2012 27th Annual ACM/IEEE Symposium on Logic in Computer Science

# The HOM Problem is EXPTIME-Complete

Carles Creus  Adrià Gascón  Guillem Godoy  Lander Ramos
Departament de Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya,
Jordi Girona 1, Barcelona, Spain
ccreuslopez@gmail.com, adriagascon@gmail.com, ggodoy@lsi.upc.edu, landertxu@gmail.com

***Abstract*—The HOM problem questions whether the image of a given regular tree language through a given tree homomorphism is also regular. Decidability of HOM is an important theoretical question which was open for a long time. Recently, HOM has been proved decidable with a triple exponential time algorithm. In this paper we obtain an exponential time algorithm for this problem, and conclude that it is EXPTIME-complete. The proof builds upon previous results and techniques on tree automata with constraints.**

***Keywords*-Tree Automata, Tree Homomorphisms, Decision Problems**

## I. Introduction

A well-known classical property of regular (word) languages is that the application of any (word) homomorphisms preserves their regularity. This is no longer true for the natural extension from words to trees, since the image of a regular tree language by a tree homomorphism is not necessarily regular.

For example, consider the tree automaton $A$ defined by the rules $a \to q_1,\ b \to q_0,\ f(q_0, q_0) \to q_0,\ f(q_0, q_1) \to q_1,\ f(q_1, q_0) \to q_1,\ f(q_1, q_1) \to q_0$, where $q_0$ is the only accepting state. Note that $A$ accepts the tree language $L$ of terms over binary $f$ and nullaries $a, b$ having an even number of $a$'s. Also, consider the tree homomorphism $H$ defined by $H(a) = a,\ H(b) = a,\ H(f(x_1, x_2)) = g(H(x_1), H(x_1))$. Note that $H(L)$ is the set of complete trees over binary $g$ and nullary $a$ (see Figure 1). It is not difficult to prove that this language is not regular, since tree automata cannot test for equality between subterms.

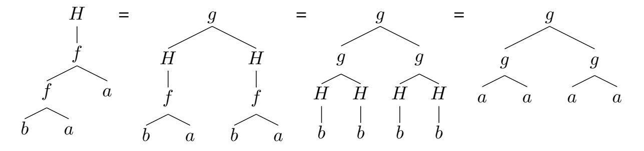


Fig. 1. Recursive application of the tree homomorphism $H$ to compute $H(f(f(b, a), a))$.

The HOM problem asks, for a given regular tree language $L$ described by a tree automaton and a tree homomorphism $H$, whether $H(L)$ is regular. Decidability of HOM is an important theoretical question which was open for a long time. From a practical perspective, preservation of regularity is a desirable property of any transformation system, since regular languages are expressive enough in different settings, are closed under many natural operations, and most of their properties are efficiently decidable. In the context of XML, extended DTD (document type descriptions) is a formalism expressively equivalent to tree automata. Thus, in this setting, a tree automaton is a type, i.e. it defines a set of valid descriptions, and HOM is equivalent to the following question: given a type $\tau$ and a homomorphic transformation $T$, is $T(\tau)$ a type?

The study of preservation of tree regularity by tree homomorphisms was introduced in [19]. In that paper, tree homomorphisms are defined for the first time, and it is proved that the application of linear tree homomorphisms preserves regularity. Tree homomorphisms are also introduced in [8], as a particular case of tree transducers. In [17], [20], [18], HOM is proved decidable for the particular cases where images are represented as instances of term patterns, or as reducible terms of a term rewrite system. In [3], HOM is proved decidable for the particular case of shallow tree homomorphisms. For the same particular case, it is shown in [7] that tree homomorphisms preserving regularity can be assumed to be linear. The HOM problem appears also in [10], where the more general case of regularity of the range of a top-down tree transducer is shown undecidable. In [16], HOM is proved decidable for the particular case where the regular tree language is defined over a monadic signature, and the case where images are represented as instances of term patterns constrained to regular tree languages. This particular case is proved to be EXPTIME-complete in [13]. As a consequence, HOM is EXPTIME-hard. Recently, in [15], HOM has been proved decidable. The obtained time complexity is triple exponential.

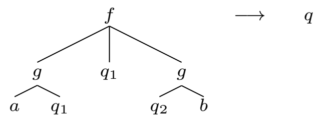
In this paper we extend the work done in [15] by proving that HOM is, in fact, EXPTIME-complete. As a first step we describe the results in [15] in terms of a new and simpler formalism, based on using identity of states to implicitly represent equality constraints (this idea already appears in the context of global constraints [9]). We build upon some intermediate results given in [15] represented with this new formalism, and the described algorithm is strongly based on the ideas introduced in [5] for proving EXPTIME-completeness of the emptiness problem for tree automata with disequality constraints.

### A. Contextualization and Approach

In this subsection, we give an overview of the results of [15] and the ideas of [5] that are used in the present paper, and describe the approach used in the proof of EXPTIME-completeness of HOM.

First of all, we briefly explain the approach of [15]. Due to technical complexity of the formalism used there, in the present paper we use a simpler formalism which is equivalent to the old one. In particular, in this introduction we explain the results of [15] in terms of the new formalism. Of course, in preliminary sections of this paper both formalisms are defined and shown equivalent.

More concretely, in [15], the regularity of the image of a regular tree language by a tree homomorphism is reduced to the question of whether the language recognized by a special kind of tree automata is regular. These automata are called *tree automata with HOM equality constraints*. Here, we define an expressively equivalent kind of tree automata called *simple tree automata with HOM equality constraints*, denoted $\mathtt{STA}_{\mathrm{hom}}$. These automata run bottom-up by bringing states from the leaves up to the root, and a term is accepted if the resulting state at the root is accepting. Their rules are of a special form, illustrated in the following example:



In the above example, $f$ is a symbol with arity 3, $g$ is a symbol with arity 2, $a, b$ are nullary symbols, and $q_1, q_2, q$ are states of the automaton, with $q_1 \neq q_2$. In general, a rule is a pair $l \to q$, where $q$ is a state, and $l$ is a term constructed over symbols of a signature, but also over state symbols, which are treated as nullaries. The application of $l \to q$ at a position of a term requires that the signature symbols match, and that the state symbols can be reached by the subterms at the corresponding relative positions. In addition, those of such subterms reaching the same state must be identical. In particular, the rule of the above example can only be applied at a subterm of the form $f(g(a,t_1), t_1, g(t_2, b))$ for terms $t_1$ and $t_2$ reaching $q_1$ and $q_2$, respectively. Thus, the fact that a state occurs more than once in the left-hand side of a rule, like $q_1$ in the example above, implicitly expresses an equality test.

In order to obtain simpler proofs, we add one further restriction to the $\mathtt{STA}_{\mathrm{hom}}$: each state which occurs more than once in the left-hand side of some rule is not accepting and cannot occur exactly once in the left-hand side of any other rule. We call such states *duplicating*. This additional restriction does not alter the expressiveness, and characterizes the subterms involved in an equality test as those ones reaching a duplicating state.

We denote as $\mathcal{L}(A)$ the set of terms accepted by an $\mathtt{STA}_{\mathrm{hom}}$ $A$. Now, consider a natural number $h$, and a more restrictive interpretation of the automaton, by additionally imposing that a subterm $t$ can reach a duplicating state only when the height of $t$ is smaller than or equal to $h$. We call $\mathcal{L}_h(A)$ to the set of terms accepted by $A$ interpreted in this way. It is clear that $\mathcal{L}_h(A) \subseteq \mathcal{L}(A)$ holds. Also, it is not difficult to see that $\mathcal{L}_h(A)$ is a regular tree language, since equality tests only need to be checked for a finite number of different terms. Thus, if we were able to find an $h$ for which the inclusion $\mathcal{L}(A) \subseteq \mathcal{L}_h(A)$ also holds, this would imply $\mathcal{L}(A) = \mathcal{L}_h(A)$ and hence, regularity of $\mathcal{L}(A)$. In [15], [14], a natural number $\tilde{h}$ exponentially bounded by the size of $A$ is defined, and the authors prove that such $\tilde{h}$ satisfies the following statement:

$$\mathcal{L}(A) \text{ is regular } \Leftrightarrow \mathcal{L}(A) \subseteq \mathcal{L}_{\tilde{h}}(A)$$

The proof of the above fact is very involved, but it leads to a simple test of regularity of $A$ in triple exponential time: a tree automaton recognizing $\overline{\mathcal{L}_{\tilde{h}}(A)}$ can be constructed such that its size is triple exponential with respect to the size of $A$, and next it can be checked whether $\mathcal{L}(A) \cap \overline{\mathcal{L}_{\tilde{h}}(A)}$ is empty in triple exponential time.

In the present paper we prove that the above inclusion can be tested in exponential time on the size of $A$. To this end, we avoid the construction of the automaton recognizing $\overline{\mathcal{L}_{\tilde{h}}(A)}$ and directly look for a witness of non-regularity, which is a term in $\mathcal{L}(A)$ and not in $\mathcal{L}_{\tilde{h}}(A)$. This search is done with a similar method to the one used in [5] for finding a term accepted by a tree automaton with disequality constraints. We generate a sequence of terms, where new terms are constructed using previous ones, with the hope of finding a minimal (in size) witness in the sequence. We define a criterion for detecting that some terms cannot be subterms of a minimal witness, in order to discard them. This criterion also ensures that this process terminates in exponential time, which concludes decidability of HOM in exponential time.

In Section II we introduce basic definitions and notations, and describe previous known results. In Section III we present the new formalism. In Section IV, we develop the criterion to detect that some terms cannot be subterms of a minimal witness. In Section V we present the algorithm.

## II. Preliminaries

### A. Terms

In this subsection we introduce notation for terms, positions, substitutions and replacements. For a survey see [1].

The size of a finite set $S$ is denoted by $|S|$. A *signature* consists of an alphabet $\Sigma$, i.e. a finite set of symbols, together with a mapping that assigns to each symbol in $\Sigma$ a natural number, its *arity*. We write $\Sigma^{(m)}$ to denote the subset of symbols in $\Sigma$ of arity $m$. The *set of all terms over* $\Sigma$ is denoted $\mathcal{T}(\Sigma)$ and is inductively defined as the smallest set $T$ such that for every $f \in \Sigma^{(m)}$, $m \geq 0$, and $t_1, \ldots, t_m \in T$, the term $f(t_1, \ldots, t_m)$ is in $T$. For a term of the form $a()$ we simply write $a$. We fix the set $\mathcal{X} = \{x_1, x_2, \ldots\}$ of variables, i.e. any set $V$ of variables is always assumed to be a subset of $\mathcal{X}$. The set of terms over $\Sigma$ with variables in $\mathcal{X}$, denoted $\mathcal{T}(\Sigma \cup \mathcal{X})$, is the set of terms over $\Sigma \cup \mathcal{X}$ where every symbol in $\mathcal{X}$ has arity zero.

By $|t|$ we denote the size of $t$, defined recursively as $|f(t_1, \ldots, t_m)| = 1 + |t_1| + \ldots + |t_m|$ for each $f \in \Sigma^{(m)}$, $m \geq 0$, and $t_1, \ldots, t_m \in \mathcal{T}(\Sigma)$. By $\mathsf{height}(t)$ we denote the height of $t$, defined recursively as $\mathsf{height}(f(t_1, \ldots, t_m)) = 1 + \max\{\mathsf{height}(t_1), \ldots, \mathsf{height}(t_m)\}$ for each $f \in \Sigma^{(m)}$,

$m \geq 1$, and $t_1, \ldots, t_m \in \mathcal{T}(\Sigma)$, and $\mathsf{height}(a) = 0$ for each $a \in \Sigma^{(0)}$. Positions in terms are sequences of natural numbers. Given a term $f(t_1, \ldots, t_m) \in \mathcal{T}(\Sigma)$, its set of positions $\mathsf{Pos}(t)$ is defined recursively as $\{\lambda\} \cup_{1 \leq i \leq m} \{i.p \mid p \in \mathsf{Pos}(t_i)\}$. Here, $\lambda$ denotes the empty sequence (position of the root node) and . denotes concatenation. The length of a position is denoted as $|p|$. Note that $|\lambda| = 0$ and $|i.p| = 1 + |p|$ hold. The subterm of $t$ at position $p$ is denoted by $t|_p$, and is formally defined as $t|_\lambda = t$ and $f(t_1, \ldots, t_m)|_{i.p} = t_i|_p$. For terms $s, t$ and position $p \in \mathsf{Pos}(s)$, we denote by $s[t]_p$ the result of replacing the subterm at position $p$ in $s$ by the term $t$. More formally, $s[t]_\lambda$ is $t$ and $f(s_1, \ldots, s_m)[t]_{i.p}$ is $f(s_1, \ldots, s_{i-1}, s_i[t]_p, s_{i+1}, \ldots, s_m)$. The symbol of $t$ occurring at position $p$ is denoted by $t(p)$. More formally, $f(s_1, \ldots, s_m)(\lambda) = f$ and $f(s_1, \ldots, s_m)(i.p) = s_i(p)$. For a set $\Gamma$, we use $\mathsf{Pos}_\Gamma(t)$ to denote the set of positions of $t$ that are labeled by symbols in $\Gamma$. When a position $p$ is of the form $p_1.p_2$, we say that $p_1$ is a prefix of $p$, denoted $p_1 \leq p$, and $p_2$ is a suffix of $p$. If in addition $p_2$ is not $\lambda$, then we say that $p_1$ is a proper prefix of $p$, denoted $p_1 < p$. Moreover, with $p - p_1$ we denote $p_2$. We say that two positions $p_1$ and $p_2$ are parallel, denoted $p_1 \parallel p_2$, if neither $p_1 \leq p_2$ nor $p_2 \leq p_1$ hold.

A *substitution* $\sigma$ is a mapping from variables to terms. It can be homomorphically extended to a function from terms to terms: $\sigma(t)$ denotes the result of simultaneously replacing in $t$ every $x \in \mathsf{Dom}(\sigma)$ by $\sigma(x)$. Substitutions are sometimes written as finite sets of pairs $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, where each $x_i$ is a variable and each $t_i$ is a term.

## B. Tree Automata

Tree automata and regular tree languages are well-known concepts of theoretical computer science [11], [12], [4]. We assume that the reader knows the Boolean closure properties and the decidability results on regular tree languages. Here we only recall the notion of tree automata.

**Definition II.1** (Tree Automata). *A* tree automaton, `TA` *for short, is a tuple $A = \langle Q, \Sigma, F, \Delta \rangle$, where $Q$ is a set of states, $\Sigma$ is a signature, $F \subseteq Q$ is the subset of final states, and $\Delta$ is a set of rules of the form $f(q_1, \ldots, q_m) \to q$, where $q_1, \ldots, q_m, q$ are in $Q$, and $f$ is in $\Sigma^{(m)}$.*

*The size of $A$, denoted $|A|$, is $|Q|$ plus the sum of sizes of all rules in $\Delta$, where the size of a rule of the form $f(q_1, \ldots, q_m) \to q$ is $m + 2$.*

*A* run *of $A$ on a term $t \in \mathcal{T}(\Sigma)$ is a mapping $r : \mathsf{Pos}(t) \to \Delta$ such that, for each position $p \in \mathsf{Pos}(t)$, if $t|_p$ is of the form $f(t_1, \ldots, t_m)$, then $r(p)$ is a rule of the form $f(q_1, \ldots, q_m) \to q$, and $r(p.1), \ldots, r(p.m)$ are rules with right-hand sides $q_1, \ldots, q_m$, respectively. We say that $r(p)$ is the rule applied at position $p$. The run $r$ is called* accepting *if the right-hand side of $r(\lambda)$ is a state in $F$. A term $t$ is accepted by $A$ if there exists an accepting run of $A$ on $t$. The set of accepted terms by $A$, also called the* language recognized *by $A$, is denoted by $\mathcal{L}(A)$.*

It is usual to treat a run as a term over the alphabet $\Delta$, where the label at position $p$ is precisely the rule applied at position $p$. Note that in this way, the term/run $r$ implicitly defines the term $t$ on which $r$ is executed. For example, if $r(p)$ is $f(q_1, \ldots, q_m) \to q$, then $t(p)$ is $f$. This way, it is natural to define $r|_p$ as the subrun of $r$ at position $p$.

## C. Tree Homomorphisms

**Definition II.2** (tree homomorphisms)**.** *Let $\Sigma_1, \Sigma_2$ be two signatures. A* tree homomorphism *is a function $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ which can be defined as follows.*

*Let $X_m$ represent the set of variables $\{x_1, \ldots, x_m\}$ for each natural number $m$. The definition of a tree homomorphism $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ requires to define $H(f(x_1, \ldots, x_m))$ for each function symbol $f \in \Sigma_1$ of arity $m$ as a term $t_f$ in $\mathcal{T}(\Sigma_2 \cup X_m)$. After that, $H(f(t_1, \ldots, t_m))$ is defined, for each term $f(t_1, \ldots, t_m) \in \mathcal{T}(\Sigma_1)$, as $\{x_1 \mapsto H(t_1), \ldots, x_m \mapsto H(t_m)\}(t_f)$.*

*The size of $H$, denoted $|H|$, is the sum of the sizes of all the terms $t_f$.*

**Definition II.3.** *The HOM problem is defined as follows:*
**Input:** *A `TA` $A$ and a tree homomorphism $H$.*
**Question:** *Is $H(\mathcal{L}(A))$ regular?*

## D. Tree Automata with HOM Equality Constraints

**Definition II.4** ([15])**.** *A* tree automaton with HOM equality constraints, $\mathtt{TA}_{\mathrm{hom}}$ *for short, is a tuple $A = \langle Q, \Sigma, F, \Delta \rangle$, where $Q$ is a set of states, $\Sigma$ is a signature, $F \subseteq Q$ is the subset of final states, and $\Delta$ is a set of rules of the form $l \xrightarrow{c} q$, where $l$ is a term in $\mathcal{T}(\Sigma \cup Q) - Q$, interpreting the states of $Q$ as 0-ary symbols, and $c$ is a conjunction/set of atoms of the form $(p_1 \approx p_2)$, where $p_1$ and $p_2$ are different positions in $\mathsf{Pos}(l)$ satisfying $l(p_1) = l(p_2) \in Q$. Moreover, for all positions $p_1, p_2, p_3$, if $(p_1 \approx p_2)$ and $(p_2 \approx p_3)$ occur in $c$, then $(p_1 \approx p_3)$ also occurs in $c$.*

*The size of $A$, denoted $|A|$, is $|Q|$ plus the sum of sizes of all rules in $\Delta$, where the size of a rule $l \xrightarrow{c} q$ is $|l| + |c| + 1$, and $|c|$ is the sum of the lengths of all the occurrences of positions in $c$.*

*A* run *of $A$ on a term $t \in \mathcal{T}(\Sigma)$ is a partial mapping $r : \mathsf{Pos}(t) \to \Delta$ such that $r(\lambda)$ is defined, and satisfying the following conditions for each position $p$ for which $r(p)$ is defined, say, as a rule $l \xrightarrow{c} q$. For each position $p' \in \mathsf{Pos}(l)$, it holds that $r(p.p')$ is defined if and only if $l(p') \in Q$. Moreover, if such $l(p')$ is in $Q$, then $r(p.p')$ is a rule with the state $l(p')$ as right-hand side. Otherwise, if $l(p')$ is in $\Sigma$, then $l(p') = t(p.p')$. In addition, for each $(p_1 \approx p_2) \in c$, $t|_{p.p_1} = t|_{p.p_2}$ holds.*

*The* state reached *by $r$ is the right-hand side of $r(\lambda)$. The run $r$ is* accepting *if the state reached by $r$ is in $F$. By $\mathcal{L}(A)$ we denote the* language recognized *by $A$, that is the set of terms $t$ for which there exists an accepting run of $A$ on $t$.*

In the original definition of [15], an atom $(p_1 \approx p_2)$ requires not only identity of subterms, but also identity of subruns. Nevertheless, it is also shown there that both conditions lead to the same expressiveness.

The following proposition establishes that $\mathtt{TA}_{\mathrm{hom}}$ can be used to represent images of regular tree languages through tree homomorphisms.

**Proposition II.5.** *Let $B = \langle Q, \Sigma_1, F, \Delta \rangle$ be a $\mathtt{TA}$. Let $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ be a tree homomorphism. Then, a $\mathtt{TA}_{\mathrm{hom}}$ $A$ can be computed satisfying $H(\mathcal{L}(B)) = \mathcal{L}(A)$ and $|A| \leq |B| \cdot (|H| + |H|^3)$.*

In [15], the regularity of a $\mathtt{TA}_{\mathrm{hom}}$ is characterized in terms of the following definition.

**Definition II.6** (linearization of a $\mathtt{TA}_{\mathrm{hom}}$)**.** *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathrm{hom}}$. Let $\mathtt{h}$ be a natural number. The* linearization *of $A$ by $\mathtt{h}$ is the $\mathtt{TA}_{\mathrm{hom}}$ $\langle Q, \Sigma, F, \Delta' \rangle$, denoted* $\mathsf{linearize}(A, \mathtt{h})$*, where $\Delta'$ is the set of all rules of the form $l[s_1]_{p_1} \ldots [s_n]_{p_n} \to q$ such that:*

- *A rule of the form $l \xrightarrow{c} q$ occurs in $\Delta$.*
- *$p_1, \ldots, p_n$ are the positions occurring in $c$.*
- *For each $i \in \{1, \ldots, n\}$, $s_i$ is a term such that* $\mathsf{height}(s_i) \leq \mathtt{h}$ *and there is a run of $A$ on it reaching the state $l(p_i)$.*
- *For each $i, j \in \{1, \ldots, n\}$ such that $(p_i \approx p_j)$ occurs in $c$, $s_i = s_j$ holds.*

It is straightforward that a linearization of any $\mathtt{TA}_{\mathrm{hom}}$ is computable and recognizes a regular tree language, since no equality constraints appear. It is also clear that $\mathcal{L}(A)$ includes the language of any of its linearizations. Moreover, in the case where $\mathcal{L}(A)$ is included in some of its linearizations, we can conclude that $\mathcal{L}(A)$ is regular. The following lemma and definition from [15], [14] bound the $h$ for which we should test such inclusion.

**Lemma II.7.** *There exists a polynomial $\mathcal{P}$ satisfying the following condition. Let $A$ be a $\mathtt{TA}_{\mathrm{hom}}$. Let $\tilde{h}$ be $2^{\mathcal{P}(|A|)}$. Then, $\mathcal{L}(A)$ is regular if and only if $\mathcal{L}(A) \subseteq \mathcal{L}(\mathsf{linearize}(A, \tilde{h}))$.*

**Definition II.8.** *Let $A$ be a $\mathtt{TA}_{\mathrm{hom}}$. By $\tilde{h}(A)$ we denote the function $2^{\mathcal{P}(|A|)}$ given by Lemma II.7, and write $\tilde{h}$ when $A$ is clear from the context.*

Lemma II.7 leads to a simple decision algorithm of HOM: $\mathcal{L}(A)$ is regular if and only if $\mathcal{L}(A) \subseteq \mathcal{L}(\mathsf{linearize}(A, \tilde{h}))$ if and only if $\mathcal{L}(A) \cap \overline{\mathcal{L}(\mathsf{linearize}(A, \tilde{h}))} \neq \emptyset$, $\overline{\mathcal{L}(\mathsf{linearize}(A, \tilde{h}))}$ can be recognized by a $\mathtt{TA}$ of triple exponential size, and non-emptiness of its intersection with $\mathcal{L}(A)$ can be tested in triple exponential time.

## III. Simple Tree Automata with HOM Equality Constraints

In [15], the authors use $\mathtt{TA}_{\mathrm{hom}}$ to describe the image of a regular tree language through a tree homomorphism. This formalism is a straightforward application of a homomorphism to the rules of a $\mathtt{TA}$. Its main disadvantage is that dealing with constrainted rules makes the presentation of technical proofs a laborious task. For this reason we define an equivalent kind of automata in which equality tests are implicitly encoded in the states.

**Definition III.1.** *A simple tree automaton with HOM equality constraints, $\mathtt{STA}_{\mathrm{hom}}$ for short, is a tuple $A = \langle Q, \Sigma, F, \Delta \rangle$, where $Q$ is a set of states, $\Sigma$ is a signature, $F \subseteq Q$ is the subset of final states, and $\Delta$ is a set of rules of the form $l \to q$, where $l$ is a term in $\mathcal{T}(\Sigma \cup Q) - Q$, interpreting the states of $Q$ as 0-ary symbols. Moreover, if a state $q \in Q$ occurs more than once in the left-hand side of some rule of $\Delta$, then it is not in $F$ and does not occur just once in the left-hand side of any rule in $\Delta$. We call* duplicating *to such states and* $\mathsf{Dup}(A)$ *to the set of such states.*

*By $h(A)$ we denote the maximum among the heights of left-hand sides of rules in $\Delta$. The size of $A$, denoted $|A|$, is $|Q|$ plus the sum of sizes of all rules in $\Delta$, where the size of a rule $l \to q$ is $|l| + 1$.*

*A* run *of $A$ on a term $t \in \mathcal{T}(\Sigma)$ is a partial mapping $r : \mathsf{Pos}(t) \to \Delta$ such that $r(\lambda)$ is defined, and satisfying the following conditions for each position $p$ for which $r(p)$ is defined, say, as a rule $l \to q$. For each position $p' \in \mathsf{Pos}(l)$, it holds that $r(p.p')$ is defined if and only if $l(p') \in Q$. Moreover, if $l(p')$ is in $Q$, then $r(p.p')$ is a rule with the state $l(p')$ as right-hand side. Otherwise, if $l(p')$ is in $\Sigma$, then $l(p') = t(p.p')$. In addition, for each two positions $p_1, p_2 \in \mathsf{Pos}(l)$ satisfying $l(p_1) = l(p_2) \in Q$, $t|_{p.p_1} = t|_{p.p_2}$ holds.*

*The* state reached *by $r$ is the right-hand side of $r(\lambda)$. The run $r$ is* accepting *if the state reached by $r$ is in $F$. By $\mathcal{L}(A)$ we denote the* language recognized *by $A$, that is the set of terms $t$ for which there exists an accepting run of $A$ on $t$. Moreover, for a given term $t \in \mathcal{T}(\Sigma)$, we denote by $A(t)$ the set of states reachable by runs of $A$ on $t$.*

**Example III.2.** *Consider the tree language $L = \{f(g^n(a), g^n(a)) \mid n \geq 0\}$. We define an $\mathtt{STA}_{\mathrm{hom}}$ $A = \langle Q, \Sigma, F, \Delta \rangle$ recognizing $L$ as $Q = \{q, q_{\mathsf{dup}}, q_{\mathsf{accept}}\}$, $F = \{q_{\mathsf{accept}}\}$ and $\Delta = \{a \to q,\ a \to q_{\mathsf{dup}},\ g(q) \to q,\ g(q) \to q_{\mathsf{dup}},\ f(q_{\mathsf{dup}}, q_{\mathsf{dup}}) \to q_{\mathsf{accept}}\}$. Note that, although the same set of terms reach $q$ and $q_{\mathsf{dup}}$, we need both states due to the restriction on the states in* $\mathsf{Dup}(A)$*.*

Analogously to tree automata, we say that $r|_p$ is the subrun of a run $r$ at position $p$, when $r(p)$ is defined.

As mentioned for $\mathtt{TA}_{\mathrm{hom}}$, duplicating states occurring in a rule of an $\mathtt{STA}_{\mathrm{hom}}$ can be assumed to ask for equality not only of subterms, but also of subruns. Runs holding this property are called regular runs.

**Definition III.3.** *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. A run $r$ of $A$ is called* regular *if, for each positions $p, p_1, p_2$ such that $r(p)$ is defined as a rule $l \to q$ satisfying $l(p_1) = l(p_2) \in Q$, $r|_{p.p_1} = r|_{p.p_2}$ holds.*

**Lemma III.4.** *Let $A$ be an $\mathtt{STA}_{\mathrm{hom}}$. Then, any run of $A$ can be transformed into a regular run on the same term and reaching the same state.*

As mentioned in the introduction, we do not want to explicitly construct the linearization of an $\mathtt{STA}_{\mathrm{hom}}$. Instead, we use a new notion of run where equality tests can only be satisfied when the height of the involved subterms is bounded.

**Definition III.5.** *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. Let $h$ be a natural number. Let $r$ be a run of $A$ on a term $t \in \mathcal{T}(\Sigma)$. We say that $r$ is an* $h$-run *if for each subrun $r|_p$ of $r$ reaching a duplicating state,* $\mathsf{height}(t|_p) \leq h$ *holds.*

*We define $\mathcal{L}_h(A)$ as the set of terms of $\mathcal{T}(\Sigma)$ reaching an accepting state in an $h$-run of $A$. Moreover, for a given term $t \in \mathcal{T}(\Sigma)$, we denote by $A_h(t)$ the set of states reachable by $h$-runs of $A$ on $t$.*

**Example III.6.** *Consider the tree language $L$ and the $\mathtt{STA}_{\mathrm{hom}}$ $A$ of Example III.2. Note that $A(f(g^n(a), g^n(a))) = \{q_{\mathsf{accept}}\}$ for all natural number $n$. But given a natural number $h$, $A_h(f(g^n(a), g^n(a))) = \{q_{\mathsf{accept}}\}$ only holds for those $n \leq h$. Otherwise, for $n > h$ we have $A_h(f(g^n(a), g^n(a))) = \emptyset$.*

The following lemmas state equivalence in expressiveness between $\mathtt{TA}_{\mathrm{hom}}$ and $\mathtt{STA}_{\mathrm{hom}}$.

**Lemma III.7.** *There is a polynomial time algorithm that transforms a given $\mathtt{TA}_{\mathrm{hom}}$ $A$ into an $\mathtt{STA}_{\mathrm{hom}}$ $A'$ satisfying $\mathcal{L}(A') = \mathcal{L}(A)$ and $|A'| \leq 2 \cdot |A|^2$.*

**Lemma III.8.** *There is a polynomial time algorithm that transforms a given $\mathtt{STA}_{\mathrm{hom}}$ $A$ into a $\mathtt{TA}_{\mathrm{hom}}$ $A'$ such that $\mathcal{L}(A) = \mathcal{L}(A')$, $|A'| \leq |A| + |A|^3$, and for each natural number $h$, $\mathcal{L}_h(A) = \mathcal{L}(\mathsf{linearize}(A', h))$.*

The concrete transformations stated in the previous two lemmas can be found in the long version of this paper [6].

The following definition and corollary capture the results in [15] adapted to our new formalism. The corollary follows trivially from Lemmas II.7 and III.8.

**Definition III.9.** *Let $A$ be an $\mathtt{STA}_{\mathrm{hom}}$. Let $A'$ be the $\mathtt{TA}_{\mathrm{hom}}$ obtained from $A$ using the transformation stated in Lemma III.8. We define $\tilde{h}(A)$ as $\tilde{h}(A')$. As before, we write $\tilde{h}$ when $A$ is clear from the context.*

**Corollary III.10.** *Let $A$ be an $\mathtt{STA}_{\mathrm{hom}}$. Then, $\mathcal{L}(A)$ is regular if and only if $\mathcal{L}(A) \subseteq \mathcal{L}_{\tilde{h}}(A)$.*

Thanks to previous corollary, non-regularity is guaranteed when $\mathcal{L}(A) \not\subseteq \mathcal{L}_{\tilde{h}}(A)$. Our approach to decide non-regularity is to find a pair term-state that is a witness of non-regularity according to the following definition.

**Definition III.11.** *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. A pair term-state $\langle t, q\rangle \in \mathcal{T}(\Sigma) \times Q$ is called a* feasible pair *of $A$ if $q \in A(t)$. Moreover, it is called a* witness *of the non-regularity of $A$ if $q \in F$ and $t \notin \mathcal{L}_{\tilde{h}}(A)$.*

In order to reduce the search space, we look for the minimum witness with respect to a total ordering (essentially size ordering plus a notion for comparing terms with equal size). We assume a given well-founded ordering $\ll$, total on terms, fulfilling the strict size relation (if $|t| < |t'|$ then $t \ll t'$) and monotonic (if $s \ll t|_p$ then $t[s]_p \ll t$). Note that particular cases of the Knuth-Bendix ordering [2] satisfy these conditions. We compare pairs term-state by the lexicographic extension of $\ll$ and an arbitrary total ordering on states. We also use $\ll$ to denote the ordering of pairs term-state. To find the minimum witness efficiently, we do not want to consider feasible pairs that cannot be used to construct it. The fact that a pair is used to construct another one can be formalized as follows.

**Definition III.12.** *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. Let $\langle t, q\rangle, \langle t', q'\rangle \in \mathcal{T}(\Sigma) \times Q$ be pairs term-state. We say that $\langle t', q'\rangle$ is a* piece *of $\langle t, q\rangle$ if there exists a run $r$ of $A$ on $t$ reaching $q$ such that there is a position $p \in \mathsf{Pos}(t)$ satisfying that $r(p)$ is defined, $r|_p$ reaches $q'$, and $t|_p = t'$.*

## IV. The Discarding Criterion

We need a criterion to determine that a pair $\langle t, q\rangle$ is not a piece of the minimum witness. To this end, we will have to prove that, whenever $\langle t, q\rangle$ is a piece of a witness $\langle w, q_f\rangle$, then this witness is not minimum. This is done by considering an alternative pair $\langle s, q\rangle$ smaller than $\langle t, q\rangle$ with respect to $\ll$, and replacing occurrences of $t$ in $w$ by $s$, thus obtaining a smaller witness. In general, it is not enough to replace just one occurrence of $t$ by $s$: the need to do multiple replacements is related to the equalities imposed by the implicit equality constraints in the rules of $A$. Thus, several positions $p_1, \ldots, p_n$ must be replaced at once in order to obtain a new correct run of $A$, i.e. we will obtain a new run reaching $q_f$ on the term $w' = w[s]_{p_1} \ldots [s]_{p_n}$ by replacing some subruns on $t$ reaching $q$ by runs on $s$ reaching $q$. But for $\langle w', q_f\rangle$ to be a witness it is also required that $A_{\tilde{h}}(w')$ does not contain an accepting state. Thus, the term $s$ chosen as an smaller alternative to $t$ must ensure this condition, too. To this end it suffices to guarantee that a rule which was not applicable at a position above some position of $p_1, \ldots, p_n$, must not be applicable after the replacement.

It might be the case that a rule $l \to \bar{q}$ is not applicable at a position of $w$ because the symbols or the states of $l$ do not match. By choosing $s$ to coincide with $t$ at its topmost positions, in symbols and states reachable with $\tilde{h}$-runs, we guarantee that $l \to \bar{q}$ is not applicable in $w'$ either. For this reason, in Section IV-A we define an equivalence relation between terms assuring that two equivalent terms match in symbols and reached states with $\tilde{h}$-runs at the topmost positions.

It still might be the case that a rule $l \to \bar{q}$ is not applicable in $w$ but applicable in $w'$ with $\tilde{h}$-runs because an equality constraint not satisfied in $w$ becomes satisfied in $w'$. If this happens, we must look for an alternative term to $s$. In Sections IV-B and IV-C we define a notion of independence of a set of terms $\{s_1, \ldots, s_k, t\}$ ensuring that for a given equality constraint that is not satisfied in $w$, at most one of the $s_i$'s makes it satisfied when using it to replace $t$. Thus, the rest of $s_j$'s are valid alternatives to keep that equality constraint unsatisfied. Moreover, it suffices to consider an exponential number $K(A)$ of equality constraints that might become satisfied after the replacement. In summary, the discarding criterion for $\langle t, q\rangle$ as a piece of the minimum witness, consists on finding a set of feasible pairs $\{\langle s_1, q\rangle, \ldots, \langle s_k, q\rangle\}$ bigger than $K(A)$ such that the $s_i$'s are equivalent to $t$ and smaller than $t$ with respect to $\ll$, and $\{s_1, \ldots, s_k, t\}$ is independent.

## A. Equivalence Relations

In this section we define an equivalence relation $\sim^A$ on terms, induced by an $\mathtt{STA}_{\mathrm{hom}}$ $A$, which is necessary for describing our criterion for detecting feasible pairs that are not pieces of the minimum witness. This relation is defined as the conjunction of some other equivalence relations, which will be useful later for proving that the criterion is correct. In any of such relations, any two related terms must share the same set of positions that are suffixes of positions in left-hand sides of rules of $A$.

**Definition IV.1.** *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. We define the set of positions $\mathsf{suff}(\Delta)$ as the set of suffixes of positions occurring in left-hand sides of rules in $\Delta$, more formally $\mathsf{suff}(\Delta) = \{p \mid \exists (l \to q) \in \Delta, \exists p' : p'.p \in \mathsf{Pos}(l)\}$.*

We roughly describe the relations appearing in the following definition. As we have mentioned, all the relations impose that the corresponding $t_1$ and $t_2$ share the same positions in $\mathsf{suff}(\Delta)$. Additionally, they impose other properties involving the corresponding subterms at such positions. In particular, the expression $t_1 \sim^A_= t_2$ says that $t_1$ and $t_2$ satisfy the same equality and disequality relations among such subterms, $t_1 \sim^A_{\mathsf{ht}} t_2$ says that corresponding subterms of $t_1$ and $t_2$ have similar heights, $t_1 \sim^A_\Sigma t_2$ says that $t_1$ and $t_2$ label identically such positions, and $t_1 \sim^A_Q t_2$ says that the states reachable by $\tilde{h}$-runs on the subterms of $t_1$ and $t_2$ at such positions coincide.

**Definition IV.2.** *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. We define the following equivalence relations on $\mathcal{T}(\Sigma)$:*

$$\begin{aligned}
t_1 \sim^A_{\mathsf{suff}} t_2 &\Leftrightarrow \forall p \in \mathsf{suff}(\Delta) : p \in \mathsf{Pos}(t_1) \Leftrightarrow p \in \mathsf{Pos}(t_2)\\
t_1 \sim^A_= t_2 &\Leftrightarrow t_1 \sim^A_{\mathsf{suff}} t_2 \ \wedge\ \forall p_1, p_2 \in \mathsf{suff}(\Delta) \cap \mathsf{Pos}(t_1) :\\
&\qquad t_1|_{p_1} = t_1|_{p_2} \Leftrightarrow t_2|_{p_1} = t_2|_{p_2}\\
t_1 \sim^A_{\mathsf{ht}} t_2 &\Leftrightarrow t_1 \sim^A_{\mathsf{suff}} t_2 \ \wedge\ \forall p \in \mathsf{suff}(\Delta) \cap \mathsf{Pos}(t_1) :\\
&\qquad (\mathsf{height}(t_1|_p) = \mathsf{height}(t_2|_p)) \vee\\
&\qquad (\mathsf{height}(t_1|_p), \mathsf{height}(t_2|_p) > \tilde{h})\\
t_1 \sim^A_\Sigma t_2 &\Leftrightarrow t_1 \sim^A_{\mathsf{suff}} t_2 \ \wedge\ \forall p \in \mathsf{suff}(\Delta) \cap \mathsf{Pos}(t_1) :\\
&\qquad t_1(p) = t_2(p)\\
t_1 \sim^A_Q t_2 &\Leftrightarrow t_1 \sim^A_{\mathsf{suff}} t_2 \ \wedge\ \forall p \in \mathsf{suff}(\Delta) \cap \mathsf{Pos}(t_1) :\\
&\qquad A_{\tilde{h}}(t_1|_p) = A_{\tilde{h}}(t_2|_p)\\
t_1 \sim^A_{=,\mathsf{ht}} t_2 &\Leftrightarrow (t_1 \sim^A_= t_2) \wedge (t_1 \sim^A_{\mathsf{ht}} t_2)\\
t_1 \sim^A_{\Sigma,Q} t_2 &\Leftrightarrow (t_1 \sim^A_\Sigma t_2) \wedge (t_1 \sim^A_Q t_2)\\
t_1 \sim^A t_2 &\Leftrightarrow (t_1 \sim^A_{=,\mathsf{ht}} t_2) \wedge (t_1 \sim^A_{\Sigma,Q} t_2)
\end{aligned}$$

The following lemma bounds the number of different equivalence classes induced by an $\mathtt{STA}_{\mathrm{hom}}$, which is later needed to prove that our algorithm terminates in exponential time. (the proof is given in the long version of this paper [6]).

**Lemma IV.3.** *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. The number of different equivalence classes induced by $\sim^A$ is bounded by $2^{|\mathsf{suff}(\Delta)|} \cdot |\mathsf{suff}(\Delta)|^{|\mathsf{suff}(\Delta)|} \cdot (\tilde{h} + 2)^{|\mathsf{suff}(\Delta)|} \cdot |\Sigma|^{|\mathsf{suff}(\Delta)|} \cdot 2^{|Q| \cdot |\mathsf{suff}(\Delta)|}$.*

## B. Independent Sets of Tuples

Here we present preliminary definitions and results that, in the following section, are translated to the specific notion of independence introduced at the begining of Section IV. The contents of this section is rather abstract and its results seem, at first look, to fit better in a handbook on combinatorics than in a paper on tree automata. Nevertheless, in [5], similar notions were needed for proving EXPTIME-completeness of emptiness for tree automata with disequality constraints. We explain the differences with such notions after Definition IV.4.

We assume a given set (the universe) $U$ and a natural number $n$, and work with $n$-tuples $t = \langle e_1, \ldots, e_n \rangle$ of elements of $U$. For such a tuple $t$, with $t[i]$ we denote the $i$'th component $e_i$. We denote the set of all such possible tuples as $T$. For a given finite set of tuples, we are interested on finding a "big" subset which is independent according to the following definition.

**Definition IV.4.** *A finite set of tuples $\{t_1, \ldots, t_k\} \subseteq T$ is independent if for all $i \in \{1, \ldots, n\}$, either all elements $t_1[i], \ldots, t_k[i]$ are the same, or the elements $t_1[i], \ldots, t_k[i]$ are pairwise different.*

Note that, if a set is independent, then any of its subsets also is.

In [5], it is proved for a fixed natural number $K$ that, given a set $S$ with $K^n n!$ tuples, there exists an independent subset $\tilde{S}$ of $S$ with size $K$. This fact is used in [5] to decide emptiness of tree automata with disequality constraints. In order to produce simpler arguments in our setting, we need more than just the existence of such $\tilde{S}$. We also need to ensure that a certain tuple $t$ in $S$ is also in $\tilde{S}$. As a first step, we note that, since all tuples of an independent subset coincide at certain components, we can restrict our search of such $\tilde{S}$ to subsets of $S$ whose tuples already coincide with $t$ at some fixed components.

**Definition IV.5.** *Let $S, t, I$ be such that $t \in S \subseteq T$ and $I \subseteq \{1, \ldots, n\}$. We define the set of tuples $\mathsf{coincidents}(S, t, I)$ as $\{t' \in S \mid \forall i \in I : t'[i] = t[i]\}$.*

Note that, if $t'$ belongs to $\mathsf{coincidents}(S, t, I)$, then $\mathsf{coincidents}(S, t', I) = \mathsf{coincidents}(S, t, I)$.

Finding all such independent subsets is trivially computable. Yet, it is too hard for our purposes. For this reason, we instead focus on a different approach: we define a criterion that, when satisfied, guarantees the existence of one such independent subset $\tilde{S}$. More formally, for a natural number $K$, we define a counting property on sets of tuples such that, when it is not satisfied by a given set $S \subseteq T$, it follows the existence of an independent subset $\tilde{S} \subseteq S$ of size $K$. Such property is defined as follows.

**Definition IV.6.** *Let $K$ be a natural number. Let $S \subseteq T$ be a set of tuples. We say that $S$ is $K$-small if the following statement holds:*

$$\forall t \in S : \forall I \subsetneq \{1, \ldots, n\} : |\mathsf{coincidents}(S, t, I)| < K^{n-|I|}(n - |I|)!$$

The following lemma states that $K$-small sets are indeed "small".

**Lemma IV.7.** *Let $K$ be a natural number. Let $S \subseteq T$ be a non-empty $K$-small set. Then, $|S| < K^n n!$ holds.*

*Proof:* Note that for any tuple $t \in S$, $S = \mathsf{coincidents}(S, t, \emptyset)$ holds. Thus, $|S| = |\mathsf{coincidents}(S, t, \emptyset)| < K^{n-|\emptyset|}(n - |\emptyset|)! = K^n n!$ holds. ∎

In order to show that detecting $K$-smallness is enough for our purposes, we prove in Lemma IV.12 that, given a set $S$ and a tuple $t \in S$ such that $S - \{t\}$ is $K$-small but $S$ is not, there always exists an independent subset $\tilde{S} \subseteq S$ of size $K$ and including $t$. To this end, as a first ingredient, we relate the existence of an independent subset of $S$ with the existence of an edge-free subset of nodes of a graph. In the literature, edge-free subsets of nodes are simply called independent. Here, we use this other name in order to avoid confusion with our notion of independent sets of tuples.

**Definition IV.8.** *Let $G = \langle V, E\rangle$ be an undirected graph. Let $\tilde{V}$ be a subset of $V$. We say that $\tilde{V}$ is edge-free in $G$ if each two nodes of $\tilde{V}$ are not connected, i.e. if $\{(u, v) \mid u, v \in \tilde{V}\} \cap E = \emptyset$ holds.*

The graph where we want to find edge-free subsets of nodes is defined to have $\mathsf{coincidents}(S, t, I)$ as its set of nodes, for a fixed $I$, and to have an edge between each two different tuples $t_1, t_2$ if and only if $t_1$ and $t_2$ coincide at some component not in $I$. This is defined formally as follows.

**Definition IV.9.** *Let $S, t, I$ be such that $t \in S \subseteq T$ and $I \subseteq \{1, \ldots, n\}$. We define $\mathsf{graph}(S, t, I)$ as:*

$$\left\langle \begin{array}{l} \mathsf{coincidents}(S, t, I), \\ \{(t_1, t_2) \in \mathsf{coincidents}(S, t, I)^2 \mid \\ \quad t_1 \neq t_2 \wedge \exists i \in \{1, \ldots, n\} - I : t_1[i] = t_2[i]\} \end{array} \right\rangle$$

The following trivial lemma formally establishes the relation between independent sets of tuples and edge-free sets of nodes of a graph.

**Lemma IV.10.** *Let $S, t, I$ be such that $t \in S \subseteq T$ and $I \subseteq \{1, \ldots, n\}$. Let $\tilde{S}$ be a subset of $\mathsf{coincidents}(S, t, I)$ which is edge-free in $\mathsf{graph}(S, t, I)$. Then, $\tilde{S}$ is independent.*

In the proof of Lemma IV.12, the existence of an edge-free subset of nodes is concluded using, as a last ingredient, the following simple and well-known statement from graph theory, where $\mathsf{maxdegree}(G)$ denotes the maximum among all degrees of nodes of $G$:

**Lemma IV.11.** *Let $G = \langle V, E\rangle$ be an undirected graph. Let $u$ be a node of $G$. Then, there exists a subset $\tilde{V}$ of $V$ which is edge-free in $G$, includes $u$, and satisfies $|\tilde{V}| = \left\lceil \frac{|V|}{\mathsf{maxdegree}(G)+1} \right\rceil$.*

**Lemma IV.12.** *Let $K$ be a natural number. Let $S \subseteq T$ be a set of tuples which is not $K$-small. Let $t \in S$ be a tuple satisfying that $S - \{t\}$ is $K$-small.*

*Then, there exists an independent subset of tuples $\tilde{S} \subseteq S$ including $t$ and satisfying $|\tilde{S}| \geq K$.*

*Proof:* Since $S$ is not $K$-small, there exists a set $I \subsetneq \{1, \ldots, n\}$ and a tuple $t' \in S$ satisfying $|\mathsf{coincidents}(S, t', I)| \geq K^{n-|I|}(n - |I|)!$. Among all the possible $I$'s satisfying such a condition, we choose one maximal in size.

Note that $t$ belongs to $\mathsf{coincidents}(S, t', I)$ since, otherwise, $\mathsf{coincidents}(S, t', I) = \mathsf{coincidents}(S - \{t\}, t', I)$, and hence, $|\mathsf{coincidents}(S - \{t\}, t', I)| \geq K^{n-|I|}(n - |I|)!$, which implies that $S - \{t\}$ is not $K$-small, contradicting the assumptions of the lemma. Therefore, $\mathsf{coincidents}(S, t', I) = \mathsf{coincidents}(S, t, I)$ holds. In other words, the mentioned tuple $t'$ can be assumed to be $t$.

In the case $|I| = n - 1$, $|\mathsf{coincidents}(S, t, I)| \geq K^{n-(n-1)}(n - (n - 1))! = K$ holds. In this case, note that $\mathsf{coincidents}(S, t, I)$ itself is necessarily an independent set because its tuples coincide in all components but one, and hence they must be all pairwise different at such component. Thus, we conclude by defining $\tilde{S}$ as $\mathsf{coincidents}(S, t, I)$.

At this point, we assume $|I| < n-1$. Under this assumption, by the maximality selection of $I$, the following condition holds:

$$\begin{array}{l} \forall t' \in S : \forall I' \subsetneq \{1, \ldots, n\}, |I'| = |I| + 1 : \\ |\mathsf{coincidents}(S, t', I')| < K^{n-|I|-1}(n - |I| - 1)! \end{array}$$

Now, we analyse some properties of $G = \langle V, E\rangle = \mathsf{graph}(S, t, I)$. First, note that $|V| = |\mathsf{coincidents}(S, t, I)| \geq K^{n-|I|}(n - |I|)!$ holds, since $\mathsf{coincidents}(S, t, I)$ is the set of nodes of $G$. Second, we bound $\mathsf{maxdegree}(G)$ by bounding the degree of each node $t'$ of $G$ as follows:

$$\begin{array}{l} \mathsf{degree}(G, t') \\ \leq \sum_{i \in \{1,\ldots,n\}-I} \left(|\{t'' \in \mathsf{coincidents}(S, t, I) \mid t'[i] = t''[i]\}| - 1\right) \\ = \sum_{i \in \{1,\ldots,n\}-I} (|\mathsf{coincidents}(S, t', I \cup \{i\})| - 1) \\ < (n - |I|)K^{n-|I|-1}(n - |I| - 1)! - 1 \\ = K^{n-|I|-1}(n - |I|)! - 1 \end{array}$$

Therefore, $\mathsf{maxdegree}(G) < K^{n-|I|-1}(n - |I|)! - 1$. By Lemma IV.11, there exists a subset $\tilde{S}$ of $\mathsf{coincidents}(S, t, I)$ which is edge-free in $G$, includes $t$, and satisfies:

$$|\tilde{S}| = \left\lceil \frac{|V|}{\mathsf{maxdegree}(G) + 1} \right\rceil \geq \left\lceil \frac{K^{n-|I|}(n - |I|)!}{K^{n-|I|-1}(n - |I|)!} \right\rceil = K$$

By Lemma IV.10, it follows that $\tilde{S}$ is an independent set, and we are done. ∎

## C. Independent Sets of Terms

In this section, we simply translate the results of the previous one from tuples to terms and positions.

**Definition IV.13.** *Let $P$ be a set of positions. Let $p_1, \ldots, p_n$ be the positions in $P$, ordered lexicographically*[1]. *Let $t \in \mathcal{T}(\Sigma)$ be a term. Then, we define $\mathsf{Tuple}_P(t)$ as the tuple $\langle s_1, \ldots, s_n\rangle$, where each $s_i$ is $t|_{p_i}$ when $p_i$ is in $\mathsf{Pos}(t)$, and a special symbol $\perp$ not in $\Sigma$, otherwise.*

*Let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms. Then, we define $\mathsf{Tuples}_P(S)$ as $\{\mathsf{Tuple}_P(t) \mid t \in S\}$. We say that $S$ is $P$-independent if $\mathsf{Tuples}_P(S)$ is independent. Let $K$ be a natural*

[1]The concrete selected order for positions is not important at all, but we choose this one in order to fix a precise definition.

*number. We say that $S$ is $(K,P)$-small if $\mathsf{Tuples}_P(S)$ is $K$-small.*

In order to adapt the previous results we need to guarantee that there is a bijection between the set of terms $S$ and the set of tuples $\mathsf{Tuples}_P(S)$. In our concrete setting, this holds thanks to the fact that the considered set of positions always contains $\lambda$. Hence, consider a set of positions $P$ including $\lambda$ and a set of terms $\{t_1,\dots,t_m\} \subseteq \mathcal{T}(\Sigma)$ such that $P \cap \mathsf{Pos}(t_1) = \dots = P \cap \mathsf{Pos}(t_m)$. Note that in this case, $\{t_1,\dots,t_m\}$ is $P$-independent if and only if for each $p \in P$, either $p$ is not in any of the sets $\mathsf{Pos}(t_1),\dots,\mathsf{Pos}(t_m)$, or it is in all of them and either $t_1|_p = \dots = t_m|_p$ holds or the subterms $t_1|_p,\dots,t_m|_p$ are pairwise different.

The following two facts are straightforwardly implied by Lemmas IV.7 and IV.12, respectively.

**Lemma IV.14.** *Let $P$ be a set of positions including $\lambda$ and let $K$ be a natural number. Let $S \subseteq \mathcal{T}(\Sigma)$ be a non-empty $(K,P)$-small set of terms. Then, $|S| < K^{|P|}|P|!$ holds.*

**Lemma IV.15.** *Let $P$ be a set of positions including $\lambda$, let $K$ be a natural number, and let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms which is not $(K,P)$-small. Let $t \in S$ be a term satisfying that $S - \{t\}$ is $(K,P)$-small.*

*Then, there exists a $P$-independent set of terms $\tilde{S} \subseteq S$ including $t$ and satisfying $|\tilde{S}| \geq K$.*

### *D. Discarding Criterion*

We are now ready to define our criterion using the previously introduced background. Roughly speaking, it says that, when for a term $t$ reaching $q$ there exists a "big enough" set $S$ of terms reaching $q$, smaller than $t$ with respect to $\ll$, $\sim^A$-equivalent to $t$ and such that $S \cup \{t\}$ is $\mathsf{suff}(\Delta)$-independent, then $\langle t,q\rangle$ cannot be a piece of the minimum witness. The concrete size of $S$ is defined as follows.

**Definition IV.16.** *Let $A = \langle Q,\Sigma,F,\Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. We define $K(A)$ as $1 + |\mathsf{suff}(\Delta)|^3 \cdot (\tilde{h} + h(A) + 1)$.*

Lemma IV.18 formalizes the criterion that we use to discard feasible pairs. Its proof is based on the concept of *abstract position*, which we define as follows, for a given regular run $r$ on a term $t$ (see Definition III.3) and a position $p \in \mathsf{Pos}(t)$. It is a sequence of the form $q_1.q_2\dots q_n.\bar{p}$, where $q_1,\dots,q_n$ are states and $\bar{p}$ is a position. Intuitively, $q_1,\dots,q_n$ are the states found while traversing $r$ from the root to $p$, and $\bar{p}$ is the residual suffix of $p$ after the last state.

**Definition IV.17.** *Let $A = \langle Q,\Sigma,F,\Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. Let $r$ be a regular run of $A$ on a term $t \in \mathcal{T}(\Sigma)$. Let $p$ be a position in $\mathsf{Pos}(t)$. We define the* abstract position *of $p$ in $r$, denoted $\mathsf{abstract}_r(p)$, recursively as follows, where we explicitly write $r(\lambda)$ as a rule $l \to q$. For the case where $p$ is in $\mathsf{Pos}_\Sigma(l)$, $\mathsf{abstract}_r(p)$ is defined as $q.p$. For the case where $p$ is of the form $p_1.p_2$, where $p_1$ is a position in $\mathsf{Pos}_Q(l)$, $\mathsf{abstract}_r(p)$ is defined as $q.\mathsf{abstract}_{r|_{p_1}}(p_2)$. We write $\mathsf{abstract}(p)$ when $r$ is clear from the context.*

Note that, for a regular run $r$ on a term $t$ and a position $p \in \mathsf{Pos}(t)$, $r(p)$ is defined if and only if $\mathsf{abstract}(p)$ is of the form $q_1.q_2\dots q_n.\lambda$. Furthermore, if two positions $p_1,p_2 \in \mathsf{Pos}(t)$ satisfy $\mathsf{abstract}(p_1) = \mathsf{abstract}(p_2) = q_1.q_2\dots q_n.\lambda$ for some states $q_1,\dots,q_n$, then $r(p_1)$ and $r(p_2)$ are defined and the subruns $r|_{p_1}$ and $r|_{p_2}$ are equal. Intuitively, according to the definition of regular run of an $\mathtt{STA}_{\mathrm{hom}}$, for positions sharing the same sequence of states from the root, the corresponding regular subruns must coincide.

**Lemma IV.18.** *Let $A = \langle Q,\Sigma,F,\Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. Let $t,s_1,\dots,s_{K(A)}$ be different terms in $\mathcal{T}(\Sigma)$ with runs of $A$ on them reaching a state $q \in Q$, and satisfying that $\{t,s_1,\dots,s_{K(A)}\}$ is $\mathsf{suff}(\Delta)$-independent, $t \sim^A s_1,\dots,s_{K(A)}$, and $s_1,\dots,s_{K(A)} \ll t$.*

*Then, the pair $\langle t,q\rangle$ is not a piece of the minimum witness.*

*Sketch of the proof:* It suffices to prove that if $\langle t,q\rangle$ is a piece of a witness $\langle w,q_f\rangle$, then this witness is not minimum. Let $r$ be a regular run on $w$, and let $p$ be a position such that $r|_p$ is a run of $A$ on $t$ reaching $q$. Let $q_1.\dots.q_n.\lambda$ be $\mathsf{abstract}(p)$. Let $p_1,\dots,p_m \in \mathsf{Pos}(w)$ be the positions with the same abstract position $q_1.\dots.q_n.\lambda$, i.e. $\{p_1,\dots,p_m\} = \mathsf{abstract}^{-1}(\mathsf{abstract}(p))$. By the definition of $\mathtt{STA}_{\mathrm{hom}}$, the terms $w_{s_i} = w[s_i]_{p_1}\dots[s_i]_{p_m}$ have also runs reaching $q_f$. For one of such $s_i$'s, if no more rules in $\tilde{h}$-runs on $w_{s_i}$ can be applied above the positions $p_1,\dots,p_m$, with respect to the rules applicable in $w$, then such $\langle w_{s_i},q_f\rangle$ is a smaller witness. Since $t \sim^A s_1,\dots,s_{K(A)}$ holds, a new rule can be applied only if some equality which is not satisfied with $\tilde{h}$-runs on $w$ is satisfied with $\tilde{h}$-runs on $w_{s_i}$. Since $\{t,s_1,\dots,s_{K(A)}\}$ is $\mathsf{suff}(\Delta)$-independent and $t \sim^A_{=,\mathsf{ht}} s_1,\dots,s_{K(A)}$, each equality becomes true due to at most one of such $s_i$'s: if two $s_i,s_j$ made true the same equality, then they should share the same subterm at the same relative position $p'$, and $t$ should have a different subterm at $p'$, thus contradicting the definition of $\mathsf{suff}(\Delta)$-independence. Thus, it suffices to bound exponentially (by $K(A)$) the number of equalities that might become true with $\tilde{h}$-runs. Actually, this number is much bigger, but we argue that many of them are essentially the same. Consider an equality $\bar{p}_1 = \bar{p}_2$ tested at a position $\bar{p}$. Let $q_1.\dots.q_i.p'$ be $\mathsf{abstract}(\bar{p})$. Note that such $q_1.\dots.q_i$ is a prefix of the above defined $q_1.\dots.q_n$, since an equality that becomes true must be tested above a position which has changed. For any other position $\hat{p}$ with the same abstract position $q_1.\dots.q_i.p'$, the equality $\bar{p}_1 = \bar{p}_2$ becomes true with $\tilde{h}$-runs at $\hat{p}$ if and only if it becomes true with $\tilde{h}$-runs at $\bar{p}$. Thus, we can consider both equalities as essentially the same. Moreover, if $q_1.\dots.q_i.p'$ is an abstract position of a position $\bar{p}$ where an equality becomes true with $\tilde{h}$-runs, then $n-i$ is bounded by $\tilde{h}+h(A)$: otherwise, if $n-i > \tilde{h}+h(A)$, then the path from $\bar{p}$ to the replaced position passes through the sequence of states $q_i.\dots.q_n$, which is larger than $\tilde{h}+h(A)$, thus contradicting the definition of $\tilde{h}$-run, which imposes that two terms that satisfy an equality must have height bounded by $\tilde{h}$.

In summary, two $w_{s_i}$'s cannot satisfy equalities not satisfied in $w$ and tested at positions with the same corresponding abstract position, and the number of such possible abstract positions is bounded by the number of sequences $q_1 . \dots . q_i . p'$ where $n - i$ is bounded by $\tilde{h} + h(A)$ and $p'$ belongs to $\mathsf{suff}(\Delta)$. The fact that the number of possible $p'$'s is bounded by $|\mathsf{suff}(\Delta)|$ and that the number of different possible tested equalities is bounded by $|\mathsf{suff}(\Delta)|^2$, justifies the definition of $K(A)$ as $1 + |\mathsf{suff}(\Delta)|^3 \cdot (\tilde{h} + h(A) + 1)$, since at least one $s_i$ in $\{s_1, \dots, s_{K(A)}\}$ does not satisfy new equalities, and hence $\langle w_{s_i}, q_f\rangle$ is a witness. ∎

Note that the simple definition of $\mathtt{STA}_{\mathrm{hom}}$ based on implicit equality constraints represented by identity of states is key in the above proof to obtain simple arguments. For a detailed proof of Lemma IV.18 see [6].

## V. The Algorithm

In this section we introduce an algorithm that decides HOM in exponential time. The algorithm iteratively generates feasible pairs term-state of an $\mathtt{STA}_{\mathrm{hom}}$ $A$ with the aim of finding a witness of the non-regularity of $A$.

Two sets of feasible pairs, called Definitive and Candidates are maintained as data structures of the algorithm. Initially, Definitive is empty, and Candidates has all the feasible pairs $\langle t, q\rangle$ such that a run with only one applied rule on $t$ reaches $q$. At each iteration, the minimum pair $\langle t, q\rangle$ with respect to $\ll$ in Candidates is considered. The pair $\langle t, q\rangle$ is added to the set Definitive unless it is realized that it is not a piece of the minimum witness. This fact can be detected using the concepts of the previous section. According to Lemma IV.18, in order to discard the addition of $\langle t, q\rangle$ to Definitive, we should consider the set $\{s \mid \langle s, q\rangle \in \mathsf{Definitive} \wedge s \sim^A t\}$ and look inside it to decide the existence of a subset $\{s_1, \dots, s_{K(A)}\}$ such that $\{s_1, \dots, s_{K(A)}, t\}$ is $\mathsf{suff}(\Delta)$-independent. This task itself is too hard. But Section IV-C gives us an alternative criterion to determine, in some cases, that such a subset exists. Along the execution of the algorithm we preserve an invariant stating that each of such sets $\{s \mid \langle s, q\rangle \in \mathsf{Definitive} \wedge s \sim^A t\}$ is a $(K(A)\text{+}1, \mathsf{suff}(\Delta))$-small set of terms. If the addition of $t$ to this set makes it non-$(K(A)\text{+}1, \mathsf{suff}(\Delta))$-small, then, by Lemma IV.15, it follows the existence of the subset $\{s_1, \dots, s_{K(A)}\}$ mentioned above. Thus, in this case we must discard the pair $\langle t, q\rangle$, since it is not a piece of the minimum witness.

In the case where the pair $\langle t, q\rangle$ is not discarded, it is added to the set Definitive and used to generate new feasible pairs term-state, which are added to Candidates. This generation is performed (i) using the left-hand sides of rules in $\Delta$ for determining the symbols in the top-most positions of the new terms, (ii) using the pairs term-state in Definitive to instantiate the states appearing in such left-hand sides, and also (iii) guaranteeing that the specific pair $\langle t, q\rangle$ is used for the instantiation. This last condition ensures that all the pairs added to Candidates are new, i.e. that the algorithm has still not considered them to be added to Definitive (although they may be already in Candidates due to a previous generation). This generation is defined formally as follows:

**Definition V.1.** *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. Let $S \subseteq \mathcal{T}(\Sigma) \times Q$ be a set of feasible pairs term-state. Let $\langle t, q\rangle \in S$ be a feasible pair term-state. We define the* set of instantiations of $\Delta$ with $S$ and $\langle t, q\rangle$ *as the set of feasible pairs:*

$$\begin{aligned}\{\langle l[t_1]_{p_1} \dots [t_n]_{p_n}, q'\rangle \mid \\ &\exists (l \to q') \in \Delta, \{p_1, \dots, p_n\} = \mathsf{Pos}_Q(l), \\ &\forall i, j \in \{1, \dots, n\} : l(p_i) = l(p_j) \Rightarrow t_i = t_j, \\ &\forall i \in \{1, \dots, n\} : \langle t_i, l(p_i)\rangle \in S, \\ &\exists i \in \{1, \dots, n\} : \langle t_i, l(p_i)\rangle = \langle t, q\rangle\}\end{aligned}$$

When there are no more pairs in Candidates to be considered, the algorithm stops and states non-regularity if there is a witness of non-regularity in Definitive. We present in Algorithm V.1 a formalization of the previous explanations.

**Algorithm V.1** to decide regularity of the language recognized by an $\mathtt{STA}_{\mathrm{hom}}$ $A$.

```
Input: an $\mathtt{STA}_{\mathrm{hom}}$ $A = \langle Q, \Sigma, F, \Delta\rangle$.
Data structures: Definitive, Candidates sets of elements in $\mathcal{T}(\Sigma) \times Q$.
(1) Insert in Candidates the pairs $\langle l, q\rangle$ such that $(l \to q) \in \Delta$ and $l \in \mathcal{T}(\Sigma)$.
(2) While Candidates is not empty:
    (a) Let $\langle t, q\rangle$ be the smallest pair in Candidates with respect to $\ll$.
    (b) Remove $\langle t, q\rangle$ from Candidates.
    (c) If $\{s \mid \langle s, q\rangle \in$ Definitive $\wedge\ s \sim^A t\} \cup \{t\}$ is $(K(A)$+1, suff$(\Delta))$-small:
        (i) Insert $\langle t, q\rangle$ in Definitive.
        (ii) Insert in Candidates all the elements in the set of instantiations of $\Delta$ with Definitive and $\langle t, q\rangle$.
(3) If there exists a witness in Definitive, then output 'NON-REGULAR', else output 'REGULAR'.
```

The following theorem states that our algorithm runs in exponential time. It is a consequence of Lemmas IV.3 and IV.14, and Definition IV.16, and using DAG's as the internal representation for terms. For a detailed proof see [6].

**Theorem V.2.** *Algorithm V.1 takes exponential time in the worst case.*

We now focus on proving that Algorithm V.1 is correct. We start in the next lemma stating the following trivial property of our algorithm: whenever a pair is not generated, it is due to having previously discarded another pair needed for its construction.

**Lemma V.3.** *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be an $\mathtt{STA}_{\mathrm{hom}}$. Let $\langle t, q\rangle \in \mathcal{T}(\Sigma) \times Q$ be a feasible pair term-state of $A$ satisfying that it is not generated by Algorithm V.1 on input $A$.*

*Then, there exists a feasible pair $\langle t', q'\rangle \in \mathcal{T}(\Sigma) \times Q$ of $A$ such that it is a piece of $\langle t, q\rangle$ and is discarded by Algorithm V.1 on input $A$.*

We are now ready to finally prove the soundness and completeness of Algorithm V.1.

**Theorem V.4.** *Let $A$ be an $\mathtt{STA}_{\mathrm{hom}}$. Then, Algorithm V.1 on input $A$ outputs* 'REGULAR' *if and only if $\mathcal{L}(A)$ is regular.*

*Proof:* Let $A$ be $\langle Q, \Sigma, F, \Delta \rangle$ more explicitly written.

The right-to-left direction follows trivially, since in this case $\mathcal{L}(A) \subseteq \mathcal{L}_{\tilde{h}}(A)$ holds by Corollary III.10, and hence it is not possible to generate a feasible pair $\langle w, q \rangle \in \mathcal{T}(\Sigma) \times Q$ of $A$ satisfying that $q \in F$ and $w \notin \mathcal{L}_{\tilde{h}}(A)$, i.e. it is impossible to find a witness.

For the other direction, note that the algorithm outputs '$\texttt{REGULAR}$' when it cannot find a witness. We need to prove that in this case no witness exists, and hence, by Corollary III.10, $\mathcal{L}(A)$ is regular. We proceed by contradiction by assuming that there exists a witness, but the algorithm cannot find any witnesses and outputs '$\texttt{REGULAR}$'. Let $\langle w, q \rangle \in \mathcal{T}(\Sigma) \times Q$ be the minimum witness of $A$ with respect to $\ll$. By assumption, $\langle w, q \rangle$ is either discarded or not generated by the algorithm. The former case is not possible, since it implies that Definitive contains at least one pair $\langle w', q \rangle$ such that $w' \sim^A w$, and therefore $\langle w', q \rangle$ is a witness, contradicting the fact that the algorithm finds no witness. Hence, assume that $\langle w, q \rangle$ is not generated by the algorithm. By Lemma V.3, it follows that there exists a feasible pair $\langle t', q' \rangle$ of $A$ such that it is a piece of $\langle w, q \rangle$ and is discarded by the algorithm. Consider that the execution of the algorithm is at the iteration when the pair $\langle t', q' \rangle$ is discarded and let $S$ be $\{s \mid \langle s, q' \rangle \in \text{Definitive} \wedge s \sim^A t'\} \cup \{t'\}$. We know that $S - \{t'\}$ is $(K(A)+1, \mathsf{suff}(\Delta))$-small, but $S$ is not. Hence, by Lemma IV.15, it follows that there exists a $\mathsf{suff}(\Delta)$-independent set of terms $\tilde{S} \subseteq S$ including $t'$ and satisfying $|\tilde{S}| \geq K(A) + 1$. By definition, all the terms in $\tilde{S}$ have runs of $A$ on them reaching the state $q'$, are $\sim^A$-equivalent to $t'$, and smaller than $t'$ with respect to $\ll$. By Lemma IV.18, it follows that $\langle t', q' \rangle$ is not a piece of the minimum witness, contradicting the selection of $\langle w, q \rangle$. ■

The next corollary trivially follows from Theorems V.2 and V.4 and the fact that HOM in known to be EXPTIME-hard [13].

**Corollary V.5.** *The HOM problem is EXPTIME-complete.*

## VI. Conclusions

In [15], HOM was proved decidable in triple exponential time. Using some intermediate results of that paper, we have classified this problem as EXPTIME-complete. The alternative presentation using $\texttt{STA}_{\text{hom}}$ instead of $\texttt{TA}_{\text{hom}}$ has been useful to obtain more accessible definitions and proofs. It should be studied whether adopting this alternative formalism leads to simpler proofs also in [15]. Moreover, in [15], [14], other interesting problems were shown decidable in triple exponential time, like emptiness and finiteness of languages described as images of regular tree languages through tree homomorphisms, and inclusion of such languages. It would be interesting to study whether such other problems can be solved with a better time complexity using the techniques presented here.

Our exponential time algorithm is based on the techniques used in [5] for proving EXPTIME-completeness of the emptiness problem for tree automata with disequality constraints. Nevertheless, we have obtained simpler arguments in some parts. It would be interesting to try readapting them to this kind of automata in order to obtain simpler proofs, too.

## Acknowledgment

The authors were supported by Spanish Ministry of Education and Science by the FORMALISM project (TIN2007-66523). The first author was also supported by an FPI-UPC grant and by the FI-DGR grant from the AGAUR. The second author was also supported by an FPU grant from the Spanish Ministry of Education. The third author was also supported by the Spanish Ministry of Science and Innovation SweetLogics project (TIN2010-21062-C02-01).

## References

[1] F. Baader and T. Nipkow, *Term Rewriting and All That*. New York: Cambridge University Press, 1998.

[2] P. B. Bendix and D. E. Knuth, "Simple word problems in universal algebras," in *Computational Problems in Abstract Algebra*. Pergamon Press, 1970, pp. 263–297.

[3] B. Bogaert, F. Seynhaeve, and S. Tison, "The recognizability problem for tree automata with comparison between brothers," in *Foundations of Software Science and Computation Structures (FOSSACS)*, 1999, pp. 150–164.

[4] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, "Tree automata techniques and applications," Available at http://www.grappa.univ-lille3.fr/tata, 2007.

[5] H. Comon and F. Jacquemard, "Ground reducibility is EXPTIME-complete," *Information and Computation*, vol. 187, no. 1, pp. 123–153, 2003.

[6] C. Creus, A. Gascón, G. Godoy, and L. Ramos, "The HOM problem is EXPTIME-complete," Available on: www.lsi.upc.es/~ggodoy/publications.html, 2012, (long version).

[7] M. Dauchet, S. Tison, and M. Tommasi, "Reduction de la non-linearite des morphismes d'arbres recognizable tree-languages and non-linear morphisms," *Theoretical Computer Science*, vol. 281, no. 1-2, pp. 219–233, 2002.

[8] J. Engelfriet, "Bottom-up and top-down tree transformations - a comparison," *Mathematical Systems Theory*, vol. 9, no. 3, pp. 198–231, 1975.

[9] E. Filiot, J.-M. Talbot, and S. Tison, "Tree automata with global constraints," in *12th International Conference in Developments in Language Theory (DLT 2008)*, ser. Lecture Notes in Computer Science, vol. 5257. Springer, 2008, pp. 314–326.

[10] Z. Fülöp, "Undecidable properties of deterministic top-down tree transducers," *Theoretical Computer Science*, vol. 134, pp. 311–328, 1994.

[11] F. Gécseg and M. Steinby, *Tree Automata*. Akadémiai Kiadó, 1984.

[12] F. Gécseg and M. Steinby, "Tree languages," in *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Springer-Verlag, 1997, vol. 3, pp. 1–68.

[13] O. Giménez, G. Godoy, and S. Maneth, "Deciding regularity of the set of instances of a set of terms with regular constraints is EXPTIME-complete," *SIAM Journal on Computing*, vol. 40, no. 2, pp. 446–464, 2011.

[14] G. Godoy and O. Giménez, "The HOM problem is decidable," 2011, submitted to a journal.

[15] G. Godoy, O. Giménez, L. Ramos, and C. Àlvarez, "The HOM problem is decidable," in *Symposium on Theory of Computing (STOC)*, 2010, pp. 485–494.

[16] G. Godoy, S. Maneth, and S. Tison, "Classes of tree homomorphisms with decidable preservation of regularity," in *Foundations of Software Science and Computation Structures (FOSSACS)*, 2008, pp. 127–141.

[17] D. Hofbauer and M. Huber, "Computing linearizations using test sets," in *Third International Workshop on Conditional Term Rewriting Systems (CTRS)*, 1992, pp. 287–301.

[18] G. Kucherov and M. Tajine, "Decidability of regularity and related properties of ground normal form languages," *Information and Compuation*, vol. 118, pp. 91–100, 1995.

[19] J. W. Thatcher, "Transformations and translations from the point of view of generalized finite automata theory," in *Symposium on Theory of Computing (STOC)*, 1969, pp. 129–142.

[20] S. Vágvölgyi and R. Gilleron, "For a rewrite system it is decidable whether the set of irreducible, ground terms is regognizable," *Bulletin of the EATCS*, vol. 48, pp. 197–209, 1992.