Classes of Tree Homomorphisms with Decidable Preservation of Regularity*

Guillem Godoy[1], Sebastian Maneth[2], and Sophie Tison[3]

[1] Universitat Politècnica de Catalunya (UPC)
Jordi Girona 1, Barcelona, Spain
ggodoy@lsi.upc.edu
[2] NICTA and UNSW, Sydney, Australia
sebastian.maneth@nicta.com.au
[3] Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq Cedex, France
sophie.tison@lifl.fr

Abstract. Decidability of regularity preservation by a homomorphism is a well known open problem for regular tree languages. Two interesting subclasses of this problem are considered: first, it is proved that regularity preservation is decidable in polynomial time when the domain language is constructed over a monadic signature, i.e., over a signature where all symbols have arity 0 or 1. Second, decidability is proved for the case where non-linearity of the homomorphism is restricted to the root node (or nodes of bounded depth) of any input term. The latter result is obtained by proving decidability of this problem: Given a set of terms with regular constraints on the variables, is its set of ground instances regular? This extends previous results where regular constraints where not considered.

1 Introduction

Representations of sets of terms are used in many areas of computer science. The choice of formalism depends on the expressiveness, but also on the properties from a computational point of view. Tree automata [3,7] are a well studied formalism for representing term languages. They are the natural extension of standard finite automata over words to tree/term languages. For example, the tree automaton

$$\begin{array}{rl} a \to q_a & \\ g(q_a) \to q_g & \qquad g(q_g) \to q_g \\ f(q_g, q_f) \to q_{accept} & \qquad f(q_a, q_a) \to q_f \end{array}$$

recognizes the language $f(g^+(a), f(a, a))$. The languages recognized by tree automata are also called regular. They are a classical concept which has been used in many contexts: for instance, they adequately describe the parse trees of a context-free grammar or the well-formed terms over a sorted signature, and they naturally capture type formalisms for tree-structured (XML) data [14,9]. Similar as in the case of regular sets of

* The first author was supported by Spanish Min. of Educ. and Science by the LogicTools project (TIN2004-03382), and by the FORMALISM project (TIN2007-66523).

R. Amadio (Ed.): FOSSACS 2008, LNCS 4962, pp. 127–141, 2008.

words, the class of regular term languages has many convenient properties such as closure under boolean operations (intersection, union, negation), decidable properties such as inclusion and equivalence, and they are characterized by many different formalisms such as regular grammars, regular term expressions, congruence classes of finite index, deterministic bottom-up finite tree automata, nondeterministic top-down finite tree automata, sentences of monadic second-order logic, etc, cf. [3,7]. Deterministic tree automata, for instance, can effectively be minimized and give rise to efficient parsing procedures.

Nevertheless, the expressiveness of regular tree languages is considerably limited: simple languages like the recursively defined set $T_{\mathtt{bin}} = \{a\} \cup \{f(t,t) \mid t \in T_{\mathtt{bin}}\}$, i.e., the set of complete trees over $\{f, a\}$ where f is a binary symbol and a is a constant, is not regular. Accordingly, it is often convenient to use a more general formalism to describe a set of terms, loosing then some of the nice computational properties. In this setting, it makes sense to study the decidability of the regularity of a given set of terms represented by a concrete (and more expressive) formalism. Practically speaking, we want to allow the specification of term languages in a more general formalism than regular term languages, but want to be able to detect if a specific given language is in fact regular. Deciding a class of languages inside a larger class is a very hard problem; for instance, given a context-free (word) language it follows from Greibach's Theorem that is impossible to decide whether or not the language is in fact regular (see, e.g., [8]).

In this paper we study the decidability of regularity for two particular and related representations: when the set of terms is described as the image of a regular set by a tree homomorphism, and when it is described as the ground instances of a finite set of terms with regular constraints on the variables.

Tree homomorphisms are the natural extension of the usual word homomorphisms to trees. A homomorphism H is usually defined by associating to each symbol g in the input signature a term $H(g)$ with variables $x_1, x_2, \ldots, x_k$, where k is the arity of g. This definition is then homomorphically extended to any term of the signature. For example, the homomorphism $\{H(g) = f(x_1, x_1), H(a) = a\}$ applied to the language $g^*(a) = \{a, g(a), g(g(a)), \ldots\}$ generates as image the language $T_{\mathtt{bin}}$ of above. This example shows that the image of a regular language by a homomorphism is in general *not* regular. In fact, it is a long-standing open problem whether or not it is decidable if the homomorphic image of a regular set of terms is regular ("the HOM-problem"); cf., e.g., Conclusions of [6]. Some particular cases are known to be decidable, for instance when the homomorphism is linear, see [5], or when in the terms $H(g)$, for all input symbols g, multiple occurrences of the same variable all have the same parent node (i.e., are siblings of each other) [2]. If slightly more expressive tree translation devices than homomorphisms are considered, then regularity preservation quickly becomes undecidable: for a deterministic top-down tree transducer with only two states, it is undecidable whether its image of a regular tree language is again regular [6] (homomorphisms naturally correspond to the 1-state case of top-down or bottom-up tree transducers). Let us also note that the problem of regularity of a set of ground normal forms of a term rewriting system is known to be decidable but with rather intricate proofs [11], while it can be reduced to a particular subclass of the HOM-problem.

We solve the HOM-problem for two new interesting subcases: first, we show decidability in polynomial time for the case where the domain language is constructed over a monadic signature, i.e., over a signature where all symbols have arity 0 or 1. This is obtained by characterizing regularity using a pumping argument, and then reducing the regularity check to certain finiteness tests on images of the homomorphism. Second, decidability is proved for the case where non-linearity of the homomorphism is restricted to the root node of any input term (or, more precisely, to input nodes of bounded depth). The latter result is obtained by proving decidability of the following problem: Given a set of terms with regular constraints on the variables, is its set of ground instances regular? This extends previous results where regular constraints were not considered, see, e.g., [10]. For this case, we give a sufficient condition for non-regularity by the existence of certain infinite solutions of a formula. The formula is of first-order logic with equality predicates and membership constraints in regular languages, altogether interpreted over a term algebra. Our algorithm for this case makes iterated use of the decidability for this sufficient condition, which follows from a result by Comon and Delor [4].

2 Preliminaries

Convention. In this article we use the words "term" and "tree" interchangeably, and the same holds for "position" and "node".

A *signature* consists of an alphabet Σ, i.e., a finite set, together with a mapping that assigns to each symbol in Σ a natural number, its *arity*. We write $\Sigma^{(k)}$ to denote the subset of symbols in Σ that are of arity k, and we write $\sigma^{(k)}$ to denote that σ is a symbol of arity k. The *set of all terms over* Σ is denoted T_Σ and is inductively defined as the smallest set T such that for every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $t_1, \ldots, t_k \in T$, the term $\sigma(t_1, \ldots, t_k)$ is in T. For a term of the form $a()$ we simply write a. For instance, if $\Sigma = \{f^{(2)}, a^{(0)}\}$ then T_Σ is the set of terms that represent all binary trees with internal nodes labeled f and leaves labeled a. We fix the set $X = \{x_1, x_2, \ldots\}$ of variables. The set of trees over Σ with variables in X, denoted $T_\Sigma(X)$, is the set of terms over $\Sigma \cup X$ where every symbol in X has arity zero. Given a tree $\sigma(t_1, \ldots, t_k) \in T_\Sigma$, its set of positions $\mathrm{Pos}(t)$ equals $\{\varepsilon\} \cup_{1 \leq i \leq k} \{i.p \mid p \in \mathrm{Pos}(t_i)\}$. Thus, ε denotes the root node, and $p.i$ denotes the ith child of position p. The subtree of t at position p is denoted by t/p, and the symbol of t at position p is denoted by $t[p]$. For instance, for $s = \sigma(f(a, b), c)$, $s/1$ equals $f(a, b)$ and $s[1]$ equals f. For a signature Γ, we use $\mathrm{Pos}_\Gamma(t)$ to denote the set of positions of t that are labeled by symbols in Γ. E.g., for s of above, $\mathrm{Pos}_{\{c\}}(s) = \{2\}$ and $\mathrm{Pos}_X(s) = \varnothing$. For terms s, t and $p \in \mathtt{Pos}(s)$, we denote by $s[p \leftarrow t]$ the result of replacing the subtree at position p in s by the term t. For instance, $f(f(a, a), a)[1 \leftarrow a] = f(a, a)$.

A (deterministic) bottom-up tree automaton (for short, DTA) is a tuple $A = (Q, Q_a, \Sigma, \delta)$ where Q is a finite set of states, $Q_a \subseteq Q$ is the set of accepting states, Σ is a signature, and $\delta = (\delta_\sigma)_{\sigma \in \Sigma}$ is a collection of transition functions such that for every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, δ_σ is a function from Q^k to Q. The language $L(A)$ recognized by A is the set $\{t \in T_\Sigma \mid A(t) \in Q_a\}$ where A is the extension of δ_σ to trees in T_Σ, recursively defined as: $A(\sigma(t_1, \ldots, t_k)) = \delta_\sigma(A(t_1), \ldots, A(t_k))$ for $\sigma \in \Sigma^{(k)}$, $k \geq 0$,

and $t_1, \ldots, t_k \in T_\Sigma$. We also define, for a state q, the set $L(A, q) = \{t \in T_\Sigma \mid A(t) = q\}$ of trees for which A arrives in state q. A term language $L \subseteq T_\Sigma$ is *regular* if there exists a DTA A such that $L = L(A)$.

Tree Homomorphisms. Let Σ and Δ be signatures. A *homomorphism (from Σ to Δ)* is a mapping H that associates to every symbol $\sigma \in \Sigma$ of arity k a tree in $T_\Delta(\{x_1, \ldots, x_k\})$. The homomorphism H is extended to trees over Σ by defining $H(\sigma(s_1, \ldots, s_k)) = H(\sigma)[x_1 \leftarrow H(s_1), \ldots x_k \leftarrow H(s_k)]$. The term $t[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]$ denotes the *substitution* of every occurrence of x_i in t by the tree t_i. The homomorphism H is *linear* if $H(\sigma)$ is linear in $\{x_1, \ldots, x_k\}$ for each $\sigma \in \Sigma$ of arity k, i.e., if each x_i, $1 \leq i \leq k$ occurs at most once in $H(\sigma)$. We will later make use of the following known result, cf. Corollary 3.10 of [5].

Proposition 1. *Let H be a linear homomorphism and L a regular tree language. Then $H(L)$ is effectively regular.*

Sets of Terms with Regular Constraints. Let $V = \{x_1, \ldots, x_n\}$. A regular *constraint* for variables $x_1, \ldots, x_n$ maps each x_i in V to a regular tree language. Let Γ be a signature. A *solution* of C (over Γ) is a ground substitution $\varphi : V \to T_\Gamma$ such that $\varphi(x_i) \in C(x_i)$ for all $i \in \{1, \ldots, n\}$. A set of terms with regular constraints is a pair $\langle S, C\rangle$ where S is a set of terms (with variables) and C is a constraint defined at least for the variables occurring in S. The language of $\langle S, C\rangle$, denoted as $L(\langle S, C\rangle)$, is defined as $\{t \mid \exists \varphi, s : (t = \varphi(s) \wedge s \in S \wedge \varphi \text{ is a solution of } C)\}$. The following result is due to [12], cf. also [10].

Proposition 2. *Let $\mathcal{C}$ be the trivial constraint mapping every variable to the set of all terms. Then, regularity of $L(\langle S, \mathcal{C}\rangle)$ is decidable for given S.*

3 The Monadic Case

We consider the particular case where the regular input term language is constructed over a *monadic* signature with only unary symbols and constants and prove that regularity of the image by a homomorphism is decidable in polynomial time.

Let Σ be a monadic signature, i.e., $\Sigma = \Sigma^{(1)} \cup \Sigma^{(0)}$. Let L be a regular term language over Σ, and H be a homomorphism. For simplicity we assume that the signature for the domain language L contains just one constant c, and moreover, $H(c) = c$. If this was not the case we can easily transform L and H into new L' and H' such that L' satisfies this requirement and $H'(L') = H(L)$: we simply introduce a new constant c not present in Σ and define L' as $H_{\mathtt{lin}}(L)$ where $H_{\mathtt{lin}}$ is the linear homomorphism with $H_{\mathtt{lin}}(f) = f(x_1)$ for all unary functions f, and $H_{\mathtt{lin}}(e) = e(c)$ for constants e. We define H' as $H'(f) = H(f)$ for unary function symbols f, $H'(e) = H(e)$ for constants e of the original signature of L, and $H'(c) = c$. Note that constants e of the original signature of L are now unary function symbols of the signature of L'.

For the current case the language L is essentially a language on words. We use then expressions like $abbaa(c)$ or just $abbaa$ for denoting $a(b(b(a(a(c)))))$. Note that $\Sigma = \Sigma^{(1)} \cup \{c^{(0)}\}$.

A unary symbol a is called *erasing* if $H(a(x)) = x$, and by $\lfloor w \rfloor$ we denote the number of positions of w with non-erasing symbols. Intuitively, we are counting the number of symbols that generate at least one output node in the image; this means that the size of the term $H(w)$ is larger than or equal to $\lfloor w \rfloor$.

Definition 3. *Let Σ be a monadic signature, H a homomorphism over Σ, and $w \in T_\Sigma$. The* level *of a position p with respect to w and H, denoted by level(p, w, H), is $k+1$ if there exists a factoring of the form $w = w_1 a w_2$ with $\lfloor w_1 \rfloor = k$ such that $p \notin \mathtt{Pos}(H(w_1))$ but $p \in \mathtt{Pos}(H(w_1 a))$.*

Intuitively, $\mathtt{level}(p, w, H) = k$ means that p in $H(w)$ was generated by the k-th non-erasing symbol in the word w. For instance, if $H(g) = f(x_1, x_1)$ and $H(c) = c$ then $\mathtt{level}(\varepsilon, ggg(c), H) = 1$ because $ggg = w_1 g w_2$ with w_1 the empty word, $\varepsilon \notin \mathtt{Pos}(H(w_1)) = \emptyset$, and $\varepsilon \in \mathtt{Pos}(H(g))$. If H is also defined as $H(d) = x_1$, then $\mathtt{level}(1.2, gdgg(c), H) = 3$ because the node 1.2 in the term $H(gdgg(c)) = f(f(f(c,c), f(c,c)), f(f(c,c), f(c,c)))$ was generated by the third non-erasing symbol of $gdgg(c)$, i.e., by the last g. The following lemma is straightforward from this definition.

Lemma 4. *Let p, q be positions such that q is a prefix of p. If $\mathtt{level}(p, w, H) \geq k + \mathtt{level}(q, w, H)$, then $|p| \geq k + |q|$.*

Definition 5. *A position p is* live *in $\langle w, H \rangle$ if there exists p' such that $\mathtt{level}(p.p', w, H) > \mathtt{level}(p, w, H)$.*

We define $h := \mathtt{Max}_{a \in \Sigma^{(1)}}(\mathtt{height}(H(a)))$. The following two lemmas are straightforward from the previous definitions.

Lemma 6. *Let p be a position and w a word such that $\mathtt{height}(H(w)/p) > h$. Then, p is live in $\langle w, H \rangle$.*

Lemma 7. *Let p and q be positions and w a word such that, p and q are live in $\langle w, H \rangle$, and $\mathtt{level}(p, w, H) = k + \mathtt{level}(q, w, H)$. Then, $|\mathtt{height}(H(w)/p) - \mathtt{height}(H(w)/q)| \leq h \cdot (k+1)$.*

The following lemma is essential for doing a pumping argument in the lemma after, which characterizes the regularity of the image of a homomorphism for the monadic case.

Lemma 8. *Let t be a term in $H(L)$, let $p, p.q_1, p.q_2$ be positions in $\mathtt{Pos}(t)$ such that $|q_1|, |q_2| \leq h$ and $\mathtt{height}(t/p.q_1), \mathtt{height}(t/p.q_2) > h$. Then $|\mathtt{height}(t/p.q_1) - \mathtt{height}\ (t/p.q_2)| \leq h(h+1)$.*

Proof. By our assumptions, $|p.q_1| - |p| \leq h$ and $|p.q_2| - |p| \leq h$. By Lemma 4, $\mathtt{level}\ (p.q_1, w, H) = k_1 + \mathtt{level}(p, w, H)$ and $\mathtt{level}(p.q_2, w, H) = k_2 + \mathtt{level}(p, w, H)$ for some $k_1, k_2 \leq h$. Therefore, either $\mathtt{level}(p.q_1, w, H) = k + \mathtt{level}(p.q_2, w, H)$ or $\mathtt{level}(p.q_2, w, H) = k + \mathtt{level}(p.q_1, w, H)$ for some $k \leq h$. By Lemma 6, $p.q_1$ and $p.q_2$ are live in $\langle w, H \rangle$. Finally, by Lemma 7, $|\mathtt{height}(H(w)/p.q_1) - \mathtt{height}\ (H(w)/p.q_2)| \leq h(k+1) \leq h(h+1)$. □

Definition 9. *We say that a symbol $a \in \Sigma^{(1)}$ is* deleting *in H, or that H is deleting on a, if $H(a)$ does not contain x_1. A word w is deleting (or "H is deleting on w") if it contains a deleting symbol. Let $a \in \Sigma^{(1)}$. We define $L_{a\_} := \{w \mid \exists u : (uaw \in L \wedge u \text{ is non-deleting})\}$.*

We are now ready to prove the main result of this section. It characterizes regularity of the homomorphic image of a monadic tree language. A symbol $a \in \Sigma^{(1)}$ is *copying in* H, or H is *copying on* a, if $H(a)$ contains at least two occurrences x_1. A word w is *copying* (or "H is copying on w") if it contains a copying symbol.

Lemma 10. *$H(L)$ is not regular iff there exists $a \in \Sigma^{(1)}$ such that H is copying on a and $H(L_{a\_})$ is infinite.*

Proof. We first prove the *if*-direction. Let $a \in \Sigma^{(1)}$ such that H is copying on a and $H(L_{a\_})$ is infinite. We assume that $H(L)$ is regular in order to reach a contradiction. Then, there exists a bottom-up tree automaton A recognizing $H(L)$. Recall from the Preliminaries that $A(t)$ denotes the state in which A arrives after processing the term t. From $L_{a\_}$ we choose two words w_1 and w_2 such that $\mathtt{height}(H(w_1)), \mathtt{height}(H(w_2)) > h$, $|\mathtt{height}(H(w_1)) - \mathtt{height}(H(w_2))| > h(h+1)$, and $A(H(w_1)) = A(H(w_2))$; this is possible by non-finiteness of $H(L_{a\_})$. Now, since w_1 belongs to $L_{a\_}$, there exists a word of the form uaw_1 in L where u is non-deleting. Therefore, since a is copying, $H(uaw_1)$ is of the form $s[p \leftarrow t'[q_1 \leftarrow H(w_1)][q_2 \leftarrow H(w_1)]]$ for positions p, q_1, q_2 satisfying $|q_1|, |q_2| \leq h$ and q_1 is "disjoint" from q_2 (q_1 is not a prefix of q_2, and q_2 is not a prefix of q_1). Recall from the Preliminaries that $u[q \leftarrow v]$ denotes the result of replacing position q in the term u by the term v. The term $t = s[p \leftarrow t'[q_1 \leftarrow H(w_1)][q_2 \leftarrow H(w_2)]]$ is also in $H(L)$ because $A(H(w_1)) = A(H(w_2))$. By our previous conditions we have $\mathtt{height}(t/p.q_1), \mathtt{height}(t/p.q_2) > h$, and $|\mathtt{height}(t/p.q_1) - \mathtt{height}(t/p.q_2)| > h(h+1)$. This is a contradiction to Lemma 8.

Next, we prove the *only-if*-direction. Assume that for all $a \in \Sigma^{(1)}$ such that H is copying on a, $H(L_{a\_})$ is finite. We define $L_a = aL_{a\_}$, i.e., $L_a = \{aw \mid \exists u : (uaw \in L \text{ and } u \text{ is non-deleting})\}$, for every $a \in \Sigma^{(1)}$, and also define $T = \{t \mid \exists a \in \Sigma^{(1)} : (H \text{ is copying on } a \wedge t \in H(L_a))\}$, which is a finite set. Further, we define a new set of constants $C = \{c_t \mid t \in T\}$.

Our goal is to define an alternative language L' and a homomorphism H' satisfying $H'(L') = H(L)$, and for which it is easy to see that $H'(L')$ is regular. We first define some particular subsets of L'.

$$
\begin{aligned}
L_{\mathtt{cop}} &= \{uc_t \mid H \text{ is neither copying nor deleting on } u \wedge \\
&\qquad \exists a, w : ((H \text{ is copying on } a) \wedge uaw \in L \ \wedge \ H(aw) = t)\} \\
L_{\mathtt{del}} &= \{ua \mid H \text{ is neither copying nor deleting on } u \wedge \\
&\qquad H \text{ is deleting on } a \ \wedge \ \exists w : (uaw \in L)\} \\
L_{\mathtt{oth}} &= \{u \mid H \text{ is neither copying nor deleting on } u \ \wedge \ u \in L\}
\end{aligned}
$$

Finally we define $L' = L_{\mathtt{cop}} \cup L_{\mathtt{del}} \cup L_{\mathtt{oth}}$, and H' is defined like H for the symbols $a \in \Sigma$ that are not copying, H' is undefined for the $a \in \Sigma^{(1)}$ that are copying, and $H'(c_t) = t$ for every $t \in T$. Note that L' does not have words containing symbols such that H is copying on them.

We show first that $H'(L') = H(L)$. The inclusion $\subseteq$ is straightforward from the definition of L' and H'. For $\supseteq$ let v be a word of L. Either v does not contain any copying or deleting symbol, or it is of the form uaw where u does not contain any copying or deleting symbols, and a is either copying or deleting. Depending on the case, it is easily seen that either $v \in H'(L_{\texttt{oth}})$ or $v \in H'(L_{\texttt{cop}})$ or $v \in H'(L_{\texttt{del}})$.

Second, we see that L' is regular. From an automaton A recognizing L we can easily define an automata for $L_{\texttt{cop}}$, $L_{\texttt{del}}$ and $L_{\texttt{oth}}$, respectively. Without lost of generality we assume that all states of A can reach an accepting state. For the case of $L_{\texttt{oth}}$ it suffices to remove from A the transitions with copying and deleting symbols. For the case of $L_{\texttt{del}}$ it suffices to remove the transitions with copying symbols and to redirect all the transitions with deleting symbols to a new accepting state. For the case of $L_{\texttt{cop}}$ we have to remove all transitions of deleting symbols and all transitions of constant symbols. We also need to consider the transitions with a copying symbol a from a state q to a state q', such that q is reachable and q' can reach an accepting state through non-deleting and non-copying symbols. Every of these transitions has to be replaced by several new transitions to the state q' one for every constant c_t such that $t \in H(a\ L(A, q'))$. Finally, from the regularity of L' and the fact that H' is linear we conclude by Proposition 1 that $H'(L') = H(L)$ is regular. □

Before we use Lemma 10 to show decidability of regularity of $H(L)$ for monadic L, let us give an example application of the only-if direction of the previous lemma.

Example 11. Let $\Sigma = \{a^{(1)}, b^{(1)}, d^{(1)}, c^{(0)}\}$ and, for $n \geq 0$, $L_n = \{d(a|b)^k(c) \mid k \leq n\}$. The language L_n is recognized by the automaton $A_n = (\{q_0, q_1, \ldots, q_n, q_f\}, \{q_f\}, \Sigma, \delta)$ where $\delta_c() = q_0$, $\delta_a(q_0) = \delta_b(q_0) = q_1$, $\delta_a(q_1) = \delta_b(q_1) = q_2$, …, $\delta_a(q_{n-1}) = \delta_b(q_{n-1}) = q_n$, and $\delta_d(q_i) = q_f$ for all $0 \leq i \leq n$. The tree homomorphism H is defined by $H(a) = a(x_1)$, $H(b) = b(x_1)$, $H(d) = f(x_1, x_1)$, and $H(c) = c$. Thus, $H(L_n) = \{f(w, w) \mid w \in (a|b)^k c, k \leq n\}$. Since $H(L_n)$ is finite, it is obviously regular. Let us now follow the only-if direction of the proof of Lemma 10 in order to construct a tree automaton B_n which recognizes $H(L_n)$. Note that $L_{d\_} = \{w \in (a|b)^k c \mid k \leq n\}$. The set T defined in the proof equals $H(\{dw \mid w \in L_{d\_}\}) = \{f(w, w) \mid w \in L_{d\_}\}$. Thus, for every possible $w \in \{a, b\}^*$ of length at most n, the set T contains the tree $t = f(w, w)$, and accordingly, C contains the constant c_t. For instance, if $n = 2$, then $C = \{c_{f(c,c)}, c_{f(a(c),a(c))}, c_{f(b(c),b(c))}, c_{f(aa(c),aa(c))}, c_{f(ab(c),ab(c))}, c_{f(ba(c),ba(c))}, c_{f(bb(c),bb(c))}\}$.

Now, to define $L_{\texttt{oth}}$ we simply remove from A_n all d-transitions because d is copying. Since the resulting automaton has no transitions to accepting states, its language is empty, i.e., $L_{\texttt{oth}} = \varnothing$. Also $L_{\texttt{del}} = \varnothing$ because d-transition are removed from A_n and no new transitions are added because there are no deleting symbols. The automaton for $L_{\texttt{cop}}$ is obtained from A_n by deleting the transition $\delta_c() = q_0$ and by replacing each transition $\delta_d(q_i) = q_f$, $0 \leq i \leq n$ by the new transitions $\delta_{c_t}() = q_f$ for each $c_t \in C$. Thus, the resulting automaton B_n recognizes precisely the set C. Since for every $t \in H(L_n)$ the constant c_t is in C, and the new homomorphism H' has $H'(c_t) = t$ for all $c_t \in C$, we obtain that $H'(L(B_n)) = H'(C) = H(L_n)$. □

It is interesting to observe that a tree automaton that recognizes the language $H(L_n)$ of Example 11, is bound to be of size at least $O(2^n)$. This is easy to see, because for

every possible subtree s of $f(s,s) \in H(L_n)$ we need one extra state. In fact, even if we consider *nondeterministic* bottom-up tree automata, which are obtained by generalizing the transition functions $\delta_\sigma : Q^k \to Q$ to functions $\delta_\sigma : Q^k \to \mathcal{P}(Q)$, then any smallest automaton for $H(L_n)$ is still of size $O(2^n)$. Thus, through copying of a finite set of terms (recognized by a DTA A_n with $O(n)$ states) we obtain a finite set of terms for which any tree automaton needs exponentially more states than A_n. In other words, the description of $H(L_n)$ through the homomorphism H and the automaton A_n is *exponentially more succinct* than the description through any (even nondeterministic) tree automaton for $H(L_n)$. We obtain the following proposition. Let the size of a homomorphism be defined as the sum of sizes of the trees $H(g)$ for which H is defined, and the size of a term t equals the cardinality of $\mathtt{Pos}(t)$.

Proposition 12. *For every n, there exists a homomorphism H and a DTA A_n such that (1) H is of size $O(n)$ (2) A_n has $O(n)$-many states, and (3) $H(L(A_n))$ cannot be recognized by any nondeterministic tree automaton with less than 2^n states.*

Note that the language $H(L_n)$ of Example 11 cannot be recognized by a deterministic top-down tree automaton. We are ready to state the main result of this section.

Theorem 13. *Regularity of the homomorphic image of a regular term language L is decidable in polynomial time if L is defined over a unary alphabet.*

Proof. By Lemma 10 it suffices to prove that the infiniteness of $H(L_{a\_})$ can be checked in polynomial time, for a given tree automaton A recognizing the regular language L, a homomorphism H, and copy symbol a. To this end, we just have to look for a transitions $\delta_a(q_1) = q_2$ and $\delta_b(q_3) = q_4$ such that q_1 is reachable from the initial state using no deleting symbols, b is not deleting nor erasing, q_3 is reachable from q_2 without deleting symbols, and q_3 is reachable from q_4 without deleting symbols. The statement of the theorem follows from the fact that all these checks can be done in polynomial time with usual algorithms for finding paths. □

4 Sets of Terms with Regular Constraints and Bounded-Depth Copying Homomorphisms

In this section we prove decidability of regularity for the set of ground instances of a given finite set of terms. This is done in three steps. In Section 4.1 we give a sufficient condition for non-regularity by the existence of a certain infinite set of instances. In Section 4.2 we prove decidability of such existence as a direct consequence of a result from [4] and provide an algorithm for this problem. We also define in Section 4.3 a class of pairs (L, H), where L is a language and H is a homomorphism, called *bounded-depth copying*, and prove that regularity of $H(L)$ is decidable.

4.1 A Sufficient Condition for Non-regularity

Some technical definitions are needed in order to prove our condition in Lemma 14. We say that a term s is *determined* on a position p if either p is a position of s and

is labeled by a non-variable symbol, i.e., $s[p] \notin X$, or there exists a prefix p' of p such that s/p' is a non-variable symbol, i.e., $s/p' = s[p'] \notin X$. Equivalently, s is determined on p if for any two substitutions φ_1, φ_2, either p is not defined in both $\varphi_1(s)$ and $\varphi_2(s)$, or p is defined in both $\varphi_1(s)$ and $\varphi_2(s)$ and $\varphi_1(s)[p] = \varphi_2(s)[p]$. We say that s is determined on a set of positions P if s is determined on all positions p in P. Recall from the Preliminaries that a set of terms with regular constraints is a pair $\langle S, C\rangle$ where C maps each variable occurring in S to a regular term language, and that $L(\langle S, C\rangle) = \{\varphi(s) \mid s \in S, \varphi \text{ is a solution of } C\}$.

Lemma 14. *Let s be a term, S a set of terms, and C a constraint for the variables occurring in s and S. Let x be a variable that occurs twice in s. Let $\{\varphi_1, \varphi_2, \ldots\}$ be an infinite set of solutions of C satisfying $\varphi_i(s) \notin L(\langle S, C\rangle)$ for all $i > 0$, and $\varphi_i(x) \neq \varphi_j(x)$ for all $1 \leq i < j$. Then $L(\langle\{s\} \cup S, C\rangle)$ is not regular.*

Proof. Without lost of generality we may suppose that the set of variables occurring in s is disjoint from the set of variables occurring in S; if it is not the case we simply introduce a new variable for each variable that occurs in s and S. Let $S = \{s_1, \ldots, s_n\}$. We may make the following assumption. For a term t, let $\mathtt{Pos}_{\mathtt{nv}}(t)$ denote the positions p in $\mathtt{Pos}(t)$ that are not labeled by a variable, i.e., for which $t[p] \notin X$.

Assumption 1: s is determined on $\bigcup_{i\in\{1,\ldots,n\}} \mathtt{Pos}_{\mathtt{nv}}(s_i)$.

If the assumption is not satisfied, we proceed as follows. Since s is not determined on $\bigcup_{i\in\{1,\ldots,n\}} \mathtt{Pos}_{\mathtt{nv}}(s_i)$, there exists a position $p \in \mathtt{Pos}(s)$ such that s/p is a certain variable y and $p \in \bigcup_{i\in\{1,\ldots,n\}} \mathtt{Pos}_{\mathtt{nv}}(s_i)$. Let A be a deterministic bottom-up tree automaton that recognizes the regular term language $C(y)$. The automaton A has a finite set of transitions of the form $\delta_g(q_1, \ldots, q_m) \to q$ for states $q_1, \ldots, q_m, q$ of A where q is an accepting state. For every of these rules $\delta_g(q_1, \ldots, q_m) \to q$ we construct a substitution $\gamma_{g,q_1,\ldots,q_m,q}$ that is the identity for all variables except for y, for which it is defined as $\gamma_{g,q_1,\ldots,q_m,q}(y) = g(z_1, \ldots, z_m)$, for new variables $z_1, \ldots, z_m$. We also extend the constraint C to a new constraint C' by defining $C'(z_i) = L(A, q_i)$. We do this for all transitions, choosing new variables for every transition. Let $s'_1, \ldots, s'_l$ be the terms obtained by applying all the substitutions $\gamma_{g,q_1,\ldots,q_m,q}$ to s. Note that $L(\langle\{s'_1, \ldots, s'_l\} \cup S, C'\rangle)$ coincides with $L(\langle\{s\}\cup S, C\rangle)$, and that the languages $L(\langle s'_1, C'\rangle), \ldots, L(\langle s'_l, C'\rangle)$ are disjoint, since A is deterministic. By finiteness, for one of the terms $s'_1, \ldots, s'_l$, say s'_1, there exists an infinite subset $\{\varphi_{i_1}, \varphi_{i_2}, \ldots\}$ of $\{\varphi_1, \varphi_2, \ldots\}$ such that the run of A on each $\varphi_{i_j}(y)$ applies the same transition $\delta_g(q_1, \ldots, q_m) = q$ at the root node. Let k be the arity of this g, and note that in the case where y equals x, k is greater than 0 due to the condition $\varphi_i(x) \neq \varphi_j(x)$ for all $1 \leq i < j$. There exist substitutions $\theta_{i_1}, \theta_{i_2}, \ldots$ such that every φ_{i_j} coincides with the corresponding $\theta_{i_j} \circ \gamma_{g,q_1,\ldots,q_m,q}$. Moreover, in the case where y equals x, there exists k' between 1 and k and an infinite subset Θ of $\{\theta_{i_1}, \theta_{i_2}, \ldots\}$ such that for all different $\theta, \theta' \in \Theta$, $\theta(z_{k'}) \neq \theta'(z_{k'})$, where $z_1, \ldots, z_k$ are the new variables in $\gamma_{q,q_1,\ldots,q_m,q}(y)$.

The term s'_1, the set of terms $\{s'_2, \ldots, s'_l\} \cup S$, the set of substitutions Θ or $\{\theta_{i_1}, \theta_{i_2}, \ldots\}$, deppending on whether x equals y or not, the constraint C', and the variable $z_{k'}$ if y was x, or the variable x otherwise, satisfy the conditions of the lemma, but also, p is determined on s'_1. Hence, since $L(\langle\{s'_1, \ldots, s'_l\} \cup S, C'\rangle)$ coincides with $L(\langle\{s\} \cup$

$S, C\rangle)$, the statement of the lemma remains unchanged when replacing the elements by these new ones, for which there is strictly one less position that is not determined. We can repeat this process if there are still positions in $\bigcup_{i\in\{1,\ldots,n\}} \mathtt{Pos}_{\mathtt{nv}}(s_i)$ that s'_1 is not determined on them, until Assumption 1 on determined positions is satisfied.

Now, we come back to the principal proof with this additional assumption. We prove that $L(\langle\{s\} \cup S, C\rangle)$ is not regular by contradiction. Hence, suppose that B is a deterministic tree automaton recognizing this language. We also assume an implicit election of an automaton $A_{C(y)}$ recognizing $C(y)$, for every variable $y \in \mathtt{Dom}(C)$. At this point we need an additional assumption.

Assumption 2: For every position $p \in \mathtt{Pos}(s)$, the run of B gives the same state on all terms $\varphi_i(s)/p$. Moreover, for every of such $p \in \mathtt{Pos}(s)$ and $A_{C(y)}$, the run of $A_{C(y)}$ gives the same state on all terms $\varphi_i(s)/p$.

Similar as before, Assumption 2 can easily be enforced by choosing a certain infinite subset Γ of the substitutions $\varphi_1, \varphi_2, \ldots$

Now, note that the fact that a certain term $\varphi_i(s)$ is not in $L(\langle\{s_j\}, C\rangle)$ for any $s_j \in S$ can be due to several different causes:

(i) There is a position $p \in \mathtt{Pos}_{\mathtt{nv}}(s_j)$ such that either p is not defined in s or it is defined in s but then $s[p] \neq s_j[p]$.
(ii) Item (i) is not satisfied, but there exists a position p such that s_j/p is a variable and $\varphi_i(s)/p \notin C(s_j/p)$.
(iii) Items (i) and (ii) are not satisfied, but there exist positions p_1, p_2 such that s_j/p_1 and s_j/p_2 are the same variable, but $\varphi_i(s)/p_1 \neq \varphi_i(s)/p_2$.

We say that either (i), or (ii), or (iii) with the corresponding chosen p_1 and p_2, are *the cause* for $\varphi_i \in \Gamma$ and $s_j \in S$, depending on the case.

Finally, we fix a position q such that $s/q = x$ and a substitution $\varphi \in \Gamma$.

Now, consider φ and any other substitution $\varphi' \in \Gamma$ such that $\varphi'(x)$ is not a subterm of $\varphi(s)$. Note that such a φ' exists thanks to the fact that all the $\varphi_i(x)$ are different. The term $\varphi(s)[q \leftarrow \varphi'(x)]$ is accepted by B due to Assumption 2. Hence, to reach a contradiction it suffices to see that it is not in $L(\langle\{s\} \cup S, C\rangle)$, i.e. the language accepted by B. Of course it is not in $L(\langle\{s\}, C\rangle)$, since x appears multiple times in s, and in particular at position q and in some other position q' satisfying $\varphi(s)[q \leftarrow \varphi'(x)]/q \neq \varphi(s)[q \leftarrow \varphi'(x)]/q'$. But also, $\varphi(s)[q \leftarrow \varphi'(x)]$ is not in any $L(\langle\{s_j\}, C\rangle)$ for $s_j \in S$, and we prove it deppending on which is the cause for φ and s_j. If the cause for φ and s_j is (i), this is trivial. If the cause for φ and s_j is (ii), this follows directly from Assumption 2. If the cause for φ and s_j is (iii) with positions p_1 and p_2, then, either q is disjoint with p_1 and p_2 and this is trivial, or otherwise, q has to be a suffix of either p_1 or p_2, due to Assumption 1. In the latter case, due to the election of φ', $\varphi(s)[q \leftarrow \varphi'(x)]/p_1 \neq \varphi(s)[q \leftarrow \varphi'(x)]/p_2$, and hence, $\varphi(s)[q \leftarrow \varphi'(x)]$ is not in $L(\langle\{s_j\}, C\rangle)$ again. □

4.2 Computing the Existence of Infinite Instances

Due to our sufficient condition, and the use of it that is done later by the algorithm for deciding regularity, we need to prove that the following problem is computable:

Infinite-Instances Problem
Input: $x, s, \{s_1, ..., s_n\}, C$.
Output: If there exists an infinite set $\varphi_1, \varphi_2, \ldots$ of solutions of C such that $\varphi_i(x) \neq \varphi_j(x)$, and $\varphi_i(s)$ is not in $L(\langle\{s_1, \ldots, s_n\}, C\rangle)$, then give output "yes". Otherwise, give as output the finite set $\{t_1, \ldots, t_k\}$ of ground terms t_i satisfying: $t_i \neq t_j$ for $1 \leq i < j \leq k$, t_i is ground for $1 \leq i \leq k$, and for any solution φ such that $\varphi(s)$ is not in $L(\langle\{s_1, \ldots, s_n\}, C\rangle)$, it holds that $\varphi(x) = t_i$ for some i in $\{1, \ldots, k\}$.

The decidability of this problem is *directly deduced* from the work of Comon and Delor [4] on equational formulae with membership constraints [1]. An *equational formula with membership constraints* is a first order formula whose atoms are equations $t = u$, membership constraints $t \in S$ (S is called a sort symbol) or $\bot$. Formulae are interpreted w.r.t. to a mapping $\mathcal{U}$ which associates with each sort symbol S a subset of T_Σ. A ground substitution σ is a solution of the equation $s = t$ if its domain contains the set of free variables of $\{s, t\}$ and $t\sigma \equiv s\sigma$, where $\equiv$ denotes the syntactic equality in terms. This is a solution of $t \in S$ if $t\sigma$ belongs $\mathcal{U}(S)$. The connectives are interpreted as usual and with a formula we associate the set of solutions. Two formulae are said equivalent w.r.t. $\mathcal{U}$ if they have the same solutions w.r.t. $\mathcal{U}$. From now on, we will suppose that the interpretation of the sort symbols is fixed and that sorts are interpreted as recognizable languages. In [4], the authors provide a complete, correct and terminating set of rules, which reduces a formula to an equivalent solved form which is either $\bot$, or a finite disjunction of formulae (called definition with constraints) of the form:
$\exists \boldsymbol{w} : x_1 = t_1 \wedge ... \wedge x_n = t_n \wedge z_1 \neq u_1 \wedge z_m \neq u_m \wedge y_1 \in S_1 ... \wedge y_k \in S_k$ where:

- $x_1, \ldots, x_n$ are free variables that occur only once in the formula
- $z_1, \ldots, z_m$ are variables, s.t. for all i, $z_i \notin Var(u_i)$
- $y_1, \ldots, y_k$ are distinct variables
- for all i, S_i is interpreted as an infinite (recognizable) language.
- $t_1, \ldots, t_n, u_1, \ldots, u_m$ are -non necessarly ground- terms.

Furthermore, Comon and Delor prove that every definition with constraints has at least one solution (Lemma 2 of [4]). Using those results we can even get:

Lemma 15. *Let $D = \exists \boldsymbol{w} : x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge z_1 \neq u_1 \wedge \ldots \wedge z_m \neq u_m \wedge y_1 \in S_1 \wedge \ldots \wedge y_k \in S_k$ be a definition with constraints satisfying the above conditions. Let x a free variable in D. D has an infinite set of solutions satisfying $\sigma_i(x) \neq \sigma_j(x)$ for all $j > i > 0$ iff it does not contain any equation $x = t$ with t ground.*

Proof. Of course, if D contains an equation $x = t$ with t ground, D has no infinite set of solutions satisfying $\sigma_i(x) \neq \sigma_j(x)$.

Conversely, let us suppose that D does not contain any equation $x = t$ with t ground. We can suppose without lost of generality that $\{y_1, \ldots, y_k\}$ contains all the variables except the x_i's (we can add $v \in \top$ if necessary). By Lemma 2 of [4], D has one solution σ_1. Now consider a new formula D' obtained by replacing every S_i by $S_i - \{\sigma_1(y_i)\}$. This formula is again a definition with constraints (the new sets are regular and infinite).

[1] Thanks to Hubert Comon who made us aware of this fact.

For D', Lemma 2 of [4] applies again and we have a solution σ_2 for D', which in fact is also a solution of D. Moreover, $\sigma_1(x)$ differs from $\sigma_2(x)$ due to the definition of σ_2, and the fact that either x is some y_i, or x is some x_i and the corresponding t_i is not ground and with all its variables in $\{y_1, \ldots, y_k\}$. This process can be repeated again by replacing every S_i by $S_i - \{\sigma_1(y_i), \sigma_2(y_i)\}$, and so on, thus obtaining the desired sequence. □

Now we come back to the Infinite-Instances problem. Let $x, s, S = \{s_1, ..., s_n\}, C$ be an input of the problem. For this input we define Φ as the following equational formula with membership constraints:

$$(\wedge_{y \in Var(s)} y \in C(y)) \wedge$$
$$(\wedge_{1 \leq i \leq n, \{x_1, \ldots, x_k\} = Var(s_i)} (\forall x_1, \ldots, x_k : (x_1 \in C(x_1) \wedge \ldots \wedge x_k \in C(x_k) \Rightarrow s_i \neq s)))$$

Let $Sol(\Phi)$ the set of solutions of Φ. There exists an infinite set $\varphi_1, \varphi_2, \ldots$ such that $\varphi_i(x) \neq \varphi_j(x)$, and $\varphi_i(s)$ is not in $L(\langle\{s_1, \ldots, s_n\}, C\rangle)$, iff there exists an infinite set $\varphi_1, ..., \varphi_j ..$ in $Sol(\Phi)$ such that $\varphi_i(x) \neq \varphi_j(x), 0 < i < j$.

But now, as said before, we can transform Φ into an equivalent finite disjunction of definitions with constraints $\vee_{1 \leq i \leq k} D_i$. For each of these definitions D_i, we check whether it contains an equation $x = t_i$, with t_i ground. If the answer is no for at least one element of the disjunction, there exists an infinite set $\{\varphi_1, \varphi_2, \ldots\}$ included in $Sol(\Phi)$ satisfying that $\varphi_i(x) \neq \varphi_j(x)$ for all $j > i > 0$ and we output yes. If the answer is yes for each definition D_i, we output "no" and the set $\{t_1, \ldots, t_k\}$ fulfills the required conditions.

Lemma 16. *The Infinite-Instances problem is computable.*

Deciding Regularity of the Instances of a Set of Terms with Regular Constraints Using Lemmas 14 and 16 we obtain an algorithm for deciding whether $L(\langle S, C\rangle)$ is regular for given S and C. It checks repeatedly the previous sufficient condition looking for non-regularity. After a finite number of steps it terminates with an equivalent linear set, thus concluding regularity.

1 Input: A finite set of terms S and a constraint C.
2 If all terms in S are linear then stop with answer "yes".
3 Otherwise, chose a term $s \in S$ with a certain variable x occurring at least twice in s, and ask for the existence of an infinite set $\{\varphi_1, \varphi_2, \ldots\}$ of solutions of C such that $\varphi_i(x) \neq \varphi_j(x)$, and $\varphi_i(s)$ is not in $L(\langle\{S - \{s\}, C\rangle)$.
4 If the answer is "yes" then stop with answer "no".
5 Otherwise consider the set of ground terms $\{t_1, \ldots, t_k\}$ satisfying $t_i \neq t_j$ for $1 \leq i < j \leq k$, t_i is ground for $1 \leq i \leq k$, and for any solution φ of C such that $\varphi(s)$ is not in $L(\langle\{S - \{s\}, C\rangle)$, it holds that $\varphi(x) = t_i$ for some i in $\{1, \ldots, k\}$, and do: $S := (S - \{s\}) \cup \{\{x \mapsto t_1\}(s), \ldots, \{x \mapsto t_k\}(s)\}$.
6 Go to 2.

Theorem 17. *It is decidable whether $L(\langle S, C\rangle)$ is regular for given set of terms S and constraint C.*

4.3 Deciding the HOM-Problem for Bounded-Depth Copying Homomorphisms

We now come back to the problem of deciding regularity of homomorphic images of regular tree languages, i.e., deciding whether $H(L)$ is regular for a given tree homomorphism H and regular tree language L. Let L be an arbitrary regular tree language. We show that if H is "bounded-depth copying", then we can construct a set of terms with constraints $\langle S, C\rangle$ such that $L(\langle S, C\rangle) = H(L)$. Since for such sets we can decide regularity by Theorem 17, we obtain decidability of regularity of $H(L)$. Roughly speaking, "bounded-depth copying" means that symbols σ for which $H(\sigma)$ is non-linear ("copying"), occur at bounded depth. Let us first fix some notations. For a homomorphism H, denote by $\mathtt{Copy}(H)$ the set of symbols σ for which $H(\sigma)$ is non-linear, i.e., for which a variable x_i occurs more than once in $H(\sigma)$. A symbol σ is called *erasing* if $H(\sigma)$ is a variable, i.e., $H(\sigma) \in X$. A position p of t is *deleted in* t *by* H if there exists q, r and an $i \geq 1$ such that $p = q.i.r$ and $t[q]$ is a symbol b such that $H(b)$ does not contain the variable x_i.

Definition 18. *Let L be a set of trees, H a tree homomorphism, and k a natural number. The pair (L, H) is* depth-k copying *if for any t in L and position p in t such that $t[p]$ belongs to $\mathtt{Copy}(H)$, either p is deleted in t by H or the path from the root of t to p contains at most k occurrences of non-erasing symbols. The pair (L, H) is* bounded-depth copying *if there exists a k such that (L, H) is depth-k copying.*

Note that, given (L, H), it is decidable whether or not (L, H) is bounded-depth copying: we can construct a finite state word automaton A that recognizes the labeled paths of trees in L which lead to a (non-deleted) copy symbol in $\mathtt{Copy}(H)$. Then (L, H) is bounded-depth copying if and only if we can bound the number of non-erasing symbols occurring in words in $L(A)$, which is decidable. Note that this is similar to computing the inverse image of a regular tree language under a homomorphism, used at the end of the proof of Lemma 20; cf. also the discussion in the Conclusions. Clearly, if symbols of $\mathtt{Copy}(H)$ only occur at deleted positions in trees of L, then $H(L)$ is regular. The reason for this is that only linear rules of H are applied when translating trees in L.

Lemma 19. *Let L recognizable and H a homomorphism such that any position in a tree $t \in L$ labeled by a symbol in $\mathtt{Copy}(H)$ is deleted in t by H. Then $H(R)$ is regular.*

Lemma 20. *Let L be recognizable and H a tree homomorphism such that (L, H) is bounded-depth copying. There effectively exists a finite set of terms with regular constraints $\langle S, C\rangle$ such that $L(\langle S, C\rangle) = H(L)$.*

Proof. Let $A = (Q, Q_a, \Sigma, \delta)$ be a deterministic tree automaton with $L(A) = L$ and let $k \geq 0$ such that (L, H) is depth-k copying. The proof is by induction on k.

If $k = 0$ then for any $t \in L$ and $p \in \mathtt{Pos}(t)$ such that $t[p] \in \mathtt{Copy}(H)$, the path from the root of t to p does not contain non-erasing symbols. Let a be a non-erasing symbol. Let $Set_a = \{(q_1, \ldots, q_n) \mid \exists t(x), \delta(t(a(q_1, \ldots, q_n)) \in Q_f \wedge H(t(x)) = x\}$. Then, by assumption, for any $(q_1, \ldots, q_n)$ in Set_a, the languages $L(A, q_1), \ldots, L(A, q_n)$ do not contain any occurrence of symbols in $\mathtt{Copy}(H)$ at non-deleted positions. Hence, by Lemma 19, their images by H are recognizable languages, let us say $R_{q_1}, \ldots, R_{q_n}$. Now, any term u can be decomposed into $u_1(u_2)$ where $H(u_1(x)) = x$ and the root of

u_2 is labeled by a non-erasing symbol. So, $H(L)$ is the union of the $L(\langle\{H(a)\}, C\rangle)$ for a non-erasing, where $C(x_i) = R_{q_i}$. Thus, by introducing new variables for each such a and extending C appropriately, we obtain a set of terms with constraints $\langle S, C\rangle$ which equals $H(L)$.

Induction step: Let us suppose the lemma holds for pairs of regular tree language and homomorphism which are depth-k copying. Let (L, H) be depth-$(k+1)$ copying. This means that for any position p in t such that $t[p]$ belongs to $\mathtt{Copy}(H)$, the path from the root of t to p contains at most $k+1$ non-erasing symbols. Once more, let a be a non-erasing symbol and Set_a defined as before. Then $H(L)$ is the union of the $L(\langle H(a), \{(x_i, H(L(A, q_i)) \mid 1 \leq i \leq n\}\rangle)$ for non-erasing a of arity n and $(q_1, \ldots, q_n)$ in Set_a. The $H(L(A, q_i))$ are not necessarily regular but $(H, L(A, q_i))$ is depth-k copying. Thus, by induction hypothesis, for any $L(A, q_i)$ there exists some $\langle S_i, C_i\rangle$ such that $H(L(A, q_i)) = L(\langle S_i, C_i\rangle)$. By renaming of variables we can compose these sets $\langle S_i, C_i\rangle$ to obtain a set $\langle S, C\rangle$ such that $L(\langle S, C\rangle) = H(L)$. This concludes our inductive proof.

In order to show that the above proof is constructive, it suffices to show that Set_a is computable. Thus, we need to show how to compute the set of trees $t(x)$ for which $H(t(x)) = x$. To this end, let $\perp$ be a new constant and let G be the extension of H by $G(\perp) = \perp$. Then $G^{-1}(\perp)$ is effectively regular because inverses of tree transducers effectively preserve the regular tree languages (see, e.g., Proposition 20.1 of [7] where this is shown for the class of top-down tree transducers which includes the class of tree homomorphisms). Let $L_\perp = \{t[p \leftarrow \perp] \mid t \in L, p \in \mathtt{Pos}(t)\}$ be the regular tree language of all trees t in L in which exactly one node has been replaced by $\perp$. Then $G^{-1}(\perp) \cap L_\perp$ is the required set of trees $t(x)$ for which $H(t(x)) = x$. □

Using Lemma 20 and Theorem 17 we obtain our main result about pairs of regular tree languages L and homomorphisms H with decidable regularity of $H(L)$.

Theorem 21. *Let L be a regular tree language and H a tree homomorphism such that (L, H) is bounded-depth copying. It is decidable whether or not $H(L)$ is regular.*

Conclusions and Future Work. For a given regular tree language L and a tree homomorphism H we have shown that regularity of $H(L)$ is decidable in the following cases: (1) if L is monadic, i.e., only uses symbols of arity 1 or 0, (2) if non-linear rules of H are only applied at positions of bounded depth in the trees of L. For case (1) we have given a decision procedure that runs in polynomial time; furthermore, we have shown that, even in the monadic case, there are pairs of L, H that generate regular languages for which any smallest (nondeterministic) tree automaton is exponentially larger than the representation of L and H. It remains open what the precise complexity of the decision procedure of case (2) is: currently we apply the inverse image of a tree homomorphism in the proof, which can be expensive: for top-down tree transducers this problem is known to be EXPTIME-complete [13]. What is the complexity of this problem for tree homomorphisms? Can we drop the bounded-depth restriction of case (2) and extend decidability to homomorphisms that copy at most once? Or even to those that copy a bounded number of times? The latter is the "finite-copying" restriction [1], a well-studied topic in tree transducer theory.

References

1. Aho, A.V., Ullman, J.D.: Translations on a context-free grammar. Information and Control 19, 439–475 (1971)
2. Bogaert, B., Seynhaeve, F., Tison, S.: The recognizability problem for tree automata with comparison between brothers. In: Thomas, W. (ed.) FOSSACS 1999. LNCS, vol. 1578, pp. 150–164. Springer, Heidelberg (1999)
3. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (2002), `http://www.grappa.univ-lille3.fr/tata`
4. Comon, H., Delor, C.: Equational formulae with membership constraints. Inf. Comput. 112, 167–216 (1994)
5. Engelfriet, J.: Bottom-up and top-down tree transformations — A comparison. Math. Systems Theory 9, 198–231 (1975)
6. Fülöp, Z.: Undecidable properties of deterministic top-down tree transducers. Theoret. Comput. Sci. 134, 311–328 (1994)
7. Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, ch. 1, Springer, Heidelberg (1997)
8. Hopcroft, J.W., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley, Reading (1979)
9. Hosoya, H.: Foundations of XML processing (2002), `http://arbre.is.s.u-tokyo.ac.jp/~hahosoya/xmlbook/xmlbook.pdf`
10. Kucherov, G., Rusinowitch, M.: Patterns in words versus patterns in trees: A brief survey and new results. In: Ershov Memorial Conference, pp. 283–296 (1999)
11. Kucherov, G., Tajine, M.: Decidability of regularity and related properties of ground normal form languages. Inf. Comput. 118, 91–100 (1995)
12. Lassez, J.-L., Marriott, K.: Explicit representation of terms defined by counter examples. J. Autom. Reasoning 3, 301–317 (1987)
13. Martens, W., Neven, F.: On the complexity of typechecking top-down xml transformations. Theor. Comput. Sci. 336, 153–180 (2005)
14. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of xml schema languages using formal language theory. ACM Trans. Internet Techn. 5(4), 660–704 (2005)
15. Suciu, D.: The XML typechecking problem. SIGMOD Record 31, 89–96 (2002)