} t_k$$

with $\mathcal{R}^\#$ starting from a Σ-tree t yields the *one-pass leaf-started reduction sequence*

$$t \Rightarrow_{\mathcal{R}} \delta(t_1) \Rightarrow_{\mathcal{R}} \delta(t_2) \Rightarrow_{\mathcal{R}} \ldots \Rightarrow_{\mathcal{R}} \delta(t_k)$$

with $\mathcal{R}$. Moreover, the *set of one-pass leaf-started sentential forms* and the *set of one-pass leaf-started normal forms* of t with respect to $\mathcal{R}$ are defined by

$$1\ell\mathrm{S}_{\mathcal{R}}(t) = \{\, \delta(s) \mid s \in \mathrm{S}_{\mathcal{R}^\#}(t) \,\},$$

and

$$1\ell\mathrm{N}_{\mathcal{R}}(t) = \{\, \delta(s) \mid s \in \mathrm{N}_{\mathcal{R}^\#}(t) \,\},$$

respectively. Finally, for a tree language $T \subseteq T_\Sigma$, we put

$$1\ell S_\mathcal{R}(T) = \bigcup_{t \in T} 1\ell S_\mathcal{R}(t),$$

and

$$1\ell N_\mathcal{R}(T) = \bigcup_{t \in T} 1\ell N_\mathcal{R}(t).$$

The extended TRS $\mathcal{R}_e$ may seem redundant, but without the new rules in $R_e \setminus R$ many natural one-pass leaf-started rewriting sequences would be missed. In particular, if $\mathcal{R}$ contains no ground rules, such as the rule $g(c) \to f(c,c)$, no non-trivial one-pass leaf-started rewriting sequence could be initiated since the left-hand sides of all rules of $\mathcal{R}^\#$ would contain the symbol $\#$. Hence we would have $1\ell S_\mathcal{R}(t) = 1\ell N_\mathcal{R}(t) = \{t\}$ for every $t \in T_\Sigma$.

4 The one-pass root-started inclusion problems

First we define the one-pass root-started normal form inclusion problem for a TRS $\mathcal{R} = (\Sigma, R)$. It is assumed that the recognizable tree languages are given in the form of tree recognizers.

Theorem 4.1. For any left-linear TRS $\mathcal{R} = (\Sigma, R)$, the following *one-pass root-started normal form inclusion problem* is decidable.
Instance: Recognizable Σ-tree languages T_1 and T_2.
Question: $1rN_\mathcal{R}(T_1) \subseteq T_2$?

For proving Theorem 4.1, we need the following auxiliary notations. For a set A of unary symbols such that $A \cap \Sigma = \emptyset$ and any alphabet Y, let $T_\Sigma(A(Y))$ be the least subset T of $T_{\Sigma \cup A}(Y)$ for which

(1) $\Sigma_0 \subseteq T$,

(2) $a(y) \in T$ for all $a \in A$, $y \in Y$, and

(3) $m \geq 1$, $f \in \Sigma_m$, $t_1, \ldots, t_m \in T$ implies $f(t_1, \ldots, t_m) \in T$.

Let $\mathcal{A} = (A, \Sigma, P, a_0)$ be a top-down Σ-recognizer. For any $a \in A$, $n \geq 0$ and any $t \in T_{\Sigma,n}$, the set $\mathcal{A}(a,t)$ $(\subseteq T_\Sigma(A(X_n)))$ is defined so that

(1) $\mathcal{A}(a, x_i) = \{a(x_i)\}$ for all $x_i \in X_n$,

(2) for $c \in \Sigma_0$, $\mathcal{A}(a,c) = \{c\}$ if $a(c) \to c \in P$, and $\mathcal{A}(a,c) = \emptyset$ otherwise, and

(3) for $m \geq 1$, $t = f(t_1, \ldots, t_m)$,

$$\mathcal{A}(a,t) = \{\, f(s_1, \ldots, s_m) \mid s_1 \in \mathcal{A}(a_1, t_1), \ldots, s_m \in \mathcal{A}(a_m, t_m), \\ a(f) \to f(a_1, \ldots, a_m) \in P \,\}.$$

For any $s \in T_\Sigma(A(X))$ and any variable $x_i \in X$, we denote by $\mathrm{st}(s, x_i)$ the set of states $b \in A$ such that $b(x_i)$ appears as a subterm in s.

Clearly, $\mathcal{A}(a,t) \neq \emptyset$ if and only if there is a computation of $\mathcal{A}$ which starts at the root of t and continues successfully along all paths to the leaves of t, and moreover, if $\mathcal{A}$ reaches in a state b ($\in A$) a leaf labelled by a nullary symbol c, then the rule $b(c) \to c$ is in P. Each term s in $\mathcal{A}(a,t)$ represents the situation when such a successful computation has been completed so that all leaves labelled with a nullary symbol have also been processed. If $t \in \widetilde{T}_{\Sigma,n}$, then every $s \in \mathcal{A}(a_0,t)$ is of the form $s = t[a_1(x_1), \ldots, a_n(x_n)]$ and for any $t_1, \ldots, t_n \in T_\Sigma$, the tree s appears in a computation of $\mathcal{A}$ on $t[t_1, \ldots, t_n]$ of the form

$$a_0(t[t_1, \ldots, t_n]) \Rightarrow_{\mathcal{A}}^* t[a_1(t_1), \ldots, a_n(t_n)] = s[t_1, \ldots, t_n] \Rightarrow_{\mathcal{A}}^* \cdots$$

in which each subterm t_i is processed starting in the corresponding state a_i. However, if t is not linear, then a variable x_i may appear in a term $s \in \mathcal{A}(a_0,t)$ together with more than one state symbol, and then the corresponding subterm t_i should be accepted by a computation starting with each $a \in \mathrm{st}(s, x_i)$.

Proof of Theorem 4.1. Consider a left-linear TRS $\mathcal{R} = (\Sigma, R)$ and any recognizable Σ-tree languages T_1 and T_2. Let $\mathcal{A} = (A, \Sigma, P_1, a_0)$ and $\mathcal{B} = (B, \Sigma, P_2, b_0)$ be top-down Σ-recognizers for which $T(\mathcal{A}) = T_1$ and $T(\mathcal{B}) = T_2^c$ ($= T_\Sigma \setminus T_2$). We construct a generalized top-down Σ-recognizer $\mathcal{C}$ such that for any Σ-term t,

$$t \in T(\mathcal{C}) \quad \text{iff} \quad t \in T(\mathcal{A}) \text{ and } s \in T(\mathcal{B}) \text{ for some } s \in 1r\mathrm{N}_{\mathcal{R}}(t). \qquad (*)$$

Then $1r\mathrm{N}_{\mathcal{R}}(T_1) \subseteq T_2$ if and only if $T(\mathcal{C}) = \emptyset$, and the latter condition is decidable.

Let $\mathcal{C} = (C, \Sigma, P, (a_0, \{b_0\}))$ be the generalized top-down Σ-recognizer with the state set

$$C = (A \times \wp(B)) \cup (\bar{A} \times \wp(B)),$$

where $\wp(B)$ is the power set of B and $\bar{A} = \{\, \bar{a} \mid a \in A \,\}$ is a disjoint copy of A, and the set P of transition rules is defined as follows. The rules are of three different types.

Type 1. If $p(x_1, \ldots, x_n) \to r(x_1, \ldots, x_n)$ is a rule in R and $(a, H) \in A \times \wp(B)$, where $H = \{b_1, \ldots, b_k\}$, we include in P any rule

$$(a,H)(p(x_1, \ldots, x_n)) \to p[(a_1, H_1)(x_1), \ldots, (a_n, H_n)(x_n)]$$

where

(a) $p[a_1(x_1), \ldots, a_n(x_n)] \in \mathcal{A}(a,p)$ and

(b) there are terms $s_1 \in \mathcal{B}(b_1, r)$, $\ldots$, $s_k \in \mathcal{B}(b_k, r)$ such that, for all $i = 1, \ldots, n$,

$$H_i = \mathrm{st}(s_1, x_i) \cup \cdots \cup \mathrm{st}(s_k, x_i).$$

For $H = \emptyset$ ($k = 0$), this is interpreted to mean that $H_1 = \cdots = H_n = \emptyset$ should hold, and if $p \to r$ is a ground rule ($n = 0$), we include $(a, H)(p) \to p$ in P iff $a(p) \Rightarrow_{\mathcal{A}}^* p$ and $b_i(r) \Rightarrow_{\mathcal{B}}^* r$ for all $i = 1, \ldots, k$.

Type 2. Let NI be the set of all terms $q \in \tilde{T}_{\Sigma,X}$ such that

(1) $\mathrm{hg}(q) \leq \max\{\, \mathrm{hg}(p) \mid p \in \mathrm{lhs}(\mathcal{R}) \,\} + 1$, and

(2) $\sigma(q) \neq \sigma'(p)$ for all $p \in \mathrm{lhs}(\mathcal{R})$ and all substitutions σ and σ'.

For each $p(x_1, \ldots, x_n) \in NI$ and any $(a, H) \in A \times \wp(B)$ with $H = \{b_1, \ldots, b_k\}$, we include in P any rule

$$(a, H)(p(x_1, \ldots, x_n)) \to p[(\bar{a}_1, H_1)(x_1), \ldots, (\bar{a}_n, H_n)(x_n)],$$

where

(a) $p[a_1(x_1), \ldots, a_n(x_n)] \in \mathcal{A}(a, p)$, and

(b) there are terms $s_1 \in \mathcal{B}(b_1, p), \ldots, s_k \in \mathcal{B}(b_k, p)$ such that, for all $i = 1, \ldots, n$,

$$H_i = \mathrm{st}(s_1, x_i) \cup \cdots \cup \mathrm{st}(s_k, x_i).$$

The cases $H = \emptyset$ and $n = 0$ are treated similarly as above.

Type 3. For each $(\bar{a}, H) \in \bar{A} \times \wp(B)$, where $H = \{b_1, \ldots, b_k\}$, we add to P rules as follows.

(1) For $c \in \Sigma_0$, we include in P the rule

$$(\bar{a}, H)(c) \to c$$

if and only if $a(c) \to c$ is in P_1 and P_2 contains $b_i(c) \to c$ for every $b_i \in H$.

(2) For $f \in \Sigma_m$, $m > 0$, we add to P all rules

$$(\bar{a}, H)(f(x_1, \ldots, x_m)) \to f((\bar{a}_1, H_1)(x_1), \ldots, (\bar{a}_m, H_m)(x_m))$$

where

(a) $a(f(x_1, \ldots, x_m)) \to f(a_1(x_1), \ldots, a_m(x_m))$ is in P_1, and

(b) there are rules $b_i(f(x_1, \ldots, x_m)) \to f(b_{i1}(x_1), \ldots, b_{im}(x_m))$ ($i = 1, \ldots, k$) such that for each $j = 1, \ldots, m$, $H_j = \{b_{1j}, \ldots, b_{kj}\}$.

We can show that $\mathcal{C}$ has the property described in (*). If $t \in T(\mathcal{C})$, then $(a_0, \{b_0\})(t) \Rightarrow_{\mathcal{C}}^* t$ and this derivation can be split into two parts

$$(a_0, \{b_0\})(t) \Rightarrow_{\mathcal{C}}^* \tilde{t}[(a_1, H_1)(t_1), \ldots, (a_n, H_n)(t_n)] \Rightarrow_{\mathcal{C}}^* \tilde{t}[t_1, \ldots, t_n] = t, \qquad (**)$$

where $n \geq 0$, $t \in \tilde{T}_{\Sigma,n}$ and, for every $1 \leq i \leq n$, $t_i \in T_\Sigma$ and $(a_i, H_i) \in A \times \wp(B)$. In the first part of (**) only Type 1 rules are used, and hence $\tilde{t}[a_1(x_1), \ldots, a_n(x_n)] \in \mathcal{A}(a_0, \tilde{t})$. Moreover, for some $k \geq 0$, $\tilde{s} \in \tilde{T}_{\Sigma,k}$, and $s_1, \ldots, s_k \in T_\Sigma$,

$$\#(t) = \#(\tilde{t}[t_1, \ldots, t_n]) \Rightarrow_{\mathcal{R}_\#} \cdots \Rightarrow_{\mathcal{R}_\#} \tilde{s}[\#(s_1), \ldots, \#(s_k)] = s,$$

where every s_j is a copy of exactly one of the t_i. (Of course, s_j may be equal to more than one t_i.) For each $i = 1, \ldots, n$, let $K(i) = \{\, j \mid s_j \text{ is a copy of } t_i \,\}$. Then for some $u \in \mathcal{B}(b_0, \tilde{s})$, $H_i = \bigcup\{\, \mathrm{st}(u, x_j) \mid j \in K(i) \,\}$ for all $i = 1, \ldots, n$.

In the second part of (**), it is first checked using Type 2 rules that $\tilde{s}[s_1, \ldots, s_k] \in 1r\mathrm{N}_\mathcal{R}(t)$, and the computations $(a_i, H_i)(t_i) \Rightarrow_\mathcal{C}^* t_i$ are finished using Type 3 rules. That means for every $i = 1, \ldots, n$, that (a) $t_i \in T(\mathcal{A}, a_i)$ and (b) $t_i \in T(\mathcal{B}, b)$ for all $b \in H_i$. Therefore

$$a_0(t) \Rightarrow_\mathcal{A}^* \tilde{t}[a_1(t_1), \ldots, a_n(t_n)] \Rightarrow_\mathcal{A}^* \tilde{t}[t_1, \ldots, t_n] = t$$

and there are $b_1, \ldots, b_k \in B$ such that

$$b_0(\tilde{s}[s_1, \ldots, s_k]) \Rightarrow_\mathcal{B}^* \tilde{s}[b_1(s_1), \ldots, b_k(s_k)] \Rightarrow_\mathcal{B}^* \tilde{s}[s_1, \ldots, s_k].$$

The converse of (*) can be proved similarly. □

The corresponding result for sentential forms can be proved by modifying suitably the definition of the recognizer $\mathcal{C}$.

Theorem 4.2. For any left-linear TRS $\mathcal{R} = (\Sigma, R)$, the following *one-pass root-started sentential form inclusion problem* is decidable.

Instance: Recognizable Σ-tree languages T_1 and T_2.

Question: $1r\mathrm{S}_\mathcal{R}(T_1) \subseteq T_2$? □

Proof. Consider a left-linear TRS $\mathcal{R} = (\Sigma, R)$ and any recognizable Σ-tree languages T_1 and T_2. Let $\mathcal{A} = (A, \Sigma, P_1, a_0)$ and $\mathcal{B} = (B, \Sigma, P_2, b_0)$ be top-down Σ-recognizers for which $T(\mathcal{A}) = T_1$ and $T(\mathcal{B}) = T_2^c$. We shall now construct a generalized top-down Σ-recognizer $\mathcal{D}$ such that for any Σ-term t,

$$t \in T(\mathcal{D}) \quad \text{iff} \quad t \in T(\mathcal{A}) \text{ and } s \in T(\mathcal{B}) \text{ for some } s \in 1r\mathrm{S}_\mathcal{R}(t).$$

Then $1r\mathrm{S}_\mathcal{R}(T_1) \subseteq T_2$ if and only if $T(\mathcal{D}) = \emptyset$, and the latter condition is decidable.

Let $\mathcal{D} = (C, \Sigma, P', (a_0, \{b_0\}))$ be the generalized top-down Σ-recognizer, where the state set is that of the recognizer $\mathcal{C}$ used in the proof of Theorem 4.1 and the set P' of transition rules is defined as follows. All rules of $\mathcal{C}$ of Type 1 or of Type 3 will be included also in P'. The rules of Type 2 are replaced by the rules $(a, H)(c) \to c$ and

$$(a, H)(f(x_1, \ldots, x_m)) \to f((\bar{a}_1, H_1)(x_1), \ldots, (\bar{a}_m, H_m)(x_m))$$

which are identical to the Type 3 rules of $\mathcal{C}$ except that the a's have no bars in the left-hand sides.

The recognizer $\mathcal{D}$ is almost the same as $\mathcal{C}$, but it may stop following the chosen one-pass root-started reduction of $\mathcal{R}$ at any moment and switch to states in which

it checks whether the input tree is in $T(\mathcal{A})$ and whether the one-pass root-started sentential form produced by the corresponding simulated reduction sequence of $\mathcal{R}$ is in $T(\mathcal{B})$. □

5 The one-pass leaf-started inclusion problems

First we consider the one-pass leaf-started sentential form inclusion problem. Again the tree languages are assumed to be given in the form of tree recognizers.

Theorem 5.1. For any left-linear TRS $\mathcal{R} = (\Sigma, R)$, the following *one-pass leaf-started sentential form inclusion problem* is decidable.
Instance: Recognizable Σ-tree languages T_1 and T_2.
Question: $1\ell\,\mathrm{S}_{\mathcal{R}}(T_1) \subseteq T_2$?

Proof. Let $\mathcal{A} = (A, \Sigma, P_1, A_f)$ and $\mathcal{B} = (B, \Sigma, P_2, B_f)$ be bottom-up Σ-recognizers that recognize the tree languages T_1 and T_2, respectively. We may assume that $\mathcal{B}$ is total deterministic. We construct a generalized bottom-up Σ-recognizer $\mathcal{C}$ such that $T(\mathcal{C}) = \emptyset$ if and only if $1\ell\,\mathrm{S}_{\mathcal{R}}(T_1) \subseteq T_2$. This recognizer $\mathcal{C} = (C, \Sigma, P, C_f)$ is defined as follows.

(1) Let $C = (A \times B) \cup (\bar{A} \times \bar{B})$, where $\bar{A} = \{\, \bar{a} \mid a \in A \,\}$ and $\bar{B} = \{\, \bar{b} \mid b \in B \,\}$.

(2) The set P consists of the following rules which are of three different types.

Type 1. For every rule $p \to r \in R_e$ with $p,\ r \in T_{\Sigma,n}$, $n \geq 0$, and for all states $a_1, \ldots, a_n$, $a \in A$, $b_1, \ldots, b_n$, $b \in B$ such that $p[a_1, \ldots, a_n] \Rightarrow_{\mathcal{A}}^* a$ and $r[b_1, \ldots, b_n] \Rightarrow_{\mathcal{B}}^* b$, let P contain the rule $p[(a_1, b_1), \ldots, (a_n, b_n)] \to (a, b)$.

Type 2. For all $a \in A$ and $b \in B$, let $(a, b) \to (\bar{a}, \bar{b})$ be in P.

Type 3. For all $f \in \Sigma_m$ with $m \geq 0$, all $a_1, \ldots, a_m$, $a \in A$ and $b_1, \ldots, b_m$, $b \in B$ such that $f(a_1, \ldots, a_m) \to a \in P_1$ and $f(b_1, \ldots, b_m) \to b \in P_2$, let P contain the rule $f((\bar{a}_1, \bar{b}_1), \ldots, (\bar{a}_m, \bar{b}_m)) \to (\bar{a}, \bar{b})$.

(3) Let $C_f = \{\, \bar{a} \mid a \in A_f \,\} \times \{\, \bar{b} \mid b \in (B \setminus B_f) \,\}$.

The way $\mathcal{C}$ processes a Σ-tree t can be described as follows. First $\mathcal{C}$, using rules of Type 1, follows some one-pass leaf-started rewriting sequences by $\mathcal{R}$ on subtrees of t computing in the first components of its states the evaluations by $\mathcal{A}$ of these subtrees and in the second components the evaluations by $\mathcal{B}$ of the translations of the subtrees produced by these one-pass leaf-started rewriting sequences. At any time $\mathcal{C}$ may switch by rules of Type 2 to a mode in which it by rules of Type 3 computes in the first components of its states the evaluation of t by $\mathcal{A}$ and in the second components the evaluation by $\mathcal{B}$ of the one-pass leaf-started sentential form of t produced by $\mathcal{R}$ when the rewriting sequences on the subtrees are combined. This means that for any $t \in T_\Sigma$, $a \in A$ and $b \in B$,

$$t \Rightarrow_{\mathcal{C}}^* (\bar{a}, \bar{b}) \quad \text{iff} \quad t \Rightarrow_{\mathcal{A}}^* a \text{ and } s \Rightarrow_{\mathcal{B}}^* b \text{ for some } s \in 1\ell\,\mathrm{S}_{\mathcal{R}}(t).$$

By recalling the definition of C_f we see that the above equivalence implies immediately that $T(C) = \emptyset$ if and only if $1\ell\,\mathrm{S}_{\mathcal{R}}(T_1) \subseteq T_2$, as required. □

Finally we consider the one-pass leaf-started normal form inclusion problem.

Theorem 5.2. For any left-linear TRS $\mathcal{R} = (\Sigma, R)$, the following *one-pass leaf-started normal form inclusion problem* is decidable.

Instance: Recognizable Σ-tree languages T_1 and T_2.

Question: $1\ell\,\mathrm{N}_{\mathcal{R}}(T_1) \subseteq T_2$?

Proof. Let $\mathcal{A} = (A, \Sigma, P_1, A_f)$ and $\mathcal{B} = (B, \Sigma, P_2, B_f)$ be total deterministic bottom-up Σ-recognizers such that $T(\mathcal{A}) = T_1$ and $T(\mathcal{B}) = T_2$. We shall construct a generalized bottom-up Σ-recognizer C such that $T(C) = \emptyset$ if and only if $1\ell\,\mathrm{N}_{\mathcal{R}}(T_1) \subseteq T_2$.

Let $mx = \max\{\,\mathrm{hg}(p) \mid p \in \mathrm{lhs}(\mathcal{R}_e)\,\}$ and let $T_{mx} = \{\,t \in \widetilde{T}_{\Sigma,X} \mid \mathrm{hg}(t) \leq mx\,\}$. Now we define $C = (C, \Sigma, P, C_f)$ as follows.

(1) $C = (A \times B) \cup (\bar{A} \times \bar{B} \times (T_{mx} \cup \{ok\}))$, where $\bar{A} = \{\,\bar{a} \mid a \in A\,\}$ and $\bar{B} = \{\,\bar{b} \mid b \in B\,\}$.

(2) P consists of the following rules which are of five different types.

Type 1. For every rule $p \to r \in R_e$ with $p, r \in T_{\Sigma,n}$, and any states $a_1, \ldots, a_n$, $a \in A$, $b_1, \ldots, b_n$, $b \in B$ such that $p[a_1, \ldots, a_n] \Rightarrow_{\mathcal{A}}^* a$ and $r[b_1, \ldots, b_n] \Rightarrow_{\mathcal{B}}^* b$, let P contain the rule $p[(a_1, b_1), \ldots, (a_n, b_n)] \to (a, b)$.

Type 2. For all $a \in A$ and $b \in B$, let $(a, b) \to (\bar{a}, \bar{b}, x_1)$ be in P.

Type 3. For any $f \in \Sigma_m$ with $m \geq 0$, $a_1, \ldots, a_m$, $a \in A$, $b_1, \ldots, b_m$, $b \in B$ and $u_1, \ldots, u_m \in T_{mx}$ such that $f(a_1, \ldots, a_m) \to a \in P_1$, $f(b_1, \ldots, b_m) \to b \in P_2$, $u = \|f(u_1, \ldots, u_m)\|$ and $u \in T_{mx} \setminus \mathrm{lhs}(\mathcal{R}_e)$, let P contain the rule $f((\bar{a}_1, \bar{b}_1, u_1), \ldots, (\bar{a}_m, \bar{b}_m, u_m)) \to (\bar{a}, \bar{b}, u)$.

In case $m = 0$, the rule has the form $f \to (\bar{a}, \bar{b}, f)$.

Type 4. For any $f \in \Sigma_m$ with $m \geq 0$, $a_1, \ldots, a_m$, $a \in A$, $b_1, \ldots, b_m$, $b \in B$ and $u_1, \ldots, u_m \in T_{mx}$ such that $f(a_1, \ldots, a_m) \to a \in P_1$, $f(b_1, \ldots, b_m) \to b \in P_2$ and $\|f(u_1, \ldots, u_m)\| \notin T_{mx}$, let P contain the rule $f((\bar{a}_1, \bar{b}_1, u_1), \ldots, (\bar{a}_m, \bar{b}_m, u_m)) \to (\bar{a}, \bar{b}, ok)$.

Type 5. For any $f \in \Sigma_m$ with $m \geq 1$, $a_1, \ldots, a_m$, $a \in A$, $b_1, \ldots, b_m$, $b \in B$, and sequence $y_1, \ldots, y_m \in T_{mx} \cup \{ok\}$ such that $ok \in \{y_1, \ldots, y_m\}$, $f(a_1, \ldots, a_m) \to a \in P_1$, $f(b_1, \ldots, b_m) \to b \in P_2$, let P contain the rule $f((\bar{a}_1, \bar{b}_1, y_1), \ldots, (\bar{a}_m, \bar{b}_m, y_m)) \to (\bar{a}, \bar{b}, ok)$.

(3) Let $C_f = \{\,\bar{a} \mid a \in A_f\,\} \times \{\,\bar{b} \mid b \in (B \setminus B_f)\,\} \times (T_{mx} \cup \{ok\})$.

The recognizer C evaluates an input tree t by rules of Type 1 so that it simulates both the computation by $\mathcal{A}$ on subtrees of t and the computation by $\mathcal{B}$ on the translations of those subtrees produced by $\mathcal{R}$. Then C checks that the sentential form produced from t by the computation with $\mathcal{R}$ is a normal form. For this C

switches by rules of Type 2 into a mode in which it, by rules of Type 3, forms in the third components of its states sufficiently large portions of the tree above the nodes where the rewriting with $\mathcal{R}$ ended. If one of these parts turns out to be the left-hand side of a rule of $\mathcal{R}$, then the evaluation by $\mathcal{C}$ stops and t is rejected. If not, then the sentential form produced by $\mathcal{R}$ is a normal form, which is acknowledged by rules of Type 4 by putting ok in the third components of the corresponding states of $\mathcal{C}$. Then $\mathcal{C}$ evaluates t in the same way as it was done in the proof of Theorem 5.1 by rules of Type 3.

This means that for any $t \in T_\Sigma$, $a \in A$, $b \in B$, and $y \in T_{mx} \cup \{ok\}$,

$$t \Rightarrow_{\mathcal{C}}^{*} (\bar{a}, \bar{b}, y) \quad \text{iff} \quad t \Rightarrow_{\mathcal{A}}^{*} a \text{ and } s \Rightarrow_{\mathcal{B}}^{*} b \text{ for some } s \in 1\ell\,\mathrm{N}_{\mathcal{R}}(t).$$

Thus $T(\mathcal{C}) = \emptyset$ if and only if $1\ell\,\mathrm{N}_{\mathcal{R}}(T_1) \subseteq T_2$. □

References

[Ave95] J. Avenhaus. *Reduktionssysteme.* Springer, 1995.

[DDC87] M. Dauchet and F. De Comite. A gap between linear and non-linear term-rewriting systems. In *Rewriting Techniques and Applications*, (Proc. Conf., Bordeaux, France, May 25–27, 1987), Lect. Notes Comput. Sci. **256**. Springer, 1987, 95–104.

[DG89] A. Deruyver and R. Gilleron. The reachability problem for ground TRS and some extensions. In *TAPSOFT'89*, (Proc. Conf. CAAP'89, Barcelona, Spain, March 1989), Lect. Notes Comput. Sci. **351**. Springer, 1989, 227–243.

[DJ90] N. Dershowitz and J.-P. Jouannaud. *Rewrite Systems*, volume B of *Handbook of Theoretical Computer Science*, chapter 6, pages 243–320. Elsevier, 1990.

[ES77] J. Engelfriet and E. M. Schmidt. IO and OI. I. *J. Comput. Syst. Sci.*, 15(3):328–353, 1977.

[ES78] J. Engelfriet and E. M. Schmidt. IO and OI. II. *J. Comput. Syst. Sci.*, 16(1):67–99, 1978.

[FJSV98] Z. Fülöp, E. Jurvanen, M. Steinby, and S. Vágvölgyi. On one-pass term rewriting. In *MFCS'98*, (Proc. Conf., Brno, Czech Republic, August 1998), Lect. Notes Comput. Sci. **1450**. Springer, 1998, 248–256.

[Gil91] R. Gilleron. Decision problems for term rewriting systems and recognizable tree languages. In *STACS'91*, (Proc. Conf., Hamburg, Germany, February 14–16, 1991), Lect. Notes Comput. Sci. **480**. Springer, 1991, 148–159.

[GS84] F. Gécseg and M. Steinby. *Tree automata.* Akadémiai Kiadó, Budapest, 1984.

[GS97] F. Gécseg and M. Steinby. *Tree Languages*, volume 3 of *Handbook of Formal Languages*, chapter 1, pages 1–68. Springer, 1997.

[GT95] R. Gilleron and S. Tison. Regular tree languages and rewrite systems. *Fundam. Inf.*, 24(1,2):157–175, 1995.

[GV98] P. Gyenizse and S. Vágvölgyi. Linear generalized semi-monadic rewrite systems effectively preserve recognizability. *Theor. Comput. Sci.*, 194(1-2):87–122, 1998.

[HH94] D. Hofbauer and M. Huber. Linearizing term rewriting systems using test sets. *J. Symb. Comput.*, 17(1):91–129, 1994.

[Hue80] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980.

[KT95] G. Kucherov and M. Tajine. Decidability of regularity and related properties of ground normal form languages. *Inf. Comput.*, 118(1):91–100, 1995.

[VG92] S. Vágvölgyi and R. Gilleron. For a rewrite system it is decidable whether the set of irreducible, ground terms is recognizable. *Bull. EATCS*, 48:197–209, 1992.