Regular Tree Languages and Rewrite Systems *
†

Rémi Gilleron, Sophie Tison
LIFL, URA 369 CNRS, IEEA Université de Lille I.
email: {gilleron,tison}@lifl.lifl.fr

Abstract

We present a collection of results on regular tree languages and rewrite systems. Moreover we prove the undecidability of the preservation of regularity by rewrite systems. More precisely we prove that it is undecidable whether or not for a set E of equations the set $E(R)$ congruence closure of set R is a regular tree language whenever R is a regular tree language. It is equally undecidable whether or not for a confluent and terminating rewrite system S the set $S(R)$ of ground S-normal forms of set R is a regular tree language whenever R is a regular tree language. Finally we study fragments of the theory of ground term algebras modulo congruence generated by a set of equations which can be compiled in a terminating, confluent rewrite system which preserves regularity.

1 Introduction

Equations and rewrite systems have been extensively used to specify programs and data types (see Huet and Oppen [22], Dershowitz and Jouannaud [16] for overviews). The paradigm of rewrite systems modelizes evaluation in Logic Programming as well as interpretors in functional programming.

An equation $u = v$ is said to be valid in a variety, i.e., a class of algebras that are models of a set E of equations, if and only if $u = v$ is true in all these models. This is equivalent to checking whether $u = v$ can be derived from the equations in E using instantiation and replacement of equals by equals as inference rules. When E can be compiled in a terminating (noetherian) and confluent rewrite system S, S provides a decision procedure for the equational theory of set E. Termination and confluence are undecidable properties. That is why research work is devoted to the study of partial properties or subclasses of rewrite systems. The most popular result is the Knuth-Bendix completion procedure. Similar techniques are extended to equational rewriting, many-sorted signatures and order-sorted signatures.

For applications such as abstract data types, attention is focused on a specific model, the initial algebra defined by the set of equations and some kind of induction becomes necessary. The observation that an equation $u = v$ is valid in the

*This research was partially supported by "PRC/GDR Mathématiques et Informatique" and ESPRIT Basic Research Action 6317 ASMICS2.

†A preliminary version of this paper was published in the proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science STACS' 91, Theorem 7 was published in the proceedings of the International Conference FCT'89.

initial algebra if and only if adding it to the set of axioms does not result in an inconsistency is the basis of the proof by consistency method pioneered by Musser [31]. In this framework induction completion procedures based on the concept of ground-reducibility have been developped (Jouannaud and Kounalis [23]). A term t is S-ground reducible if all its ground instances are reducible for a rewrite system S. Ground reducibility is decidable for finite R (Plaisted [32], Kapur et al. [24]).

In this context (and others) one has to consider a set of ground terms in the initial algebra. A set of ground terms can be considered as a tree language. The class of regular (recognizable) tree languages defined by tree automata is a well-known class. Regular tree languages are a natural generalization to the tree case of regular languages of words over finite alphabet and all basic results about regular languages have their counterparts in the tree case. This class is closed under boolean operations. Decision algorithms are known for emptiness, inclusion. Henceforth it seems natural to use them when studying problems in initial algebras. We will show throughout this paper that regular tree languages are suitable for linear terms, linear equations or linear rewrite rules. We will consider only the mono-sorted case. But we must notice that an order-sorted signature can be viewed as a bottom-up tree automaton and the well-formed terms of sort s is a recognizable tree language (see Comon [10]).

In Section 3, we present some results on regular tree languages and rewrite systems. We give a summary of the results of Lassez and Marriott [29] on the explicit representation of terms defined by counter examples. Using tree automata techniques Lugiez and Moysset obtained decidability results for complement problems in AC-theories [30]. The set $IRR(S)$ of ground S-normal forms is a regular tree language if S is a left-linear rewrite system. The proof of this result is straightforward but from this result we can infer that ground reducibility of a linear term t for a left-linear rewrite system S is decidable. Therefore classical algorithms on regular tree languages can be applied (see Kucherov [26]). Moreover the result can be easily extended to the case of a (non linear) term t. The problem appears to be much more complicated in the case of non left-linear rewrite systems. It was solved by Plaisted [32] and some complexity measures are presented in Kapur et al. [24]. Many algorithms were proposed in the left-linear case, i.e. when the set of ground normal forms is a regular tree language. The question is : is it decidable whether or not the set of ground S-normal forms is a regular tree language ? Indeed this problem is decidable and three independent proofs were proposed (Kucherov and Tajine [28], Hofbauer and Huber [21], Vágvölgyi and Gilleron [35]). These proofs were based on a result by Kucherov [27] stating that if the set $IRR(S)$ of ground S-normal forms is a regular tree language, a left-linear rewrite system S' exists such that $IRR(S) = IRR(S')$.

In Section 4 we study preservation of regularity by rewrite systems. Let S be a rewrite system and R be a set of terms, a term v is said to be a descendant of a term u if there is a derivation from u to v in S, and the set of descendants of set R is denoted by $S^*(R)$. We prove that it is undecidable whether or not for a set E of equations the set $E(R)$ congruence closure of set R is a regular tree language whenever R is a regular tree language (Section 4.1). It is equally undecidable whether or not for a confluent and terminating rewrite system S the set $S(R)$ of ground S-normal forms of set R is a regular tree language whenever R is a regular tree language (Section 4.2). The general properties for the preservation of regularity by rewrite systems are not surprisingly undecidable. Nevertheless we present in Section 4.3 some results

for subclasses of rewrite systems. A ground rewrite system is one of which all rules are ground (without variables). Dauchet and Tison proved that the theory of ground rewrite systems is decidable ([15]). In the proof, the authors encode the ground rewrite relation in a recognizable tree language by means of ground tree transducers. Using this new tool, one can easily prove the following. Let E be a set of ground equations and S be a ground rewrite system, for every regular tree language R, the sets $E(R)$ congruence closure of set R, $S^*(R)$ of descendants of set R and $S(R)$ of ground S-normal forms of set R are regular tree languages (see also Brainerd [5]). Moreover one can design algorithms which with input a bottom-up automaton for R and a ground rewrite system S output bottom-up tree automata for the sets $S^*(R)$ and $S(R)$ ([11]). Term monadic rewrite systems are an extension to the tree case of string monadic rewrite systems (see Book [2] for an overview on string rewrite systems). A (term) rewrite system S is monadic if, for every rule $l \to r$ in S, $depth(l) \geq 1$ and $depth(r) \leq 1$. Let S be a right-linear monadic rewrite system, for every regular tree language R, the set $S^*(R)$ of descendants of set R is a regular tree language (K. Salomaa [33]). The two above results were extended to the case of semi-monadic rewrite systems (Coquidé et al. [12]). A (term) rewrite system S is semi-monadic if, for every rule $l \to r$ in S, $depth(l) \geq 1$ and variables occur at depth at most 1 in r. Let S be a linear semi-monadic rewrite system, for every regular tree language R, the set $S^*(R)$ of descendants of set R is a regular tree language.

In Section 5 we study some applications of the preservation of regularity. The first-order Reachability Problem for rewrite systems is to determine, given a rewrite system S and two ground terms u and v, whether u can be reduced in v. A second-order Reachability Problem for rewrite systems is to determine, given a rewrite system S and two regular tree languages R and R', whether the set $S^*(R)$ of descendants of set R is included in the set R'. From a theoretical point of view the first-order reachability problem is the simplest problem to consider for rewriting theories (see Caucal [8] for an overview on theory of rewriting). From a practical point of view, if you consider a regular tree language R as set of ground terms of sort r, a second-order reachability problem can be formulated by : is the set of reductions of terms of sort r included in the set of terms of sort r' ? Using the results of Section 4.3 it immediatly follows that the Reachability Problems are decidable for ground, right-linear monadic and linear semi-monadic rewrite systems. In Section 5.3 we study fragments of the theory of ground terms algebras modulo some congruence generated by a set of equations. Related works are the decidability results of Comon [9] and the undecidability results of Treinen [34]. Treinen gives a general method for undecidability proofs of first order theories. His technique demonstrates the undecidability of the Σ_3-fragment of the theory of ground term algebras in the AC-case and of the Σ_2-fragment in the A-case. Comon proves the decidability of the Σ_1-fragment in the AC-case. Here we consider a set E of equations which can be compiled in a terminating, confluent rewrite system S such that for every regular tree language R, the set $S(R)$ of ground S-normal forms of set R is a regular tree language. We prove that the Σ_1-fragment of the theory of ground term algebras modulo congruence generated by such a set of equations is undecidable. Nevertheless we obtain a decidability result when restricted to a class of linear formulas. This result is an extension to the tree case of a result by Book [3] on decidable sentences of Church-Rosser congruences.

2 Preliminaries

Although some notations and basic definitions are mentioned, the reader is supposed to be familiar with the basic concepts of rewrite systems and of tree language theory (see, e.g. Dershowitz, Jouannaud [16], Huet, Oppen [22], Gécseg, Steinby [19]).

2.1 Terms, Substitutions

An alphabet A is a finite set. A^* is the free monoid generated by A with empty word ϵ as identity element. The length of a word $w \in A^*$, denoted by $length(w)$, is defined as usual. The set of non-negative integers is denoted by ω. A ranked alphabet is a finite set Σ in which every symbol has a unique rank in ω. For $m \geq 0$, Σ_m denotes the set of all elements of Σ which have rank m. We specify a countable set $\mathcal{X} = \{ x_1, x_2, \dots \}$ of variables which will be kept fixed in this paper. Moreover, we put $\mathcal{X}_m = \{x_1, \dots, x_m\}$, for $m \geq 0$. $\mathcal{T}(\Sigma, \mathcal{X})$ denotes the free algebra over $\mathcal{X}$ also called term algebra. The initial algebra will be denoted by $\mathcal{T}(\Sigma)$; its elements are called ground terms. We shall need a few functions on terms. For a term $t \in \mathcal{T}(\Sigma, \mathcal{X})$, the depth $depth(t) \in \omega$ the set $var(t) \subseteq \mathcal{X}$, and the set of positions $O(t)$ are defined by induction:

(a) if $t \in \Sigma_0 \cup \mathcal{X}$, then $depth(t) = 0$, $var(t) = \emptyset$ if $t \in \Sigma_0$ and $var(t) = t$ if $t \in \mathcal{X}$, and $O(t) = \{ \epsilon \}$;

(b) if $t = b(t_1, \dots, t_m)$ with $m \geq 1$ and $b \in \Sigma_m$, then $depth(t) = 1 + max\{ depth(t_i) \mid 1 \leq i \leq m \}$, $var(t) = var(t_1) \cup \dots \cup var(t_m)$, and $O(t) = \{ \epsilon \} \cup \{ i\alpha \mid 1 \leq i \leq m$ and $\alpha \in O(t_i) \}$. We note that $depth(t) = max\{ length(\alpha) \mid \alpha \in O(t) \}$.

If $\alpha \in O(t)$, $t_{/\alpha}$ denotes the subterm of t at position α. The head $\mathcal{H}ead(t)$ of a term t is the symbol at the position ϵ in t, i.e. if $t \in \Sigma_0 \cup \mathcal{X}$, then $\mathcal{H}ead(t) = t$, if $t = b(t_1, \dots, t_m)$ with $m \geq 1$ and $b \in \Sigma_m$, then $\mathcal{H}ead(t) = b$. A term t is called linear if variables of $var(t)$ have exactly one occurrence in t. A context c is a linear term in $\mathcal{T}(\Sigma, \mathcal{X}_1)$.

A substitution is a map $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$ which is different from the identity only for a finite subset $Dom(\sigma)$ of $\mathcal{X}$. We do not distinguish σ from its canonical extension to $\mathcal{T}(\Sigma, \mathcal{X})$. For any trees $t \in \mathcal{T}(\Sigma, \mathcal{X}_m)$, $t_1, \dots, t_m \in \mathcal{T}(\Sigma, \mathcal{X})$ and for the substitution σ with $Dom(\sigma) = \mathcal{X}_m$ and $\sigma(x_i) = t_i$ for $i = 1, \dots, m$ we denote the term $\sigma(t)$ by $t(t_1, \dots, t_m)$ as well. A substitution σ is ground if for every variable $x \in Dom(\sigma)$, $\sigma(x)$ is a ground term. A term t matches a term s if $\sigma(s) = t$ for some substitution σ, in that case we also say that t is an instance of s. We denote by $G(t)$ the set of ground instances of a term t. An equation $t = t'$ has a solution if $\sigma(t) = \sigma(t')$ for some substitution σ, in this case, we say that the two terms t and t' are unifiable. The least solution with respect to subsumption is called the most general unifier of t, t'.

2.2 Rewrite Systems

Let Σ be a ranked alphabet. Then a rewrite system S over Σ is a finite subset of $\mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$ such that for each $(l, r) \in S$, each variable of r also occurs in l. Elements (l, r) of S are called rules and are denoted by $l \to r$. S induces a binary relation $\to_S$ over $\mathcal{T}(\Sigma, \mathcal{X})$ defined as follows: for any $t, t' \in \mathcal{T}(\Sigma, \mathcal{X})$, $t \to_S t'$ if and only if there exist a context c, a rule $l \to r$ in S and a substitution σ such that $t = c(\sigma(l))$ and $t' = c(\sigma(r))$. $\overset{*}{\to}_S$ is the reflexive and transitive closure of $\to_S$, $\to_S^n$

denotes the n-fold composition of $\rightarrow_S$. if $s \xrightarrow{*}_S t$, s is said to be an ancestor of t, and t is said to be a descendant of s.

A ground rewrite system is one of which all rules are ground (i.e. elements of $\mathcal{T}(\Sigma) \times \mathcal{T}(\Sigma)$). A left-linear (respectively right-linear, linear) rewrite system is one in which no variable occurs more than once on any left-hand side (respectively right-hand side, right-hand side and left-hand side).

A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is called irreducible for S if t' with $t \rightarrow_S t'$ does not exist. For ground terms $s, t \in \mathcal{T}(\Sigma)$, we say that t is a normal form of s with respect to S if $s \xrightarrow{*}_S t$ and t is irreducible for S. The set of all irreducible, ground terms for S is denoted by $IRR(S)$ and is also called the set of ground S-normal forms. The set of all reducible, ground terms for S is denoted by $RED(S)$.

Let R be a set of ground terms, the set of descendants of set R is denoted by $S^*(R)$, i.e $S^*(R) = \{t \in \mathcal{T}(\Sigma) \mid \exists u \in R \; u \xrightarrow{*}_S t\}$. The set of ground S-normal forms of set R is denoted by $S(R)$, i.e $S(R) = \{t \in \mathcal{T}(\Sigma) \mid t \in IRR(S)$ and $\exists u \in R \; u \xrightarrow{*}_S t\}$, i.e. $S(R) = S^*(R) \cap IRR(S)$.

A term t in $\mathcal{T}(\Sigma, \mathcal{X})$ is ground reducible for S if each of its ground instances in $\mathcal{T}(\Sigma)$ is reducible for S, that is, if for each ground substitution $\sigma : X \rightarrow \mathcal{T}(\Sigma)$ with $Dom(\sigma) = var(t)$ then $\sigma(t)$ is reducible for S.

Let $\rightarrow$ be a binary relation on a set A. We say that $\rightarrow$ is

(i) confluent if, for every $u, v_1, v_2 \in A$, it holds that if $u \xrightarrow{*} v_1$ and $u \xrightarrow{*} v_2$, then there exists a $v_3 \in A$ such that $v_1 \xrightarrow{*} v_3$ and $v_2 \xrightarrow{*} v_3$;

(ii) terminating if there is no infinite sequence $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots$;

(iii) convergent if $\rightarrow$ is confluent and terminating.

A rewrite system S is confluent (respectively terminating, convergent) if the induced rewrite relation is confluent (respectively terminating, convergent).

2.3 Regular Tree Languages

Regular tree languages are a natural generalization to the tree case of regular languages of words over finite alphabet. All basic results about regular languages have their counterparts in the tree case. We essentially use the presentation of regular tree languages with tree automata, the presentation with regular tree grammars or with regular expressions is left out.

A bottom-up tree automaton is a quadruple $A = (\Sigma, Q, Q_f, \Delta)$, where Σ is a ranked alphabet, Q is a finite set of states of rank 0, $Q_f (\subseteq Q)$ is the set of final states, Δ is a finite set of rules of one of the following two types:

(i) $b(q_1, \ldots, q_n) \rightarrow q$ with $n \geq 0$, $b \in \Sigma_n$, $q_1, \ldots, q_n \in Q$.

(ii) $q \rightarrow q'$ with $q, q' \in Q$ (ϵ-rules).

We consider Δ as a ground rewrite system over $\mathcal{T}(\Sigma \cup Q)$. The move relation $\rightarrow_A$ of A is the rewrite relation $\rightarrow_\Delta$, i.e., $\rightarrow_A = \rightarrow_\Delta$. We say that A is deterministic if Δ has no ϵ-rule and no two rules with the same left-hand side.

The tree language recognized by A is $L(A) = \{ t \in \mathcal{T}(\Sigma) \mid (\exists q \in Q_f) \; t \xrightarrow{*}_A q\}$. A tree language R is regular (recognizable) if there exists a bottom-up tree automaton A such that $L(A) = R$.

Morphisms are defined in the usual algebraic way ($\mathcal{T}(\Sigma)$ regarded as a Σ-algebra). Each morphism h can be specified by giving a term $t_b = h(b(x_1, \ldots, x_n))$ in $\mathcal{T}(\Gamma, \mathcal{X}_n)$ for each $b \in \Sigma_n$. h is linear if t_b is linear for every $b \in \Sigma_n$.

We recall from Gécseg, Steinby [19] the following results. For each bottom-up tree automaton A there exists a deterministic bottom-up tree automaton B

such that $L(A) = L(B)$. The class of regular tree languages is closed under boolean operations, and linear morphisms, and there exist decision algorithms for emptiness, equality, inclusion of regular tree languages.

3 Ground normal form languages

3.1 Sets of Ground Instances and Complement Problems

Let R be a finite set of terms in $\mathcal{T}(\Sigma, \mathcal{X})$, the set of ground instances of set R is denoted by $G(R)$. If R is a set of linear terms then $G(R)$ is a regular tree language.

Theorem 1 *(Lassez and Marriott [29])*

- *Let R be a finite set of linear terms, there exists a finite set of linear terms $\bar{R}$ such that*
$$\overline{G(R)} = G(\bar{R}) \text{ where } \overline{G(R)} = \mathcal{T}(\Sigma) - G(R) \text{ (*) ;}$$
- *For any finite set R, there exists a finite set R satisfying (*) if and only if there exists a finite set of linear terms L satisfying $G(L) = G(R)$;*
- *There is an algorithm which tests for any finite set R the existence of a finite set R satisfying (*) and computes it when it exists.*

We mention also the two following results.

Theorem 2 *(Lugiez and Moysset [30]) Let R be a finite set of linear terms in $\mathcal{T}(\Sigma, \mathcal{X})$, the set $G_{AC}(R)$ of ground AC-instances of set R is a regular tree language.*

This result is not difficult to prove but we must note that it is not a consequence of the same result in the empty theory because the AC-congruence closure of a regular tree language is not in general a regular tree language (Deruyver and Gilleron [17]).

Theorem 3 *(Lugiez and Moysset [30]) The complement problem $t \neq t_1 \wedge \ldots \wedge t \neq t_p$ where the t_i's are linear is decidable, i.e. it is decidable whether or not there is a ground instance of t modulo AC which is not an instance of any t_i modulo AC.*

3.2 Ground Normal Form Languages and Ground Reducibility

Theorem 4 *If S is a left-linear rewrite system then $IRR(S)$ is a regular tree language.*

This result is a consequence of the closure properties of the class of regular tree languages. Hence an algorithm for ground reducibility of a linear term t for a left-linear rewrite system S is :

- First step (Compilation). Compute a bottom-up automaton for the set $IRR(S)$
- Second step (Decision). Compute a bottom-up automaton for the set $G(t)$ and use the decision algorithm for the problem $G(t) \cap IRR(S) = \emptyset$.

Testing for ground reducibility requires exponential time, even for left-linear rewrite systems (Kapur et al. [25]). For the first step of the above algorithm, first compute a (non deterministic) bottom-up automaton for $RED(S)$, second use the determinization process, and third deduce a deterministic bottom-up automaton for $IRR(S)$. For determinization of bottom-up automata it is possible to carry out the subset construction as for usual finite automata. This construction may lead to an automaton with a number of states which is an exponential in the number of states of the original automaton. We note that, in the worst case, the number of states of a bottom-up automaton (even a non deterministic one) for $IRR(S)$, is exponential. After the compilation procedure, the second step of the above algorithm is in linear time on the size of t plus the size of the automaton for $IRR(S)$. This algorithm can be improved to deal with a non-linear term t. Using Theorem 2 one deduces an algorithm for ground reducibility modulo AC of a linear term for a left-linear rewrite system. Ground reducibility modulo AC was proved decidable for a left-linear rewrite system by Jouannaud and Kounalis [23] and was proved undecidable in the general case by Kapur et al. [24].

The situation (in the empty theory) is generally much more complex and was solved by Plaisted [32]. Consequently it is interesting to characterize rewrite systems such that the set $IRR(S)$ is a regular tree language :

Theorem 5 *(Kucherov [27]) Let S be a rewrite system, if $IRR(S)$ is a regular tree language, then there exists a left-linear rewrite system S' such that $IRR(S) = IRR(S')$. Moreover S' can be choosed in such a way that any left-hand side of a rule in S' is an instance of a left-hand side of a rule in S.*

Theorem 6 *(Kucherov and Tajine [28]) The following problem is decidable.*

- **Instance.** *A rewrite system S.*
- **Question.** *The set $IRR(S)$ of ground S-normal forms is a regular tree language.*

Which was also proved by Hofbauer and Huber [21], Vágvölgyi and Gilleron [35]. All these proofs were based on Theorem 5 and on the proof of decidability of ground reducibility. Finally, a new approach can be emphasized: Caron et al. [6] have defined encompassment automata which are able to recognize the sets of ground normal forms and which keep most of the good closure and decision properties of usual tree automata.

4 The preservation of regularity by rewrite systems

4.1 Congruence closure of regular tree languages

Courcelle studied in [14] the recognizability in free T-algebras, with T a theory presented by a signature F and a set of equations E ; he proved that the preservation of regularity by E ensures coincidence between equational sets and recognizable sets. However, this property of E is undecidable, even if E is linear. The proof of Theorem 7 is rather technical. We reduce the Post correspondence problem in our problem, i.e. we associate with any instance PCP a system of equations

which preserves regularity if and only if PCP has a solution. To understand the underlying difficulties, we compare our problem with three undecidable problems. First, given a regular tree language R and a set of equations E, it is undecidable whether the set $E(R)$ congruence closure of set R is regular. This result (Theorem 8) can be obtained with the proof of Theorem 7 but a direct proof would be easier choosing an appropriate R. Second, given a rewrite system S, it is undecidable whether, for every regular tree language R, the set $S^*(R)$ of descendants of set R is regular. This result is a corollary of Theorem 7 (consider rewrite system S obtained by orientation of equations in E in the two directions). In this case too, a direct proof should be easier. One could take advantage of orientation of rules in S and simplify rules 7 and 8 of the below construction. Third, given a regular language R and a rewrite system S, it is undecidable (even in the word case, even for restricted subclasses of rewrite systems) whether the set $S^*(R)$ of descendants of set R is regular.

Theorem 7 *The following problem is undecidable.*

- **Instance.** *A set E of equations.*
- **Question.** *For every regular tree language R, the set $E(R)$ congruence closure of set R is a regular tree language.*

PROOF. Let $PCP = (A, I, \phi, \psi)$ be an instance of the Post correspondence problem where A is an alphabet, $I = \{1, \ldots, p\}$, ϕ and ψ two homomorphisms from I^* into A^*. We consider as usual I and A as unary alphabets. Let $\Sigma = I \cup A \cup \{a, b, c, d\}$ where a, b, c and d are new symbols of arity 0, 2, 3 and 3 respectively. Let E be the set of equations on $\mathcal{T}(\Sigma, \mathcal{X})$ defined with the following metaequations.

1. $b(x, y) = b(i(x), \phi(i)(y)), \forall i \in I$;
2. $b(i(x), y) = c(a, i(x), y), \forall i \in I$;
3. $c(x, i(y), \psi(i)(z)) = c(i(x), y, z), \forall i \in I$;
4. $c(x, a, a) = d(a, a, a)$;
5. $d(x, y, z) = d(l(x), y, z)$,
$d(x, y, z) = d(x, l(y), z)$,
$d(x, y, z) = d(x, y, l(z)), \forall l \in A \cup I$;
6. $d(a, y, z) = d(d(a, a, a), y, z)$;
$d(a, y, z) = d(y, d(a, a, a), z)$;
$d(a, y, z) = d(y, z, d(a, a, a))$;
7. $d(l(\vec{x}), l'(\vec{y}), a) = b(l(\vec{x}), l'(\vec{y})), \forall l \notin I \cup \{a\}$;
$d(l(\vec{x}), l'(\vec{y}), a) = b(l(\vec{x}), l'(\vec{y})), \forall l' \notin A \cup \{a\}$;
$d(i(x), v(y), a) = b(i(x), v(y)), \forall i \in I, v(y) \in \mathcal{T}(A \cup \{a\}, \mathcal{X}), length(v) \leq length(\phi(i)), \phi(i) \notin vA^*$;
$d(a, l'(\vec{y}), a) = b(a, l'(\vec{y})), l' \neq a$;
8. $d(l(\vec{x}), y, z) = c(l(\vec{x}), y, z), \forall l \notin I \cup \{a\}$;
$d(a, a, z) = c(a, a, z)$;
$d(x, l'(\vec{y}), z) = c(x, l'(\vec{y}), z), \forall l' \notin I \cup \{a\}$;

Let S be the rewrite system obtained from E by orientation of the equations from left to right. Let R_0 be the regular tree language of trees in $\mathcal{T}(\Sigma)$ whose head is of arity 2 or 3.

Lemma 1 *If there is a solution to PCP then $E(\{b(a, a)\}) = R_0$ with the definitions above.*

PROOF.

$E(\{b(a,a)\}) \subseteq R_0$. This inclusion is straightforward, since, for each equation $s = t$ in E, $\mathcal{H}$ead(s) and $\mathcal{H}$ead(t) are symbols of arity 2 or 3.

Let us now prove the converse inclusion. Let t be a tree in $\mathcal{T}(\Sigma)$, we denote by $n(t)$ the number of symbols b, c and d in t.

Claim 1. If $t \in R_0$, $\mathcal{H}ead(t) = d$ and $n(t) = 1$ then $t \in E(\{b(a,a)\})$.

Let t be a term in R_0 verifying $\mathcal{H}ead(t) = d$ and $n(t) = 1$. There is a solution to PCP so let m in I^+ such that $\phi(m) = \psi(m)$. We have:

$b(a,a) \xrightarrow{*}_S b(m(a), \phi(m)(a))$, with rules 1;

$b(m(a), \phi(m)(a)) \rightarrow_S c(a, m(a), \phi(m)(a))$, with rule 2;

but as $\phi(m)(a) = \psi(m)(a)$, we have ($\tilde{m}$ is the mirror image of m)

$c(a, m(a), \phi(m)(a)) \xrightarrow{*}_S c(\tilde{m}(a), a, a)$, with rules 3;

$c(\tilde{m}(a), a, a) \rightarrow_S d(a,a,a)$, with rule 4;

Consequently $b(a,a) \xrightarrow{*}_S d(a,a,a)$. Reducing with rules 5, we get $b(a,a) \xrightarrow{*}_S t$. □

Claim 2. Let $n \geq 1$, if $\forall t \in \mathcal{T}(\Sigma)$ ($\mathcal{H}ead(t) = d$ and $n(t) \leq n$) $\Rightarrow$ ($t \in E(\{b(a,a)\})$) then $\forall t \in \mathcal{T}(\Sigma)(\mathcal{H}ead(t) = b$ or $\mathcal{H}ead(t) = c$ and $n(t) = n \Rightarrow (t \in E(\{b(a,a)\}))$.

Let us suppose that for $n \geq 0$, $\forall t \in \mathcal{T}(\Sigma)$, $\mathcal{H}ead(t) = d$, $n(t) \leq n$, $t \in E(\{b(a,a)\})$, and let us consider a tree t verifying $\mathcal{H}ead(t) = b$ or $\mathcal{H}ead(t) = c$, $n(t) = n$, $t \in E(\{b(a,a)\})$.

First case. $\mathcal{H}ead(t) = b$.

Let $t = b(t_1, t_2)$. Either $t_1 = m(a) \in I^*a, t_2 = m'(a) \in A^*a$ with $m' = \phi(m)$, then $b(a,a) \xrightarrow{*}_S t = b(t_1, t_2)$ with rules 1. Either not, then $t_1 = m(u_1), t_2 = m'(u_2)$ with $m' = \phi(m)$ and (($\mathcal{H}ead(u_1) \notin I \cup \{a\}$) or ($\mathcal{H}ead(u_1) = i \in I$ and $u_2 \notin \phi(i)$. $\mathcal{T}(\Sigma)$) or ($\mathcal{H}ead(u_1) = a$ and $u_2 \neq a$)). Let us now consider the tree $t' = d(u_1, u_2, a)$. We have $n(t') = n$ and using hypothesis $t' \in E(\{b(a,a)\})$. Moreover $t' = d(u_1, u_2, a) \rightarrow_S b(u_1, u_2)$ with the rule 7. As $b(u_1, u_2) \xrightarrow{*}_S t = b(t_1, t_2)$ with rules 1, we obtain $t = b(t_1, t_2) \in E(\{b(a,a)\})$.

Second case. $\mathcal{H}ead(t) = c$.

Let $t = c(t_1, t_2, t_3)$. Either $t_1 = m(a) \in I^+a$. We deduce from the first case that $b(\tilde{m}(t_2), \psi(\tilde{m})(t_3)) \in E(\{b(a,a)\})$. Moreover $b(\tilde{m}(t_2), \psi(\tilde{m})(t_3)) \rightarrow_S c(a, \tilde{m}(t_2), \psi(\tilde{m})(t_3))$ with rule 2.

$c(a, \tilde{m}(t_2), \psi(\tilde{m})(t_3)) \xrightarrow{*}_S c(m(a), t_2, t_3)$ with the rules 3, thus $t \in E(\{b(a,a)\})$.

Either $t_1 = a$ and $t_2 \notin I.\mathcal{T}(\Sigma)$. Using the hypothesis, $d(a, t_2, t_3) \in E(\{b(a,a)\})$.

AS $d(a, t_2, t_3) \rightarrow_S c(a, t_2, t_3)$ using rule 8, we deduce $t = c(a, t_2, t_3) \in E(\{b(a,a)\})$.

Either $t_1 = a$ and $t_2 \in I.\mathcal{T}(\Sigma)$. We deduce from the first case that $b(t_2, t_3) \in E(\{b(a,a)\})$. Using rule 2, we get $b(t_2, t_3) \rightarrow_S c(a, t_2, t_3)$.Thus we deduce $t = c(a, t_2, t_3) \in E(\{b(a,a)\})$.

Either $t_1 \notin I^*a$. Henceforth, there exists u such that $t_1 = m(u)$ with $u \notin I.\mathcal{T}(\Sigma)$.

Let $t' = d(u, \tilde{m}(t_2), \psi(\tilde{m})(t_3))$, we have $n(t') = n$ and $\mathcal{H}ead(t') = d$ and then $t' \in E(\{b(a,a)\})$ using hypothesis.

$t' = d(u, \tilde{m}(t_2), \psi(\tilde{m})(t_3)) \rightarrow_S c(u, \tilde{m}(t_2), \psi(\tilde{m})(t_3))$ with rule 8,

$c(u, \tilde{m}(t_2), \psi(\tilde{m})(t_3)) \xrightarrow{*}_S c(m(u), t_2, t_3) = c(t_1, t_2, t_3) = t$ with rule 3. Consequently $t \in E(\{b(a,a)\})$. □

Claim 3. Let $n > 1$, if $(\forall t \in R_0, n(t) < n \Rightarrow t \in E(\{b(a,a)\}))$then ($\forall t \in \mathcal{T}(\Sigma)(\mathcal{H}ead(t) = d$ and $n(t) = n) \Rightarrow (t \in E(\{b(a,a)\}))$).

Let us suppose that, for $n > 1$, $\forall t \in R_0$, $n(t) < n$, $t \in E(\{b(a,a)\})$, and let us consider a term $t = d(t_1, t_2, t_3)$ with $n(t) = n > 1$. Let us suppose that t_1 contains one occurrence of b, c or d. Thus there exist m and u such that $t_1 = m(u)$ with $m \in (I \cup A)^*, u \in R_0$. We have $n(u) < n$ and then by hypothesis $u \in E(\{b(a,a)\})$. Besides

$d(a, t_2, t_3) \in E(\{b(a,a)\})$ by hypothesis;

$d(a, t_2, t_3) \rightarrow_S d(d(a, a, a), t_2, t_3)$ with rule 6;
$u \in E(\{b(a, a)\}), d(a, a, a) \in E(\{b(a, a)\})$.
Consequently we obtain $d(u, t_2, t_3) \in E(\{b(a, a)\})$.
Moreover we have $d(u, t_2, t_3) \xrightarrow{*}_S d(m(u), t_2, t_3)$ using rule 5.
Henceforth $t \in E(\{b(a, a)\})$.
We have a similar proof if t_2 or t_3 contains one occurence of b, c or d. □
Using Claims 1, 2 and 3, we have a proof of Lemma 1. □

Lemma 2 *If there is a solution to PCP then $E(R)$ is a regular tree language for any regular tree language R over Σ where E is the system above.*

PROOF.

Let R be a regular tree language, let $\#$ be a new symbol of arity 0 and let h be the tree morphism from $\mathcal{T}(\Sigma)$ into $\mathcal{T}(\Sigma \cup \{\#\})$ defined by:

$h(a) = a$;
$h(l(x)) = l(x), \forall l \in A \cup I$;
$h(b(x, y)) = h(c(x, y, z)) = h(d(x, y, z)) = \#$.

Let us consider the set $R_1 = h(R)$. R_1 is a regular tree language because h is a linear morphism. We can note that $R_1 \subseteq (I \cup A)^*.\{\#, a\}$. Let R_2 be the regular tree language obtained from R_1 substituting $\#$ by R_0. We can easily prove with lemma 1 the equality $R_2 = E(R)$ and consequently $E(R)$ is a regular tree language. □

Lemma 3 *If there is no solution to PCP then $E(\{b(a, a)\})$ is not a regular tree language where E is the system above.*

PROOF.

Let $R_1 = \{b(m(a), \phi(m)(a))/m \in I^*\}$,
let $R_2 = \{c(m(a), m'(a), m''(a))/\psi(\tilde{m})m'' = \phi(\tilde{m}m'), mm' \neq \epsilon; m, m' \in I^*\}$,
and let $R = R_1 \cup R_2$. We suppose that PCP has no solution.

Claim 1. $R \subseteq E(\{b(a, a)\})$.

Indeed for every tree t in $R_1 \cup R_2$, b(a,a) can be reduced in t w.r.t S with the metarules 1, 2 and 3.

Claim 2. $R \supseteq E(\{b(a, a)\})$.

$b(a, a)$ belongs to R henceforth it is sufficient to prove the following inclusion

$$\{t \in \mathcal{T}(\Sigma)/\exists s \in R, s \rightarrow_S t \text{ or } t \rightarrow_S s\} \subseteq R$$

We exhaustively study all metarules of S.

1.1 $s \rightarrow_S t$, it is possible if and only if s is in R_1, thus we have
$s = b(m(a), \phi(m)(a))$ with $m \in I^*$ and
$t = b(i(m(a)), \phi(i)(\phi(m)(a))) = b(im(a), \phi(im)(a)) \in R$.

1.2 $t \rightarrow_S s$, the proof is similar.

2.1 $s \rightarrow_S t$, it is possible if and only if s is in R_1 with m in I^+, thus we have
$s = b(m(a), \phi(m)(a))$ with $m \in I^+$ and
$t = c(a, m(a), \phi(m)(a))$with $m \in I^+$ and one can verify that $t \in R_2$.

2.2 $t \rightarrow_S s$, it is possible if and only if $s = c(a, m(a), m''(a))$, moreover as s is in R and then in R_2, we must have $m \in I^+$ and $\phi(m) = m''$.

Henceforth $t = b(m(a), \phi(m)(a)) \in R_1$.

3 Similar verification of the correctness of the definition of R.

4.1 $s \rightarrow_S t$, as s is in R, it is possible if and only if $s = c(m(a), a, a)$ with $m \in I^+$ and $\phi(\tilde{m}) = \psi(\tilde{m})$ and thus there would be a solution to PCP!

4.2 $t \rightarrow_S s$, it is impossible as there is no term in R whose head is d.

5, 6 Impossible.
7.1 Impossible.
7.2 $t \rightarrow_S s$, s is in R so it is possible if and only if $s = b(m(a), \phi(m)(a))$.

If $m = \epsilon$, $s = b(a, a)$, we cannot obtain s with metarule 7. If $m = im'$, $s = b(im'(a), \phi(i)\phi(m')(a))$ and s can not be obtain with metarule 7.
8 Similar proof.

R_1 is not a regular tree language. R is not a regular tree language because R_1 is the intersection of R with the regular tree language of trees whose head is b. □

From lemmas 2 and 3, $E(R)$ is a regular tree language for every regular tree language R if and only if there is a solution to PCP. □

Theorem 8 *The following problem is undecidable.*

- **Instance.** *A set E of equations, a finite tree language R.*
- **Question.** *$E(R)$ congruence closure of set R is a regular tree language.*

PROOF. It is sufficient to consider the set E of equations and the regular tree language $R = \{b(a, a)\}$ defined in the proof of Theorem 7 to prove this result.

□

4.2 Regular tree languages and convergent rewrite systems.

Theorem 9 *The following problem is undecidable.*

- **Instance.** *A convergent (i.e terminating and confluent) rewrite system S.*
- **Question.** *For every regular tree language R, the set $S(R)$ of ground S-normal forms of set R is a regular tree language.*

PROOF. Let $PCP = (A, I, \phi, \psi)$ be an instance of the Post correspondence problem where A is an alphabet, $I = \{1, \ldots, p\}$, ϕ and ψ two homomorphisms from I^* into A^*. We consider as usual I and A as unary alphabets. Let $\Sigma = I \cup A \cup \Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \cup \Sigma_4$ with $\Sigma_0 = \{0, e\}$, $\Sigma_1 = \{f\}$, $\Sigma_2 = \{g, h\}$ and $\Sigma_4 = \{a, b, c, d\}$. Let S_1 be the rewrite system on $T(\Sigma, \mathcal{X})$ defined with the following metarules.

1. $\forall i \in I$, $a(f(x_1), g(e, i(x_2)), g(e, \phi(i)(x_3)), g(x_4, x_5))$

$$\rightarrow b(f(x_1), g(i(e), x_2), g(\tilde{\phi(i)}(e), x_3), g(x_4, x_5));$$

2. $\forall i \in I$, $b(f(x_1), g(x_2, i(x_3)), g(x_4, \phi(i)(x_5)), g(x_6, x_7))$

$$\rightarrow b(f(x_1), g(i(x_2), x_3), g(\tilde{\phi(i)}(x_4), x_5), g(x_6, x_7));$$

3. $b(f(x_1), g(x_2, e), g(x_3, e), g(x_4, x_5)) \rightarrow c(f(x_1), g(x_2, e), g(x_3, e), g(x_4, x_5))$;
4. $\forall i \in I$, $c(f(x_1), g(i(x_2), x_3), g(\tilde{\psi(i)}(x_4), x_5), g(x_6, x_7))$

$$\rightarrow c(f(x_1), g(x_2, i(x_3)), g(x_4, \psi(i)(x_5)), g(x_6, x_7));$$

5. $c(f(x_1), g(e, x_2), g(e, x_3), g(x_4, x_5)) \to a(x_1, g(e, x_2), g(e, x_3), g(f(x_4), f(x_5)))$;
6. $a(e, g(e, x_1), g(e, x_2), g(x_3, x_4)) \to d(x_1, x_2, x_3, x_4)$;
7. $d(i(x_1), l(x_2), x_3, x_4) \to d(x_1, x_2, x_3, x_4)$, $\forall i \in I$, $\forall l \in A$;
$d(i(x_1), e, x_2, x_3) \to d(x_1, e, x_2, x_3)$, $\forall i \in I$;
$d(e, l(x_1), x_2, x_3) \to d(e, x_1, x_2, x_3)$, $\forall l \in A$;
$d(e, e, x_1, x_2) \to h(x_1, x_2)$.

We are now in order to define the rewrite system S. The aim of the construction is that every ground S_1-normal form reduces by S in 0. For this we add to S_1 two rewrite systems S_2 and S_3.

First, let us define S_2. S_1 is a linear rewrite system. Let L_1 be the set of linear left-hand side of rules of S_1 and let L_2 be a finite set of linear terms verifying

$$G(L_2) = \overline{G(L_1)} \cap \{t \in \mathcal{T}(\Sigma)/\mathcal{H}ead(t) \in \{a, b, c, d\}\}$$

The existence of L_2 is a consequence of Theorem 1. We now consider the rewrite system $S_2 = \{l \to 0/l \in L_2\}$. Every term of head a, b, c or d which is not an instance of a left-hand side of a rule in S_1 reduces by S_2 in 0.

Second, $S_3 = \{i(0) \to 0, \forall i \in I;\ l(0) \to 0, \forall l \in A;\ f(0) \to 0;\ g(0, x_1) \to 0;\ g(x_1, 0) \to 0;\ h(0, x_1) \to 0;\ h(x_1, 0) \to 0\}$. The rewrite system S is $S = S_1 \cup S_2 \cup S_3$.

Claim 1. *S is a linear, terminating and confluent rewrite system.*

S is obviously linear.

All the rules are size reducing except metarules 1, 2, 3, 4 and 5 of S_1. However if you consider a cycling derivation (with these rules) starting from a term of head a (or b, c or d), the number of symbols f in the left-most branch of the considered term is decreasing. We can deduce from these remarks that S is terminating.

One can verify that all critical pairs are S-joinable, henceforth S is confluent. Critical pairs are formed from rules in S_3 with rules in S_1 and S_2 and are S-joinable to 0. □

Claim 2. *If for every regular tree language R the set $S(R)$ of ground S-normal forms of set R is regular then there is no solution to PCP.*

Let us suppose that PCP admits a solution m in I^+ i.e $\phi(m) = \psi(m)$. Let us consider $R = \{a(f^n(e), g(e, m(e)), g(e, u(e)), g(e, e))/m \in I^+, u \in A^*, n > 0\}$. R is obviously a regular tree language. Let us consider a term t in R.

First case. $u = \phi(m) = \psi(m)$.

$t \xrightarrow{*}_S t_1 = b(f^n(e), g(\tilde{m}(e), e), g(\tilde{u}(e), e), g(e, e)$ using rules 1, 2 as $u = \phi(m)$;

$t_1 \xrightarrow{*}_S t_2 = c(f^n(e), g(e, m(e)), g(e, u(e)), g(e, e))$ using rules 3, 4 as $u = \psi(m)$;

$t_2 \to_S t_3 = a(f^{n-1}(e), g(e, m(e)), g(e, u(e)), g(f(e), f(e)))$ using rule 5.

Either $n - 1 = 0$ and one can reduce with the rule 6, or $n - 1 \neq 0$ and we can continue the derivation using rules 1 to 5.

We obtain $t \xrightarrow{*}_S d(m(e), u(e), f^n(e), f^n(e))$.

And using rules 7, $t \xrightarrow{*}_S h(f^n(e), f^n(e))$.

In this case $h(f^n(e), f^n(e))$ is the S-normal form of t.

Second case. $u \neq \phi(m)$ or $u \neq \psi(m)$. One first reduce t with S_1 and then obtain a ground S_1-normal form t' of t whose head is a, b or c. $t' \in G(L_2)$ therefore t' reduces to 0.

Consequently $S(R) = \{h(f^n(e), f^n(e))/n > 0\} \cup \{0\}$ and $S(R)$ is not a regular tree language. □

Claim 3. *If there is no solution to PCP then for every regular tree language R the set $S(R)$ of ground S-normal forms of set R is a regular tree language.*

Let R be a regular tree language. S_1 is linear, henceforth $IRR(S_1)$ and $RED(S_1)$ are regular sets. We consider the regular sets $R_1 = R \cap IRR(S_1)$ and $R_2 = R \cap RED(S_1)$,

R is the disjoint union of the regular tree languages R_1 and R_2. We are now interested with the sets $S(R_1)$ and $S(R_2)$.
First we prove that $S(R_1)$ is a regular tree language. Let $T = \mathcal{T}(\{e, f, g, h\} \cup I \cup A)$. $S(R_1 \cap T) = R_1 \cap T$ because a term without any occurrence of a, b, c, d and 0 is irreducible. $S(R_1 \cap \bar{T}) = \{0\}$ if $R_1 \cap \bar{T} \neq \emptyset$, $\emptyset$ otherwise. Indeed if a term t contains one occurrence of 0, its S-normal form is 0. Moreover if a term t contains one occurrence of a, b, c, or d at a position p, t being S_1-irreducible, a rule in S_2 is applicable at position p. Therefore t is reducible in t', t' contains one occurence of 0, therefore its S-normal form is 0. Consequently $S(R_1) = S(R_1 \cap T) \cup S(R_1 \cap \bar{T})$ is regular.
Second we prove that $S(R_2)$ is a regular tree language. Let us consider a term t in R_2 whose head is a, b, c or d and determine its normal form $S(t)$.

We consider the case $\mathcal{H}ead(t) = a$. Let $t = a(t_1, t_2, t_3, t_4)$.
First case. $\mathcal{H}ead(t_1) \neq f$ and $t_1 \neq e$.

t is not an instance of a left-hand side of a rule in S_1, $t \in G(L_2)$ and $S(t) = 0$.
Second case. $\mathcal{H}ead(t_1) = f$.

As PCP has no solution if we reduce t with S_1 at position ϵ, we obtain a ground term t' whose head is a, b or c which is not an instance of a left-hand side of a rule in S_1. Henceforth $S(t) = S(t') = 0$.
Third case. $t_1 = e$.

Either t is not an instance of the left-hand side of rule 6 of S_1 and $S(t) = 0$,

or $t = a(e, g(e, m(e)), g(e, u(e)), g(r, s))$, then $t \rightarrow_S d(m, u, r, s)$ using the rule 6 of S_1. Then, either $m \in I^*$ and $u \in A^\times$ so we have $t \xrightarrow{*}_S h(r, s)$ using rules 7, or S(t)=0.

If $\mathcal{H}ead(t) = a$ then, either $t = a(e, g(e, m), g(e, u), g(r, s))$ with $m \in I^\times(e)$ and $u \in A^*(e)$ and $S(t) = S(h(r, s))$, or S(t)=0.

With a similar proof for the other cases, we obtain :

- (a) $t = a(e, g(e, m(e)), g(e, u(e)), g(r, s))$ with $m \in I^\times$ and $u \in A^*$ then $S(t) = S(h(r, s))$;
- (b) $t = b(f(e), g(\tilde{m}_1(e), m_2(e)), g(\tilde{u}_1(e), u_2(e)), g(r, s))$ with $m_1, m_2 \in I^\times$, $u_1, u_2 \in A^\times$, $u_2 = \phi(m_2)$, $u_1 u_2 = \psi(m_1 m_2)$ then $S(t) = S(h(f(r), f(s)))$;
- (c) $t = c(f(e), g(\tilde{m}_1(e), m_2(e)), g(\tilde{u}_1(e), u_2(e)), g(r, s))$ with $m_1, m_2 \in I^*$ and $u_1, u_2 \in A^\times$, $u_1 = \psi(m_1)$ then $S(t) = S(h(f(r), f(s)))$;
- (d) $t = d(m(e), u(e), r, s)$ with $m \in I^\times$ and $u \in A^*$ then $S(t) = S(h(r, s))$;
- (e) Otherwise $S(t) = 0$.

The technical part of the proof is now with instance a bottom-up tree automaton recognizing R_2 to define a bottom-up automaton recognizing $S(R_2)$, the set of rules of this automaton depending on the existence of terms in R_2 satisfying (a), (b), (c), (d) or (e). This construction will not be detailed here. □

The theorem is a consequence of Claims 1, 2 and 3. □

Theorem 10 *The following problem is undecidable.*

- **Instance.** *A convergent (i.e terminating and confluent) rewrite system S, a regular tree language R.*
- **Question.** *The set $S(R)$ of ground S-normal forms of set R is a regular tree language.*

Proof. It is sufficient to consider the rewrite system S defined in the proof of Theorem 9 and the regular tree language R defined in Claim 2 of the proof of Theorem 9 to obtain this result. □

4.3 The preservation of regularity for subclasses of rewrite systems

Not surprisingly, the general properties for the preservation of regularity by rewrite systems are undecidable. In this section, we recall some results concerning the preservation of regularity by some classes of rewrite systems. First, let us consider the case of ground rewrite systems.

Theorem 11 *(Dauchet and Tison [15], Brainerd [5]) Let E be a set of ground equations and S be a ground rewrite system. For every regular tree language R, the sets $E(R)$ congruence closure of set R, $S^*(R)$ of descendants of set R and $S(R)$ of ground S-normal forms of set R are regular tree languages.*

These results are not unexpected as ground rewrite system is very closed to regular grammars. M.Dauchet and S.Tison proved that the theory of ground rewrite systems is decidable ([15]). They encode the ground rewrite relation in a recognizable tree language by means of ground tree tansducers. Using this new tool, we can prove Theorem 11 and design algorithms which with, given a bottom-up automaton for R and a ground rewrite system S, compute bottom-up tree automata for the sets S*(R) and S(R) ([11]).

We now consider the case of monadic rewrite systems. Monadic rewrite systems were introduced in Book and Gallier [1] and in K. Salomaa [33] in connection with tree pushdown automata. Term monadic rewrite systems are an extension to the tree case of string monadic rewrite systems(see Book [2]).

Definition 1 *A (term) rewrite system S is monadic if, for every rule $l \to r$ in S, $depth(l) \geq 1$ and $depth(r) \leq 1$.*

Theorem 12 *(K.Salomaa, [33]) Let S be a right-linear monadic rewrite system, for every regular tree language R, the set $S^*(R)$ of descendants of set R is a regular tree language.*

Consider the monadic rewrite system $S = \{a(x) \to b(x, x)\}$ and the regular tree language $L = \{\, a(c^n(\$)) \mid n \geq 0 \,\}$. It is clear that S is not right-linear and that $S^*(L)$ is not recognizable. Hence, the result is false if the hypothesis right-linear is omitted. These two results for ground and monadic rewrite systems are embedded with a similar result for semi-monadic rewrite system.

Definition 2 *A rewrite system S over some ranked alphabet Σ is semi-monadic if, for every rule $l \to r$ in S, $depth(l) \geq 1$ and either r is a variable (that is, $r \in X$), or $r = \delta(y_1, \ldots, y_k)$, where $\delta \in \Sigma_k$, $k \geq 0$, and for each $i \in \{1, \ldots, k\}$, either y_i is a variable (i.e., $y_i \in X$) or y_i is a ground term (i.e., $y_i \in \mathcal{T}(\Sigma)$).*

Theorem 13 *(Coquide et al. [12]) Let S be a linear semi-monadic rewrite system, for every regular tree language R, the set $S^*(R)$ of descendants of set R is a regular tree language.*

We conjecture that the same result holds for right-linear semi-monadic rewrite systems. Using Theorems 4 and 13, for every regular tree language R, the set $S(R)$ of ground S-normal forms of set R is a regular tree language whenever S is a linear semi-monadic rewrite system.

5 Applications

5.1 The reachability Problem for rewrite systems

Definition 3 *The first-order Reachability Problem for rewrite systems is :*

- **Instance.** *A rewrite system S, two ground terms u and v.*
- **Question.** *$u \in S^*(\{v\})$, i.e $u \xrightarrow{*}_S v$.*

Definition 4 *A second-order Reachability Problem for rewrite systems is :*

- **Instance.** *A rewrite system S, two regular tree languages R and R'.*
- **Question.** *$S^*(R) \subseteq R'$.*

It is obviously possible to consider other second-order Reachability Problems. For instance, we can consider the question $S^*(R) \cap R' \neq \emptyset$, i.e is there a term in the regular tree language R which can be reduced in a term in the regular tree language R' ? If one considers a regular tree language R as set of ground terms of sort r, second-order reachability problems are : is the set of reductions of terms of sort r included in the set of terms of sort r' ? is there a term of sort r which can be derived with S in a term of sort r' ?, ...

Theorem 14 *If a rewrite system S preserves regularity then the Reachability Problems for S are decidable.*

The above result is an immediate consequence of the decision results on regular tree languages. From Theorems 14, 11, 12 and 13, we obtain :

Theorem 15 *The first-order and second-order Reachability Problems are decidable for ground rewrite systems, right-linear monadic rewrite systems, and linear semi-monadic rewrite systems.*

5.2 Equational Formulas

We consider sets E of equations such that there exist rewrite systems S satisfying the three following statements

- (1) $\overset{*}{\leftrightarrow}_E \; = \; \overset{*}{\leftrightarrow}_S$;
- (2) S is convergent (i.e. terminating and confluent) ;
- (3) For every regular tree language R, the set $S(R)$ of ground S-normal forms of set R is a regular tree language.

An *equational formula* is a first order formula the atoms of which are equations $s = t$, with s, t in $\mathcal{T}(\Sigma, \mathcal{X})$. We assume that negations are propagated using classical rules. Then, introducing new symbol $\neq$, we may assume that equational formulae do not contain any occurence of $\neg$. Finally, equational formulae are assumed to be in prenex normal form. A *sentence* is a first order formula without free variables. An interpretation maps variables on ground terms and $=$ is interpreted as $\overset{*}{\leftrightarrow}_E$.

Note. We do not adopt the order-sorted framework. The results of this section can be easily extended to this framework using symbolic constraints ($t \in \underline{s}$ where $\underline{s}$ is a regular tree language). A reference for equational formulas in order-sorted algebras is Comon [10].

Theorem 16 *Let E be a set of equations satisfying (1), (2) and (3) for some S. The existential fragment of the theory $\mathcal{T}(\Sigma)_{/\overset{*}{\leftrightarrow}_E}$ is undecidable.*

PROOF. Let $PCP = (A, I, \phi, \psi)$ be an instance of the Post correspondence problem where A is an alphabet, $I = \{1, \ldots, n\}$, ϕ and ψ two homomorphisms from I^* into A^*. We consider as usual I and A as unary alphabets. Let $\Sigma = I \cup A \cup \{a, b, c, d\}$ where a, b, c and d are new symbols of arity 0, 0, 2 and 2.

Let E be the set of equations on $\mathcal{T}(\Sigma, \mathcal{X})$ defined with the following metaequations.

1. $c(i(x), \phi(i)(y)) = c(x, y), \forall i \in I$;
2. $d(i(x), \psi(i)(y)) = d(x, y), \forall i \in I$;
3. $c(a, a) = b$;
4. $d(a, a) = b$.

Let S be the rewrite system obtained from E orienting equations from left to right. S is terminating, S is confluent (terminating without critical pairs), $IRR(S)$ is a regular tree language (S is left-linear, Theorem 4), S preserves regularity (S is a monadic term rewrite system, Theorem 12) and obviously $\overset{*}{\leftrightarrow}_E = \overset{*}{\leftrightarrow}_S$. Henceforth S satisfies (1), (2) and (3).

Consider the following sentence

$$\exists x \; \exists y \bigvee_{i \in I} (c(i(x), y) = d(i(x), y))$$

Claim. This formula is true under the interpretation if and only if there is a solution to PCP.

The sentence is true under the interpretation if and only if there exists two terms t and t', a symbol i in I such that $c(i(t), t') \overset{*}{\leftrightarrow}_E d(i(t), t')$. We can easily derive that this statement is true if and only if there exists m in I^+ and u in A^* such that $c(m(a), u(a)) \overset{*}{\leftrightarrow}_E d(m(a), u(a))$. Now, using the properties of E and S, the statement is true if and only if $c(m(a), u(a))$ and $d(m(a), u(a))$ have the same S-normal form b. Hence the sentence is true under the interpretation if and only if $u = \phi(m) = \psi(m)$. □

From now on we consider that in the interpretation for each variable $x \in \mathcal{X}$, there is a regular tree language $D(x)$ and each variable takes value in its domain $D(x)$. We consider the subset $LinF$ of sentences defined as follows.

$\Phi \in LinF$ if and only if ($\exists \vec{x}$ stands for $\exists x_1 \ldots \exists x_n$)

- (a) $\Phi \equiv \exists \vec{x} \; \exists \vec{y} \; \phi$;
- (b) $\Phi \equiv \exists \vec{x} \; \forall \vec{y} \; \phi$;
- (c) $\Phi \equiv \forall \vec{x} \; \exists \vec{y} \; \phi$;
- (d) $\Phi \equiv \forall \vec{x} \; \forall \vec{y} \; \phi$;

where $\phi \equiv l(\vec{x}) = l'(\vec{y})$ or $\phi \equiv l(\vec{x}) \neq l'(\vec{y})$, l and l' are linear terms, $\vec{x}$ and $\vec{y}$ are disjoint.

Theorem 17 *Let E be a set of equations satisfying (1), (2) and (3) for some S. There exists an algorithm that given a sentence in $LinF$ answers whether the sentence is true or false under the interpretation above.*

PROOF.

Let $l(\vec{x}) = l(x_1, \ldots, x_n)$ be a linear term, we define $D(l) = \{l(l_1, \ldots, l_n) / l_i \in D(x_i)\}$. $D(t)$ is a regular tree language because t is linear. Using the hypothesis (3) on S, for each t, $S(D(t))$ is a regular tree language.

Let $\phi \equiv t(\vec{x}) = t'(\vec{y})$ one can easily prove the followings.
case (a). Φ is true under the interpretation if and only if $S(D(t)) \cap S(D(t')) \neq \emptyset$;
case (b). Φ is true under the interpretation if and only if $S(D(t'))$ is a singleton set and $S(D(t)) \subseteq S(D(t'))$;
case (c). Φ is true under the interpretation if and only if $S(D(t)) \subseteq S(D(t'))$;
case (d). Φ is true under the interpretation if and only if $S(D(t)), S(D(t'))$ are singleton sets and $S(D(t)) = S(D(t'))$.

If $\phi \equiv t(\vec{x}) \neq t'(\vec{y})$, the formulas are negations of the previous ones. We now deduce the theorem from the decision properties of regular tree languages. □

Note. Under this interpretation the undecidability result of Theorem 16 can be obtained as follows.

Either we consider the same ranked alphabet Σ, the same set of equations E and the formula

$$\exists x\ \exists y\ c(x, y) = d(x, y)$$

considering the domains $D(x) = \mathcal{T}(A \cup \{a\}) - \{a\}$ and $D(y) = \mathcal{T}(A \cup \{a\})$.

Or we consider ranked alphabet $\Sigma \cup \{f\}$ where f is a new symbol of arity 2, the same set of equations E and the formula

$$\exists x\ \exists y\ f(c(x, y), d(x, y)) = f(b, b)$$

considering the domains $D(x) = \mathcal{T}(A \cup \{a\}) - \{a\}$ and $D(y) = \mathcal{T}(A \cup \{a\})$.

It is interesting to notice that we obtain a decidability result for linear sentences and an undecidability one for non linear sentences. It will be of interest to extend Theorem 17 to a larger class of linear sentences (for instance variables in a same term could be existentially or universally quantified).

6 Conclusion

In this paper we have seen that the use of an algebraic tool (regular tree languages) can clarify or simplify some proofs on rewrite systems. Regular tree languages are suitable when non-linearity is avoided. We think that the study of ground normal form languages may be successfully continued. One way to handle non-linearity is to define new classes of automata. A pioneer attempt was carried out by Bogaert and Tison [1], but their tree automata can only handle non-linear variables which are direct sons of the root of a left-hand side. Caron et al. [6] [7] have recently introduced a new class of automata. They have proved that the theory of encompassment is decidable. This theory is the set of true sentences built up with unary predicates "x encompasses t". In particular ground normal form languages are definable in this theory.

References

[1] B. Bogaert and S. Tison, Equality and Disequality Constraints on Direct Subterms in Tree Automata, *Proc. STACS'93, Lecture Notes in Computer Science* **577** (1992) 161-172.

[2] R.V. Book, Thue systems as rewriting systems, *J. Symb. Comput.* **3** (1987) 39-68.

[3] R.V. Book, Decidable Sentences of Church-Rosser Congruences, *Theoretical Computer Science* **24** (1983) 301-312.

[4] R.V. Book, J. H. Gallier, Reductions in Tree Replacement Systems, *Theoretical Computer Science* **37** (1985) 123-150.

[5] W.S. Brainerd, Tree generating regular systems, *Inf. and Control* **14** (1969) 217-231.

[6] A. C. Caron, J. L. Coquidé, M. Dauchet, Encompassment Properties and Automata with Constraints, *Proc. RTA'93, Lecture Notes in Computer Science* **690** (1993) 328-342.

[7] A. C. Caron, J. L. Coquidé, M. Dauchet, Pumping, Cleaning and Symbolic Constraints Solving, to appear in *Proc. ICALP'94.*

[8] D. Caucal, Monadic Theory of Term Rewritings, in : *Proc. Seventh Annual Symposium on Logic in Computer Science*, (1992) 266-273.

[9] H. Comon, Unification et Disunification : Théorie et Applications, Thèse de Doctorat, Institut Polytechnique de Grenoble, France, (1988).

[10] H. Comon, Equational Formulas in Order-sorted Algebras in : *Proc. ICALP'90,Lecture Notes in Computer Science* **443** (1990) 674-688.

[11] J.L. Coquidé, R. Gilleron, Proofs and Reachability Problems for Rewrite Systems, in : *Proc.IMYCS'90, Lecture Notes in Computer Science* **464** (1990) 120-129.

[12] J.L. Coquidé, M. Dauchet, R. Gilleron and S. Vágvölgyi, Bottom-up tree pushdown automata and rewrite systems, Proc. RTA'91, *Lecture Notes in Computer Science* **488** (1991) 287-298.

[13] J.L. Coquidé, M. Dauchet and S.Tison, About connections between syntactical and computational complexity, Proc. FCT'89, *Lecture Notes in Computer Science* **380** (1989) 105-115.

[14] B. Courcelle, On recognizable sets and tree automata, in "Resolution of Equations in Algebraic Structures", *Academic Press, M.Nivat & H.Ait-Kaci eds.*

[15] M. Dauchet and S. Tison, The Theory of Ground Rewrite Systems is Decidable, *Proc. LICS'90*, Philadelphia (1990) 242-248.

[16] N. Dershowitz and J.P. Jouannaud, Rewrite Systems, in: J.V. Leeuwen, ed., Handbook of *Theoretical Computer Science*, (North Holland, 1990) Vol.B 243-320.

[17] A. Deruyver, R. Gilleron, The Reachability Problem for Ground TRS and some extensions, *Proc. CAAP 89, Lecture Notes in Computer Science* **351** (1989) 227-243.

[18] J.H.Gallier, R.V. Book, Reductions in tree replacement systems, *Theoret. Comput. Sci.* **37** (1985) 123-150.

[19] F. Gécseg, and M. Steinby, *Tree Automata*, Akadémiai Kiadó. Budapest (1984).

[20] R. Gilleron, Decision problems for term rewriting systems and recognizable tree languages, Proc. STACS'91, *Lecture Notes in Computer Science* **480**, (1991) 148-159.

[21] D. Hofbauer and M. Huber, Typical terms, test sets, and linearizations, *Proc. CTRS'92* (1992) 145-149.

[22] G. Huet and D.C. Oppen, Equations and Rewrite Rules: A survey, *Formal Language Theory: Perspectives and Open Problems* (Academic Press, New York, 1980) 309-405.

[23] J. P. Jouannaud, E. Kounalis, Automatic Proofs by Induction in Equational Theories, *Inform. and Comput.* **82** (1989) 1-33.

[24] D. Kapur, P. Narendran and H. Zhang, On Sufficient-Completeness and Related Properties of Term rewriting Systems, *Acta Inform.* **24** (1987) 395-415.

[25] D. Kapur, P. Narendran, D. J. Rosenkrantz and H. Zhang, On Sufficient-completeness ground-reducibility and their complexity, *Acta Inform.* **28** (1991) 311-350.

[26] G. A. Kucherov, A new quasi-reducibility testing algorithm and its application to proofs by induction, *Proc. ALP'88, Lecture Notes in Computer Science* **343** (1988) 204-213.

[27] G. A. Kucherov, On relationship between term rewriting systems and regular tree languages, *Proc. RTA'91, Leclure Notes in Computer Science* **488** (1991) 299-311.

[28] G. Kucherov, M.Tajine, Decidability of Regularity and Related Properies of Ground Normal Form Languages, *CTRS'92* (1992) 150-156.

[29] J. L. Lassez, K. J. Marriott, Explicit Representation of Terms Defined by Counter Examples, *Journal of Automated Reasoning*, **3**, (1987) 301-317.

[30] D. Lugiez, J. L. Moysset, Complement Problems and Tree Automata in AC-like Theories, *Proc. STACS'93, Lecture Notes in Computer Science* **665** (1991) 515-524.

[31] D. R. Musser, On proving inductive properties of abstract data types in : *Proc. Seventh ACM Symposium on Principles of Programming Languges*, (1980) 154-162.

[32] D.A. Plaisted, Semantic Confluence Tests and Completion Methods, *Inform. Control* **65** (1985) 182-215.

[33] K. Salomaa, Deterministic Tree Pushdown Automata and Monadic Tree Rewriting Systems, *J. Comput. System Sci.* **37** (1988) 367-394.

[34] R. Treinen, A new Method for Undecidability Proofs in Firs Order Theories, *Journal of Symbolic Computation* **14** (1992) 437-458.

[35] S. Vàgvölgyi, R. Gilleron, For a rewrite System it is decidable whether the set of irreducible, ground terms is recognizable, in : *Bulletin of EATCS*, **48**, (1992) 197-209.