Homomorphisms and Concurrent Term Rewriting

Franck Seynhaeve, Sophie Tison, and Marc Tommasi

LIFL, Bât M3, Université Lille 1, F59655 Villeneuve d'Ascq cedex, France
seynhaev,tison,tommasi@lifl.fr

Abstract. In this paper we study applications of relations based on rewrite systems to regular tree languages. For instance, we want to deal with decidability problems of the form "$\mathcal{R}el(L_1) \subseteq L_2$" where L_1, L_2 are regular tree languages and $\mathcal{R}el$ can be either IO, OI, parallel, or one step rewriting for a given rewrite system. Our method somehow standardizes previous ones because it reveals conditions $\mathcal{R}el$ must fulfill to preserve recognizability for the language $\mathcal{R}el(L_1)$. Thanks to classes of recognizable languages wider than the regular one, we get some new results. We pursue this method to tackle the problem of computing the set of descendants of a regular tree language by a rewrite system.

1 Introduction

This paper tackles decidability questions $\mathcal{R}el(L_1) \subseteq L_2$ where L_1 and L_2 are regular tree languages and $\mathcal{R}el$ is a relation based on a term rewriting system. For instance, we want to decide whether the set of direct successors of terms in L_1 by a term rewriting system is a subset of L_2. In this case $\mathcal{R}el$ is "one-step rewriting". For example, this enables us to decide whether $\mathcal{R}^*(L_1) = L_1$ for any term rewriting system $\mathcal{R}$. We essentially investigate cases where $\mathcal{R}el$ is one-pass rewriting with several strategies: one-step rewriting, parallel rewritings, IO or OI pass rewriting ([Eng78]) or leaf or root started rewriting ([FJSV98]). All these relations have a common principle: redexes are sufficiently spread out in terms to avoid creating overlaps between two rewritings. We say in this case that rewritings can be done concurrently. Basically, we exploit this property in our constructions.

Decidability problems $\mathcal{R}el(L_1) \subseteq L_2$ are difficult to solve because $\mathcal{R}el(L_1)$ is not regular. So we suffer from lack of usual useful tools to manipulate these sets, and to decide for instance, inclusion or emptiness. Nonetheless, related works in the literature often partially solve this problem with tree automata manipulations, *e.g.* by adding or transforming rules. Authors reduce these decidability questions about complex sets to decidability questions about regular tree languages. We propose in this paper an approach that reveals why this can be done and somehow unifies them in a standard understanding. Thanks to classes of recognizable languages larger than the regular one, we obtain new decidability results.

The main principle is to reduce the inclusion $\mathcal{R}el(L_1) \subseteq L_2$ to some inclusion between regular sets in such a way the latter holds if and only if the former also holds. To do that, we use transformations via inverse tree homomorphisms and intersections by regular tree languages[1]. Inverse homomorphisms fix redexes and

[1] In the sequel, we will write just "language" and "homomorphisms" instead of "tree language" and "tree homomorphism", respectively.

intersections control that the relation $\mathcal{R}el$ is well applied. For instance, let us consider that $\mathcal{R}el$ is one-step rewriting with a given rule $l \to r$. We denote by $\mathcal{R}el(L_1)$ the set $\{t \mid u \to t,\ u \in L_1\}$. We consider a new symbol $\bullet$ and two tree homomorphisms Ψ_r and Ψ_l defined by the mappings $\phi_l(\bullet) = l$ and $\phi_r(\bullet) = r$. In this case one can show that $\mathcal{R}el(L_1)$ is exactly the set $\Psi_r(\Psi_l^{-1}(L_1) \cap \mathcal{R}_c)$, where $\mathcal{R}_c$ is the regular language of terms containing exactly one symbol $\bullet$. So we have two immediate consequences, from the closure properties of regular languages. First, $\mathcal{R}el(L_1) \subseteq L_2$ is decidable because it is equivalent to decide whether $\Psi_l^{-1}(L_1) \cap \mathcal{R}_c) \subseteq \Psi_r^{-1}(L_2)$ and this last expression only involves regular languages. Second, if Ψ_r is a linear homomorphism, then $\mathcal{R}el(L_1)$ is regular and therefore we can decide whether $\mathcal{R}el(L_1) = L_2$. It is important to note that this method leads to effective proofs.

The question is now: how this method can be extended and what are its limits? We give some ideas for the answer. Rewritings can be "imitated" by homomorphisms only in some cases. Homomorphisms are not helpful when redexes are too close, namely when some instances of left-hand sides of rules overlap in two rewriting steps. When more than one rewriting step are applied, we need to follow some strategies in order to keep the opportunity of using homomorphisms. For instance, IO and OI ([Eng75,ES77,ES78]) strategies and one-pass leaf- and root-started strategies ([FJSV98]) have this non-overlapping property but some new restrictions on linearities on left or right-hand sides of rules are now required.

Another improvement is to try to compute the set $\mathcal{R}^*(L)$ of descendants (or the set of normal forms) of a regular language L by a rewrite system $\mathcal{R}$. This topic has been studied in many papers ([Sal88,DT90,CDGV94]) and all constructions rely on similar arguments. We are able to compute $\mathcal{R}^*(L)$ with our method under some restrictions. In fact, we construct a sequence of languages $(L_k)_{k \in N}$ with $L_0 = L$ such that:

$$\forall k \in N,\ \mathcal{R}(L_k) \subseteq L_{k+1} \subseteq \mathcal{R}^*(L_k).$$

These languages are regular when $\mathcal{R}$ is right-linear and under some other restrictions on $\mathcal{R}$, there exists an integer i such that $\forall k \geq i$, $L_k = L_i$ hence $\mathcal{R}^*(L) = L_i$.

The paper is organized as follows. In section 3, we introduce our method. We apply it in Section 4 on one-step rewriting, parallel rewriting, computation of normal and sentential forms according to one-pass IO, one-pass OI, one-pass root-started or leaves started. Finally we study the language $\mathcal{R}^*(L)$ in Section 5.

The table in Section 5.1 summarizes our results.

2 Preliminaries

2.1 Signature, Terms, and Rewriting

Let $n \in N$. We denote by $[n]$ the set $\{1, 2, \ldots, n\}$ and N^* denotes the set of finite-length strings over N.

Let us consider a signature Σ and a countable set of variables $\mathcal{X}$. A *term* over $\Sigma \cup \mathcal{X}$ is a partial function $t : N^* \to \Sigma \cup \mathcal{X}$ whose domain $\mathcal{P}os(t)$ satisfies:
- $\mathcal{P}os(t)$ is nonempty and prefix-closed;
- If $t(p) \in \Sigma_n$, then $\{i | pi \in \mathcal{P}os(t)\} = \{0, 1, \ldots, n-1\}$;
- If $t(p) \in \mathcal{X}$, then $\{i | pi \in \mathcal{P}os(t)\} = \emptyset$.

Each element of $\mathcal{P}os(t)$ is called a *position.* Two positions p_1 and p_2 are *comparable* if there exists p such that $p_1 = pp_2$ or $p_2 = pp_1$, otherwise they are *incomparable.* A *branch* of t is a prefix-closed subset of $\mathcal{P}os(t)$ in which all positions are pairwise comparable and whose greater position p holds a constant or a variable, *i.e.* $t(p) \in \mathcal{X} \cup \Sigma_0$. We denote by $|p|$ the length of a position p. The *height* of a term t, denoted by $Height(t)$, is the maximal length over all positions in t. We denote by $\mathcal{V}ar(t)$ the set of variables which occur in t, and $t|_p$ is the subterm of t rooted at position p, and $t[u]_p$ is the term obtained by replacing in t the subterm at position p by u. If $|p| = n$, then $t|_p$ is called a subterm of t at depth n. Given a position p, subterms rooted at position pw, $w \in N$, are called *brother terms.*

The set of all terms (or *trees*) is denoted by $T(\Sigma, \mathcal{X})$. If $\mathcal{X} = \emptyset$ then $T(\Sigma, \mathcal{X})$ is denoted by $T(\Sigma)$. Terms of $T(\Sigma)$ are called *ground terms.* A term t in $T(\Sigma, \mathcal{X})$ is *linear* if each variable occurs at most once in t. When each variable either occurs at positions of length 1 or only once in t, we say that t is *semi-linear.* We fix some typographical conventions for the rest of the paper. In what follows, x, y and z, possibly with subscripts, are variables, and the letters t, u and v will denote terms.

We denote by $\mathcal{X}_n$ a set of n distinct variables, $\mathcal{X}_n = \{x_1, \ldots, x_n\}$. A linear term C of $T(\Sigma, \mathcal{X}_n)$ is called a *context* and the expression $C[t_1, \ldots, t_n]$ denotes the term obtained from C by replacing for each i, x_i by t_i. We denote by $\mathcal{C}^n(\Sigma)$ the set of contexts on Σ and $\mathcal{X}_n$, and $\mathcal{C}(\Sigma)$ the set of context with one variable.

A substitution $\sigma : \mathcal{X} \to T(\Sigma, \mathcal{X})$ is extended to a mapping $\sigma : T(\Sigma, \mathcal{X}) \to T(\Sigma, \mathcal{X})$ so that $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$. A term $t \in T(\Sigma)$ *encompasses* a term $u \in T(\Sigma, \mathcal{X})$ if there exists a substitution σ such that σu is a subterm of t.

Let Γ be a signature and φ a mapping which, with $f \in \Sigma_n$, associates a term $t_f \in T(\Gamma, \mathcal{X}_n)$. The *tree homomorphism* Φ from $T(\Sigma)$ into $T(\Gamma)$ determined by φ is defined as follows:

- $\Phi(a) = t_a \in T(\Gamma)$ for each $a \in \Sigma_0$;
- $\Phi(f(t_1, \ldots, t_n)) = t_f[\Phi(t_1), \ldots, \Phi(t_n)]$.

A tree homomorphism is linear when for each symbol $f \in \Sigma_n$, $\varphi(f) = t_f$ is a linear term of $T_\Gamma(\mathcal{X}_n)$ and semi-linear if t_f is semi-linear.

A *term rewriting system* (TRS) $\mathcal{R}$ over Σ is a finite set of *rewrite rules* $l \to r$ such that $l, r \in T(\Sigma, \mathcal{X})$, $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ and $l \notin \mathcal{X}$. The rewrite relation $\to_\mathcal{R}$ on $T(\Sigma)$ induced by $\mathcal{R}$ is defined so that $t \to_\mathcal{R} u$ if there exist $p \in \mathcal{P}os(t)$, a substitution σ and $l \to r \in \mathcal{R}$ such that $t|_p = \sigma l$ and $u = t[\sigma r]_p$. The reflexive, transitive closure of $\to_\mathcal{R}$ is denoted by $\to_\mathcal{R}^*$. Hence $s \to_\mathcal{R}^* t$ iff there exists a *derivation* $t_0 \to_\mathcal{R} t_1 \to_\mathcal{R} \ldots \to_\mathcal{R} t_n$ in $\mathcal{R}$ such that $n \geq 0$, $t_0 = s$ and $t_n = t$.

Given a rule $l \to r \in \mathcal{R}$, the term l is the left-hand side and r is the right-hand side of the rule. When every left-hand side of rule is linear, we say that $\mathcal{R}$ is left-linear. Similarly, we define right-linear, left-semi-linear and right-semi-linear TRS.

A term $s \in T(\Sigma, \mathcal{X})$ is *irreducible* with respect to $\mathcal{R}$ if $s \to_\mathcal{R} u$ for no u. And s is a *normal form* of $t \in T(\Sigma, \mathcal{X})$ if $t \to_\mathcal{R}^* s$ and s is irreducible with respect to $\mathcal{R}$. Finally s is *normalizable* with respect to $\mathcal{R}$ if there exists a normal form of s.

2.2 Tree Automata

Most of the constructions in this paper rely on three classes of tree automata: finite tree automata, automata with tests between brothers and reduction automata. General tree automata generalized these three classes.

A general tree automaton $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta)$ is given by a signature Σ, a finite set of unary symbols called *states* $\mathcal{Q}$ such that $\Sigma \cap \mathcal{Q} = \emptyset$, a finite set of *final states* $\mathcal{Q}_f \subseteq \mathcal{Q}$, a finite set of *rewrite rules* Δ of the form $f(q_1(x_1), \dots, q_n(x_n)) \xrightarrow{c} q(f(x_1, \dots, x_n))$ where $f \in \Sigma_n$, $q, q_1, \dots, q_n \in \mathcal{Q}$ and c is a *constraint*, *i.e.* a conjunction of positive constraint $p = p'$ and negative constraint $p \neq p'$ where p and p' are positions of length greater than or equal to 1.

Rules in Δ are constrained rules, *i.e.* $t \rightarrow_{\mathcal{A}} s$ at position p with $t, s \in T(\Sigma \cup \mathcal{Q})$ if there exists a rule $f(q_1(x_1), \dots, q_n(x_n)) \xrightarrow{c} q(f(x_1, \dots, x_n)) \in \Delta$ such that: $t \rightarrow s$ by the rule $f(q_1(x_1), \dots, q_n(x_n)) \rightarrow q(f(x_1, \dots, x_n))$ at position p, and $f(t|_{p11}, \dots, t|_{pn1})$ satisfies c.

A term $t \in T(\Sigma)$ satisfies a positive constraint $p = p'$ (respectively negative constraint $p \neq p'$) if $t|_p = t|_{p'}$ (respectively $t|_p \neq t|_{p'}$).

A term $t \in T(\Sigma)$ is *accepted* by $\mathcal{A}$ if $t \rightarrow^*_{\mathcal{A}} q(t)$ where q is a final state. The language *recognized* by $\mathcal{A}$ is the set of terms accepted by $\mathcal{A}$ and is denoted by $L(\mathcal{A})$.

$\mathcal{A}$ is *deterministic* when for each couple $f(q_1(x_1), \dots, q_n(x_n)) \xrightarrow{c} q(f(x_1, \dots, x_n)$ and $f(q_1(x_1), \dots, q_n(x_n)) \xrightarrow{c'} q'(f(x_1, \dots, x_n)$ of rules of $\mathcal{A}$, the constraint $c \wedge c'$ is unsatisfiable if $q \neq q'$.

A tree automaton is a general tree automaton where rules are without constraint. The class of tree languages recognized by tree automata is the class of regular languages.

A tree automaton with tests between brothers is a general tree automaton where the length of positions occurring in each constraint is 1, *i.e.* equalities and inequalities are imposed between brother terms. $REC_{\neq}$ denotes the class of languages recognized by tree automata with tests between brothers.

A reduction automaton $\mathcal{A}$ is a general tree automaton such that there is an ordering on the states of $\mathcal{A}$ such that, for each rule $f(q_1(x_1), \dots, q_n(x_n)) \xrightarrow{c} q(x_1, \dots, x_n))$, q is smaller than each q_i. Moreover if a positive constraint occurs in c, q is strictly smaller than each q_i. RA denotes the class of languages recognized by the class of reduction automata.

Classes of tree languages recognized by tree automata, tree automata with tests between brothers and deterministic reduction automata are closed under union, intersection, complementation. Moreover, the inclusion problem and the emptiness problem are also decidable.

The class of regular languages is closed under inverse homomorphisms and linear homomorphisms, *i.e.* $\Phi^{-1}(L)$ and $\Psi(L)$ are regular for any homomorphism Φ, linear homomorphism Ψ and regular language L. The image of a regular language L by a semi-linear homomorphism is a language of $REC_{\neq}$.

Finally the set of terms encompassing a linear term is regular and the set of terms encompassing any term is recognizable by deterministic reduction automata. The recognizability problem is decidable for $REC_{\neq}$, *i.e.* it is decidable whether L is regular for any $L \in REC_{\neq}$ [BST99].

3 Decision Problems and General Method of Resolution

Let us consider the decision questions "$\mathcal{R}el(L_1) \subseteq L_2$?" and "$\mathcal{R}el(L_1) = L_2$?" where L_1 and L_2 are regular languages, and $\mathcal{R}el(L_1)$ is the image of terms of L_1 by a relation $\mathcal{R}el$.

Our method for deciding such questions is based on the following remark: under some assumptions, we can associate with $\mathcal{R}el$ two homomorphisms Ψ_l, Ψ_r and a regular "checking" language R_c, i.e. a tree bimorphism [AD82], such that:

$$\mathcal{R}el(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c) \tag{1}$$

The construction is very simple: let $\mathcal{R}$ be a TRS of n rules $l_i \to r_i$.

We associate with each rule of the system $\mathcal{R}$ a new symbol $\bullet_i$ whose arity equals to the number of distinct variables of $l_i \to r_i$. Let Λ be the signature of these new symbols and Γ the signature $\Sigma \cup \Lambda$. Ψ_l and Ψ_r will be the tree homomorphisms from $T(\Gamma)$ into $T(\Sigma)$ defined by:

- $\forall f \in \Sigma_p$, $\Psi_r(f) = \Psi_l(f) = f(x_1, \ldots, x_p)$, and
- $\forall i \in [n]$, $\Psi_l(\bullet_i) = l_i$ and $\Psi_r(\bullet_i) = r_i$.

The set $\Psi_l^{-1}(t)$, can be viewed as the set of terms of $T(\Gamma)$ obtained by putting symbols of Λ in place of some occurrences of left-hand sides of rules in t. Homomorphism Ψ_r will apply the rewrite rules at this redexes.

The checking language R_c will be defined w.r.t. $\mathcal{R}el$. For example, when $\mathcal{R}el$ is just the one-step rewriting, R_c just checks that there is only one $\bullet_i$.

Of course this method has its limitations. Roughly speaking, we can simulate by this way rewriting which can be done concurrently; it implies non-overlapping and some conditions on linearity.

We show now how this construction gives easy decision algorithms for our questions, under some assumptions.

Let us first notice that when Ψ_r is linear (i.e. $\mathcal{R}$ is right-linear) and R_c is recognizable, $\Psi_r(\Psi_l^{-1}(L_1) \cap R_c)$ is recognizable (and you can effectively construct an automaton for it). So we can decide whether it is included in (resp. equal to) a given regular language. In fact we can reduce the right-linearity condition to the right-semi-linearity condition, since the image of a regular language by a semi-linear homomorphism is in $REC_{\neq}$ and the inclusion problem is decidable for $REC_{\neq}$.

So in this case, we get an easy decision algorithm for deciding "$\mathcal{R}el(L_1) \subseteq L_2$?" and "$\mathcal{R}el(L_1) = L_2$?" where L_1 and L_2 are regular languages.

When Ψ_r is not linear, $\Psi_r(\Psi_l^{-1}(L_1) \cap R_c)$ is no more (in general) recognizable; however, we can decide "$\mathcal{R}el(L_1) \subseteq L_2$?".

Indeed, $\Psi_r(\Psi_l^{-1}(L_1) \cap R_c)$ is included in L_1 if and only if $\Psi_l^{-1}(L_1) \cap R_c$ is included in $\Psi_r^{-1}(L_1)$. When R_c is regular, both $\Psi_l^{-1}(L_1) \cap R_c$ and $\Psi_r^{-1}(L_1)$ are (effectively) regular. Hence we just have to decide as whether a regular language is included in another one.

In fact, we don't need the recognizability of R_c; using the properties of the class of languages recognized by reduction automata (decidability of emptiness, closure under intersection), we just have to suppose that R_c is recognizable by a reduction automaton. This allows us for example to deal with computation of normal forms according to some strategies like in [FJSV98]. This works fine because, for any finite set of terms E_s, the language $R_s^{FN} = \{t \in T(\Gamma) \mid t$ encompasses no term of $E_s\}$ is recognizable by a reduction automaton [DCC95].

To summarize, our method is based on the two following propositions:

Proposition 1. *If $\mathcal{R}el(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c)$ with Ψ_r right-semi-linear and R_c regular, then "$\mathcal{R}el(L_1) = L_2$?" with L_1 and L_2 regular, is decidable.*

Proposition 2. *If $\mathcal{R}el(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c)$ with R_c recognizable by a reduction automaton, then "$\mathcal{R}el(L_1) \subseteq L_2$?" with L_1 and L_2 regular, is decidable.*

4 Applications

We use the method developed in the previous section in order to study the decision questions "$\mathcal{R}el(L_1) \subseteq L_2$?" and "$\mathcal{R}el(L_1) = L_2$?" where L_1 and L_2 are regular for different relations $\mathcal{R}el$ depending on a TRS $\mathcal{R}$ composed of n rules. We give for each studied relation, the checking language R_c and the conditions for which $\mathcal{R}el(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c)$.

4.1 One-Step Rewriting and Parallel Rewriting

One-step rewriting is the application of one rule of $\mathcal{R}$. The image of L by one-step rewriting is $\mathcal{R}(L) = \{t \in T(\Sigma) \mid \exists u \in L,\ u \to_{\mathcal{R}} t\}$. *Parallel rewriting* is the application of a set of rules on incomparable positions. Let u and t be two terms of $T(\Sigma)$. If parallel rewriting is denoted by $\dashrightarrow\!\!\!\!\!\!\|\ $, $t \dashrightarrow\!\!\!\!\!\!\|\ u$ if and only if there exists $k \in N$, $C \in \mathcal{C}^k(\Sigma)$, $i_1, \ldots, i_k \in [n]$ and $\sigma_1, \ldots, \sigma_k$ substitutions on $T(\Sigma, \mathcal{X})$ such that $t = C[l_{i_1}\sigma_1, \ldots, l_{i_k}\sigma_k]$ and $u = C[r_{i_1}\sigma_1, \ldots, r_{i_k}\sigma_k]$. The image of L by application of parallel rewriting is $\mathcal{R}_{||}(\mathcal{L}) = \{t \in T(\Sigma) \mid \exists u \in L,\ u \dashrightarrow\!\!\!\!\!\!\|\ t\}$.

Example 1. Let us consider the signature $\Sigma = \{f/2, g/1, a/0, b/0\}$ and the TRS with two rules $f(x,x) \to g(x)$ and $b \to a$. If $L = \{f(g(b), g(b))\}$ then

$$\mathcal{R}(L) = \{g(g(b)),\ f(g(a), g(b)),\ f(g(b), g(a))\}$$
$$\mathcal{R}_{||}(L) = \{f(g(b), g(b)),\ f(g(b), g(a)),\ f(g(a), g(b)),\ f(g(a), g(a)),\ g(g(b))\}$$

Let R_c^1 be the set of terms of $T(\Gamma)$ with only one symbol of Λ and $R_c^{||}$ the set of terms of $T(\Gamma)$ with at most one symbol of Λ on each branch. These languages are regular and we can prove that:

$$\mathcal{R}(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap \mathcal{R}_c^1) \qquad \mathcal{R}_{||}(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap \mathcal{R}_c^{||})$$

Hence according to Propositions 2 and 1 we obtain:

Proposition 3. *"$\mathcal{R}(L_1) \subseteq L_2$?" and $\mathcal{R}_{||}(L_1) \subseteq L_2$?" are decidable for any TRS $\mathcal{R}$, and "$\mathcal{R}(L_1) = L_2$?" and "$\mathcal{R}_{||}(L_1) = L_2$?" are decidable for any right-semi-linear TRS $\mathcal{R}$.*

Moreover for any TRS $\mathcal{R}$ and for any regular language L, $\mathcal{R}(L) \subseteq L \Leftrightarrow \mathcal{R}^*(L) = L$. This statement can be used as a halting criterion for approximations of computations of the set $\mathcal{R}^*(L)$. For instance, see related works by J. Waldmann or T. Genet [Wal98,Gen98].

Corollary 1. *For any TRS $\mathcal{R}$ and regular language L, "$\mathcal{R}^*(L) = L$?" is decidable.*

4.2 One-Pass IO and OI

If $\mathcal{R}$ is linear, a term t rewrites to a term u in *one-pass* if all rewritings in the derivation of t in u can be applied concurrently.

Example 2. Let $\Sigma = \{h/3, f/2, g/1, a/0\}$ and $\mathcal{R}$ be the TRS composed of the rules $h(x, g(y), z) \to f(y, x)$ and $g(x) \to a$. The following derivation can be done in one-pass:

$$h(f(g(a),a)\ ,g(g(a))\ ,a) \to f(g(a)\ ,f(g(a),a)\ ) \to f(g(a)\ ,f(a,a)\ ) \to f(a,f(a,a)\ )$$

In general, $\mathcal{R}$ is not linear, and hence we must indicate if subtrees are rewritten before or after generating or testing equalities defined by the rules. There are two usual strategies [Eng75]:

OI pass: Rewriting from the root (outermost) to the leaves (innermost).
IO pass: Rewriting from the leaves to the root.

If a term t rewrites to a term u in one-pass OI (respectively IO), then we denote $t \to^{oi} u$ (respectively $t \to^{io} u$).

Example 3. Let $\Sigma = \{f/2, g/2, a/0, b/0\}$ and consider the rewrites rules $f(x,x) \to g(x,x)$ and $a \to b$.

Hence $f(a,a) \to^{oi} g(a,b)$ but $f(a,a) \not\to^{io} g(a,b)$, and $f(a,b) \to^{io} g(b,b)$ but $f(a,b) \not\to^{oi} g(b,b)$.

Let us denote $\mathsf{SF}_{\mathsf{oi}}(L) = \{t \in T(\Sigma) \mid \exists s \in L, s \to^{oi} t\}$ and $\mathsf{SF}_{\mathsf{io}}(L) = \{t \in T(\Sigma) \mid \exists s \in L, s \to^{io} t\}$. When $\mathcal{R}el$ is one of these relations, there is no checking language and we can prove that $\Psi_r(\Psi_l^{-1}(L_1)) \subseteq \mathsf{SF}_{\mathsf{io}}(L_1)$ and $\Psi_r(\Psi_l^{-1}(L_1)) \subseteq \mathsf{SF}_{\mathsf{oi}}(L_1)$. Examples 4 and 5 show that these inclusions may be strict.

Example 4 (IO requires rules to be left-linear).
Let the signature $\Sigma = \{f/2, g/1, a/0, b/0\}$ and the rules $f(x,x) \to g(x)$ and $a \to b$.

We have $f(a,b) \to^{io} g(b)$ since $f(a,b) \to f(b,b) \to g(b)$. But there is no term u of $T(\Gamma)$ such that $\Psi_l(u) = f(a,b)$ and $\Psi_r(u) = g(b)$.

Example 5 (OI requires rules to be right-linear).
Let the signature $\Sigma = \{f/2, g/1, a/0, b/0\}$ and the rules $g(x) \to f(x,x)$, $g(x) \to a$ and $g(x) \to b$.

We have $g(g(a)) \to^{oi} f(a,b)$ since $g(g(a)) \to f(g(a), g(a)) \to f(a, g(a)) \to f(a,b)$. But there is no term u of $T(\Gamma)$ such that $\Psi_l(u) = g(g(a))$ and $\Psi_r(u) = f(a,b)$.

We can prove that for every left-linear (respectively right-linear) TRS $\mathcal{R}$, $\mathsf{SF}_{\mathsf{io}}(L_1) = \Psi_r(\Psi_l^{-1}(L_1))$ (respectively $\mathsf{SF}_{\mathsf{oi}}(L_1) = \Psi_r(\Psi_l^{-1}(L_1))$). Therefore,

Proposition 4. *For any left-linear TRS "$\mathsf{SF}_{\mathsf{io}}(L_1) \subseteq L_2$?" is decidable, and for any left-linear and right-semi-linear TRS "$\mathsf{SF}_{\mathsf{io}}(L_1) = L_2$?" is decidable.*

For any right-linear TRS, "$\mathsf{SF}_{\mathsf{oi}}(L_1) \subseteq L_2$?" and "$\mathsf{SF}_{\mathsf{oi}}(L_1) = L_2$?" are decidable.

4.3 One-Pass Root-Started Rewriting and One-Pass Leaf-Started Rewriting

Z. Fülöp *et al.* introduce variants of the one-pass IO or OI rewritings in [FJSV98]: *one-pass root-started and leaf-started rewritings.* The first (respectively second) is a OI pass (respectively OI) pass in which each rewriting concerns positions immediately adjacent to parts of the term rewritten in previous rewritings.

Because these strategies are very similar to IO and OI ones, we obtain here similar results.

One-pass root-started rewriting may be described as follows. Let t be the term of $T(\Sigma)$ to be rewritten. The portion of t first rewritten should include the root. Rewriting then proceeds towards the leaves so that each rewrite step applies to a root segment of a maximal unprocessed subtree but never involves any part of the tree produced by a previous step. For the formal definition a new TRS in which a new special symbol forces this kind of rewriting is associated with $\mathcal{R}$. The reader is reported to [FJSV98] for a formal definition of this rewriting strategy.

Z. Fülöp *et al.* obtain decidability results when rewrite systems are *left*-linear. Surprisingly, our method leads to decidability results in the case of *right*-linear rewrite systems. Moreover, our techniques are powerful enough to strengthen the result from inclusion problems to equality problems.

In the following, $\rightarrow^{rs}$ denotes the one-pass root-started rewriting relation.

Let us denote $\mathsf{SF}_{\downarrow}(L) = \{t \in T(\Sigma) \mid \exists s \in \mathcal{L}, s \rightarrow^{rs} t\}$ and $\mathsf{NF}_{\downarrow}(L)$ the set of all one-pass root-started normal forms of L. Z. Fülöp *et al.* [FJSV98] have shown that "$\mathsf{SF}_{\downarrow}(L_1) \subseteq L_2$?" and "$\mathsf{NF}_{\downarrow}(L_1) \subseteq L_2$?" are decidable for any left-linear TRS $\mathcal{R}$.

Let us denote, for any term of $T(\Gamma)$, by $l_\pi(t)$ the string defined by the symbols on the branch π of t.

Sentential forms In order to verify that a one-pass root-started rewriting applies to a term t of $T(\Gamma)$, each word $l_\pi(t)$ must belong to $\Lambda^*\Sigma^*$. Let $R_c^{\downarrow s}$ be the set of terms t of $T(\Gamma)$ such that for any branch π of t, $l_\pi(t) \in \Lambda^*\Sigma^*$. We can prove that $R_c^{\downarrow s}$ is regular and that for any right-linear TRS $\mathcal{R}$:

$$\mathsf{SF}_{\downarrow}(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c^{\downarrow s}).$$

Right linearity is necessary for the same reason than for one-pass OI.

Proposition 5. *For any right-linear TRS "$\mathsf{SF}_{\downarrow}(L_1) \subseteq L_2$?" and "$\mathsf{SF}_{\downarrow}(L_1) = L_2$?" are decidable.*

Normal forms The only difficulty here is to define the checking language. According to the strategy, when $u \in \Psi_l^{-1}(t)$ satisfies one of the two following properties a rewrite rule can be applied: there is an instance of a rule at the root of u or there is an instance of a rule just below a dotted symbol in Λ.

Let $E_{\downarrow}^{FN}$, $R_{\downarrow 1}^{FN}$ and $R_{\downarrow 2}^{FN}$ be the following sets of terms:

$$\begin{aligned}
E_{\downarrow}^{FN} &= \{\bullet_i[y_1,\ldots,y_{l-1},l_k[x_1,\ldots,x_{p_k}],y_{l+1},\ldots,y_{p_i}] \mid i,k \in [n], l \in [p_i] \text{ and} \\
&\qquad x_1,\ldots,x_{p_k},y_1,\ldots,y_{l-1},y_{k+1},\ldots,y_{p_i} \in \mathcal{X}\} \\
R_{\downarrow 2}^{FN} &= \{t \in T(\Gamma) \mid t \text{ encompasses no term of } E_{\downarrow}^{FN}\} \\
R_{\downarrow 2}^{FN} &= \{t \in T(\Gamma) \mid \nexists i \in [n], \nexists \sigma \text{ substitution}, t = \sigma l_i\}
\end{aligned}$$

We have $\mathsf{NF}_{\downarrow}(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c^{\downarrow s} \cap R_{\downarrow 1}^{FN} \cap R_{\downarrow 2}^{FN})$. Moreover, $R_{\downarrow 2}^{FN}$ is recognizable by a deterministic reduction automaton. Hence,

Proposition 6. *For any right-linear TRS "$\mathsf{NF}_{\downarrow}(L_1) \subseteq L_2$?" is decidable.*
For any linear TRS, "$\mathsf{NF}_{\downarrow}(L_1) = L_2$?" is decidable.

One-pass leaf-started rewriting One-pass leaf-started rewriting starts from leaves (either from constants or just above constants) and proceeds upwards. Each rewrite step applies just at the border of unprocessed part of the tree.

Z. Fülöp *et al.* obtain decidability results when rewrite systems are *left*-linear. On the one hand, we need stronger restrictions, but on the other hand we are able to strengthen the result from inclusion problems to equality problems.

In the following, $\rightarrow^{ls}$ denotes the one-pass leaf-started rewriting relation, $\mathsf{SF}_{\uparrow}(L) = \{t \in T(\Sigma) \mid \exists s \in L, s \rightarrow^{ls} t\}$ and $\mathsf{NF}_{\uparrow}(L)$ is the set of all one-pass leaf-started normal forms of L. Z. Fülöp *et al.* [FJSV98] have shown that "$\mathsf{SF}_{\uparrow}(L_1) \subseteq L_2$?" and "$\mathsf{NF}_{\uparrow}(L_1) \subseteq L_2$?" are decidable for any left-linear TRS $\mathcal{R}$.

Sentential forms. In order to verify that a one-pass leaf-started rewriting applies on a term t of $T(\Gamma)$, each string $l_\pi(t)$ must belong to $\Sigma^*\Lambda^*\Sigma_0 \cup \Sigma^*\Lambda^*$. Let $R_c^{\uparrow s}$ be the set of terms t of $T(\Gamma)$ such that for any branch π of t, $l_\pi(t) \in \Sigma^*\Lambda^*\Sigma_0 \cup \Sigma^*\Lambda^*$. We can prove that $R_c^{\uparrow s}$ is regular and that for any left-linear TRS $\mathcal{R}$:

$$\mathsf{SF}_{\uparrow}(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c^{\uparrow s}).$$

Left linearity is necessary for the same reason than for one-pass IO.

Proposition 7. *For any left-linear and right-semi-linear TRS, "$\mathsf{SF}_{\uparrow}(L_1) = L_2$?" is decidable.*

Normal forms $\forall i \in [n]$, let p_i be the number of distinct variables of l_i. Let $E_{\uparrow}^{NF}$ and $\mathcal{R}_{\uparrow}^{NF}$ be the following sets of terms:

$$\begin{aligned} E_{\uparrow}^{NF} = \{l_i[u_1, \ldots, u_{p_i}] \mid\ & i \in [n], \forall j \in [p_i], u_j \in \Sigma_0 \text{ or } \exists k \in [n], \exists y_1, \ldots, y_{p_k} \in \mathcal{X}, \\ & u_j = \bullet_k(y_1, \ldots, y_{p_k}) \text{ and variables of } \textstyle\bigcup_{j \in [p_i]} \mathcal{V}ar(u_j) \\ & \text{are pairwise distinct}\} \\ \mathcal{R}_{\uparrow}^{NF} = \{t \in T(\Gamma) \mid\ & t \text{ encompasses no term of } E_{\uparrow}^{NF}\} \end{aligned}$$

We can prove that for any term $t \in R_c^{\uparrow s} \cap \mathcal{R}_{\uparrow}^{NF}$, one-pass leaf-started rewriting cannot be continued from t since no left-hand side of rule occur just above symbols of Λ and constants. We deduce that $\mathsf{NF}_{\uparrow}(L_1) = \Psi_r(\Psi_l^{-1}(L_1) \cap R_c^{\uparrow s} \cap \mathcal{R}_{\uparrow}^{NF})$. Moreover when $\mathcal{R}$ is left-linear, terms of $E_{\uparrow}^{NF}$ are linear. Hence,

Proposition 8. *For any left-linear and right-semi-linear TRS, "$\mathsf{NF}_{\uparrow}(L_1) = L_2$?" is decidable.*

5 Multiple Rewritings

In this section, we study the language $\mathcal{R}^*(L)$ where $\mathcal{R}$ is a TRS of n rules and L is a regular language: $\mathcal{R}^*(L) = \{t \mid \exists s \in L, s \xrightarrow{*}_{\mathcal{R}} t\}$.

Generally, this language is not regular even when the language $\mathcal{R}(L)$ is. However some authors proved that $\mathcal{R}^*(L)$ is regular for some classes of TRS: M. Dauchet et S. Tison [DT90] for ground TRS, *i.e.* systems whose left and right-hand sides of rules are ground terms; K. Salomaa [Sal88] for right-linear monadic TRS, a rule $l \to r$ being monadic if $\mathit{Height}(l) \geq 1$ and $\mathit{Height}(r) \leq 1$; J.-L. Coquidé *et al.* [CDGV94] for semi-monadic TRS, a rule $l \to r$ being semi-monadic if $\mathit{height}(l) \geq 1$ and r is a variable or a term whose variables occurs at depth one.

Jacquemard proves in [Jac96b,Jac96a] that the set of ground terms normalizable with respect to a growing TRS is regular. A rule $l \to r$ is growing if it is linear and if every variable x that occurs in both sides of the rule, x occurs at depth one in l. All these results are based on a similar construction that iteratively transforms a tree automaton.

Our result is more restrictive than results of K. Salomaa [Sal88] and J.-L. Coquidé *et al.* [CDGV94] since ground terms are not allowed at depth one of right-hand sides of rules. But our proof is directly issued from the previous section.

We are able to compute $\mathcal{R}(L)$ with our method. One can iterate the construction and compute $\mathcal{R}^*(L)$ under two restrictions: the construction preserves recognizability, and it terminates. Recognizability is preserved under right-linearity assumption and termination is obtained when the size of the associated automaton does not increase. Unfortunately, when the rewrite system is not left-linear, a determinization procedure is necessary at each step to compute inverse homomorphisms and hence, an exponential blow up step appears. Therefore, we restrict ourself to the case of linear rewrite systems.

Proposition 9. *Let $\mathcal{R}$ be a rewrite system. For each language L' on Σ, $L' \cup \mathcal{R}(L') \subseteq \Psi_r(\Psi_l^{-1}(L')) \subseteq \mathcal{R}^*(L')$.*

Let us consider now the sequence of languages $(L_k)_{k \in N}$ defined by $L_0 = L$ and for every $k > 0$, $L_k = \Psi_r(\Psi_l^{-1}(L_{k-1}))$. We deduce from Proposition 9 that:

$$\forall k \in N,\ \ L_k \cup \mathcal{R}(L_k) \subseteq L_{k+1} \subseteq \mathcal{R}^*(L_k).$$

Hence for each integer k, $L_k \subseteq L_{k+1} \subseteq \mathcal{R}^*(L_k)$, therefore $L_k \subseteq L_{k+1} \subseteq \mathcal{R}^*(L_0)$. Hence the sequence $(L_k)_{k \in N}$ is ordered by inclusion and dominated by $\mathcal{R}^*(L_0)$, i.e:

$$L_0 \subseteq L_1 \ldots \subseteq L_k \subseteq \ldots \subseteq \mathcal{R}^*(L_0).$$

We want this sequence to be consist of a finite number of regular languages in order to obtain for some natural number k, $L_k = \mathcal{R}^*(L_0)$. Since the construction is based on the computation of $\Psi_r(\Psi_l^{-1}(L_i))$, we first need Ψ_r to be linear. Moreover, because we want to build a finite sequence, we require the number of states in the automaton for L_{i+1} not to be greater than the number of states in the automaton for L_i. Hence, some supplementary restrictions are needed for Ψ_l and Ψ_r. The classical construction for $\Psi_l^{-1}(L_i)$ requires a deterministic automaton for L_i. But in the case of linear morphisms Ψ, this requirement can be dropped. Finally, if Ψ_r generates terms of height greater than one, additional states are necessary. Hence,

Definition 1. *A* simple *rewrite system is a linear TRS whose each right-hand side of rule is either a constant, or a variable, or a term of height one without ground subterms.*

Therefore, mappings Ψ_l and Ψ_r associated with *simple* rewrite systems are such that Ψ_l is linear and for each symbol f of Γ: either $\psi_r(f)$ is a constant or a variable, or $\psi_r(f)$ is a linear term of height one without any ground subterm, i.e. there exist $g \in \Sigma_p$, $p \neq 0$, and p distinct variables $x_1, \ldots, x_p$ such that $\psi_r(f) = g(x_1, \ldots, x_p)$.

Proposition 10. *Let $\mathcal{R}$ be a simple TRS and let Ψ_r and Ψ_l be the homomorphisms associated with $\mathcal{R}$. Let L' be a regular language and let $\mathcal{A}$ be a tree automaton recognizing L'. There exists a tree automaton $\mathcal{D}$ whose set of states is included in the set of states of $\mathcal{A}$ recognizing $\Psi_r(\Psi_l^{-1}(L'))$.*

Proof. The standard construction that proves regular languages are closed under linear homomorphisms and inverse morphisms applies here. The reader is referred for instance to [GS84,CDG+97].

According to Proposition 10, for each integer k, the language L_k is recognized by a tree automaton $\mathcal{A}_k$ such that the set of states of $\mathcal{A}_{k+1}$ is included in the set of states of $\mathcal{A}_k$. Because the sequence of languages is ordered by inclusion and because for a given number of states, the number of tree automata recognizing a distinct language is bounded, there exists an integer i such that $\forall k \geq i$, $L_k = L_i$.

But $\forall k \in N$, $\mathcal{R}^k(L_0) \subseteq L_k$ since $\mathcal{R}(L_k) \subseteq L_{k+1}$. Hence $\mathcal{R}^k(L_0) \subseteq L_i$ since the sequence is growing. We deduce that $\mathcal{R}^*(L_0) \subseteq L_i$. Finally $L_i = \mathcal{R}^*(L_0) = \mathcal{R}^*(L)$ since $L_i \subseteq \mathcal{R}^*(L_0)$. Hence $\mathcal{R}^*(L)$ is regular.

Proposition 11. *Let $\mathcal{R}$ be a simple TRS. If L is a regular language, then $\mathcal{R}^*(L)$ is regular.*

Last example shows that the TRS has to be left-linear.

Example 6. Let the signature $\Sigma = \{a/0, g/1, f/2\}$, the TRS $\mathcal{R}$ composed of the only rule $f(x,x) \to g(x)$ and the language:

$$L = \{t \in T(\Sigma) \mid t = f(f(\ldots f(a, t_1), \ldots, t_{n-1}), t_n) \text{ where } t_1, \ldots, t_n \in T(\{g, a\})\}.$$

We associate with the rule of $\mathcal{R}$ the symbol $\bullet$. The homomorphisms Ψ_l et Ψ_r are determined by:

$$\begin{array}{ll} \psi_l(a) = a & \psi_r(a) = a \\ \psi_l(g) = g(x_1) & \psi_r(g) = g(x_1) \\ \psi_l(f) = f(x_1, x_2) & \psi_r(f) = f(x_1, x_2) \\ \psi_l(\bullet) = f(x_1, x_1) & \psi_r(\bullet) = g(x_1) \end{array}$$

For each integer n, let us denote by t_n the term $f(f(\ldots f(a,a), \ldots, g^{n-1}(a)), g^n(a))$ of L. Let $n \in N$. The term t_n rewrites in $g^{n+1}(a)$ by $n{+}1$ applications of $\mathcal{R}$'s rule hence $t_n \in \mathcal{R}^*(L)$. Moreover t_n gives $g^{n+1}(a)$ by exactly $n{+}1$ applications of $\Psi_r(\Psi_l^{-1})$, i.e $t_n \in L_{k+1}$ and $t_n \notin L_k$. We deduce that the sequence $(L_k)_{k \in N}$ is infinite.

5.1 Summary

Our other results and those of Z. Fülöp *et al.* [FJSV98] are summarized in the following table.

	ordinary	right semi-lin.	right lin.	left lin.	left lin. right semi-lin.	linear
$\mathcal{R}$	$\subseteq$	$\subseteq, =$	$\subseteq, =$	$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathcal{R}_{\|\|}$	$\subseteq$	$\subseteq, =$	$\subseteq, =$	$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathsf{SF}_{\mathsf{io}}$				$\subseteq$	$\subseteq, =$	$\subseteq, =$
$\mathsf{SF}_{\mathsf{oi}}$			$\subseteq, =$			$\subseteq, =$
$\mathsf{SF}_{\uparrow}$				$\subseteq$[FJSV98]	$\subseteq, =$	$\subseteq, =$
$\mathsf{NF}_{\uparrow}$				$\subseteq$[FJSV98]	$\subseteq, =$	$\subseteq, =$
$\mathsf{SF}_{\downarrow}$			$\subseteq, =$	$\subseteq$[FJSV98]	$\subseteq$	$\subseteq, =$
$\mathsf{NF}_{\downarrow}$			$\subseteq$	$\subseteq$[FJSV98]	$\subseteq$	$\subseteq, =$

References

[AD82] A. Arnold and M. Dauchet. Morphismes et bimorphismes d'arbres. *Theorical Computer Science*, 20:33–93, 1982.

[BST99] B. Bogaert, F. Seynhaeve, and S. Tison. The recognizability problem for tree automata with comparisons between brothers. In W. Thomas, editor, *Proceedings, Foundations of Software Science and Computation Structures*, number 1578 in Lecture Notes in Computer Science, Amsterdam, 1999. Springer Verlag.

[CDG$^+$97] H. Comon, M. Dauchet, R. Gilleron, , F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997.

[CDGV94] J.L. Coquidé, M. Dauchet, R. Gilleron, and S. Vágvolgyi. Bottom-up tree pushdown automata : Classification and connection with rewrite systems. *Theorical Computer Science*, 127:69–98, 1994.

[DCC95] M. Dauchet, A.-C. Caron, and J.-L. Coquidé. Reduction properties and automata with constraints. *Journal of Symbolic Computation*, 20:215–233, 1995.

[DT90] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 242–248. IEEE Computer Society Press, 1990.

[Eng75] J. Engelfriet. Bottom-up and top-down tree transformations. a comparision. *Mathematical System Theory*, 9:198–231, 1975.

[Eng78] J. Engelfriet. A hierarchy of tree transducers. In *Proceedings of the third Les Arbres en Algèbre et en Programmation*, pages 103–106, Lille, 1978.

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

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

[FJSV98] A. Fülöp, E. Jurvanen, M. Steinby, and S. Vágvölgy. On one-pass term rewriting. In L. Brim, J. Gruska, and J. Zlatusaksv, editors, *Proceedings of Mathematical Foundations of Computer Science*, volume 1450 of *Lecture Notes in Computer Science*, pages 248–256. Springer Verlag, 1998.

[Gen98] T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In Nipkow [Nip98], pages 151–165.

[GS84] F. Gécseg and M. Steinby. *Tree Automata*. Akademiai Kiado, 1984.

[Jac96a] F. Jacquemard. *Automates d'arbres et réécriture de termes.* PhD thesis, Université de Paris XI, 1996.

[Jac96b] F. Jacquemard. Decidable approximations of term rewriting systems. In H. Ganzinger, editor, *Proceedings. Seventh International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, 1996.

[Nip98] T. Nipkow, editor. *Proceedings. Ninth International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, Tsukuba, 1998.

[Sal88] K. Salomaa. Deterministic tree pushdown automata and monadic tree rewriting systems. *Journal of Comput. and Syst. Sci.*, 37:367–394, 1988.

[Wal98] J. Waldmann. Normalization of s-terms is decidable. In Nipkow [Nip98], pages 138–150.