SIAM J. COMPUT.
Vol. 45, No. 4, pp. 1230–1260

THE HOM PROBLEM IS EXPTIME-COMPLETE*

CARLES CREUS†, ADRIÀ GASCÓN‡, GUILLEM GODOY†, AND LANDER RAMOS†

Abstract. We define a new class of tree automata with constraints and prove decidability of the emptiness problem for this class in exponential time. As a consequence, we obtain several EXPTIME-completeness results for problems on images of regular tree languages under tree homomorphisms, like set inclusion, regularity (HOM problem), and finiteness of set difference. Our result also has implications in term rewriting, since the set of reducible terms of a term rewrite system can be described as the image of a tree homomorphism. In particular, we prove that inclusion of sets of normal forms of term rewrite systems can be decided in exponential time. Analogous consequences arise in the context of XML typechecking, since types are defined by tree automata and some type transformations are homomorphic.

Key words. homomorphisms, regular languages, transducers, tree automata

AMS subject classifications. 68Q17, 68Q42, 68Q45

DOI. 10.1137/140999104

1. Introduction.

Tree automata [7, 17] are a well studied formalism for representing term languages. They characterize the "regular term languages," a classical concept used, e.g., to describe the parse trees of a context-free grammar or the well-formed terms over a sorted signature [25], to characterize the solutions of formulas in monadic second-order logic [12], and to naturally capture type formalisms for tree-structured XML data [4, 26]. Unfortunately, the expressiveness of plain tree automata is rather limited, and hence, several extensions have been considered in the literature. For example, tree automata with local disequality constraints allowed proving EXPTIME-completeness of ground reducibility [8], tree automata with constraints between brothers allowed proving decidability of fragments of quantifier-free formulas on one-step rewriting [6], tree set automata allowed generalizing results on decidability of set constraints [19], and tree automata with global constraints allowed proving decidability of extensions of monadic second order logic [2, 14].

In this paper we define a new variant of tree automata with constraints specially suited to deal with problems on tree homomorphisms applied to regular tree languages. A tree homomorphism can be defined by equations of the form $H(f(x_1, \ldots, x_n)) =$

*Received by the editors December 5, 2014; accepted for publication (in revised form) April 28, 2016; published electronically July 6, 2016. This work was supported by funds from the Spanish Ministry of Economy and Competitiveness (MINECO) and the European Union (FEDER funds) under grant COMMAS (TIN2013-46181-C2-1-R) and by funds from the Spanish Ministry of Science and Innovation (MICINN) under grant FORMALISM (TIN2007-66523) and by the SweetLogics project (TIN2010-21062-C02-01). A preliminary version of this work appeared in [9], where we develop specific techniques to prove decidability of the HOM problem in exponential time. Here, we present a more general framework and obtain further results based on the new class of tree automata.
http://www.siam.org/journals/sicomp/45-4/99910.html

†Computer Science Department at Universitat Politècnica de Catalunya, Jordi Girona 1-3, Barcelona 08034, Spain (ccreuslopez@gmail.com, ggodoy@lsi.upc.edu, lander.ramos@upc.edu). The first author was supported by an FPU grant from the Spanish Ministry of Education. The last author was supported by an FPI grant from the Spanish Ministry of Education.

‡University of Edinburgh, 10 Crichton Street, Edinburgh, United Kingdom, EH8 9AB (agascon@inf.ed.ac.uk). The work of this author was supported under SOCIAM: The Theory and Practice of Social Machines. The SOCIAM Project is funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/J017728/1 and comprises the Universities of Oxford, Southampton, and Edinburgh.

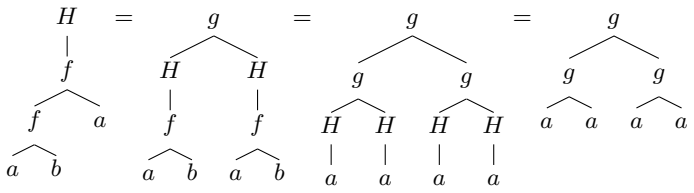

FIG. 1. *Recursive application of the tree homomorphism H to compute $H(f(f(a,b),a))$.*

t, where t is a tree labeled by either alphabet symbols, or by $H(x_1),\ldots,H(x_n)$, which may only appear at the leaves. For example, the image of the set of terms over binary f and nullaries a,b by the tree homomorphism H defined by $H(a) = a,\ H(b) = a,\ H(f(x_1,x_2)) = g(H(x_1),H(x_1))$ 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. In general, nonlinear tree homomorphisms like H (due to the duplication of the variable x_1) may produce nonregular images. The homomorphism (HOM) problem questions, for a given regular tree language L described by a tree automaton and a tree homomorphism H, whether $H(L)$ is regular.

The study of preservation of tree regularity by tree homomorphisms was introduced in [27], and many partial or related results have been obtained since then [5, 11, 13, 15, 20, 22, 23, 24, 28]. HOM has recently been proved decidable in [21], together with other problems related to images of regular tree languages under tree homomorphisms, like set inclusion and finiteness of set difference. The obtained time complexity is triple exponential for all of them. In the present paper we extend the work done in [21] by proving that all such problems are, in fact, EXPTIME-complete. To this end, we define a new class of tree automata with constraints called *tree automata with disequality and implicit HOM equality constraints* ($\mathtt{TA}_{\mathrm{ihom},\not\approx}$). $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ allow arbitrary local disequality constraints and a particular kind of equality constraints: the left-hand side of rules are terms containing states at some leaf positions, and two positions with the same state implicitly define a local equality constraint between such positions. Note that, in particular, equality constraints between nonbrother positions can be defined, as the left-hand side of the rules are not required to be flat terms. We prove that the emptiness of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ can be decided in exponential time with respect to the number of different positions involved in the rules. As a consequence, since the regularity (HOM problem), inclusion, and finiteness of the set difference of images of regular tree languages under tree homomorphisms can be reduced to the emptiness of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ in exponential time, but preserving the positions involved in the rules, the EXPTIME-completeness of all such problems follows. Decidability of emptiness for $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ in exponential time also has implications in term rewriting, since the set of reducible terms of a term rewrite system can be described as the image of a tree homomorphism. In particular, we prove that inclusion of sets of normal forms of term rewrite systems can be decided in exponential time. Analogous consequences arise in the context of XML typechecking, since types can be defined by tree automata and some type transformations are homomorphic.

1.1. Approach to decide emptiness of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ in exponential time. Tree automata with disequality constraints ($\mathtt{TA}_{\not\approx}$) are a particular case of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ where only disequality constraints are allowed. Emptiness of $\mathtt{TA}_{\not\approx}$ was proved decidable in exponential time in [8]. In the present paper, we generalize the approach used

in [8] to $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Roughly, the algorithm deciding emptiness of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ looks for an accepting run by iteratively generating all possible runs in increasing order of size: new runs are constructed using the previous runs as direct subruns. In order to guarantee termination, some runs are discarded when the algorithm realizes that they cannot be subruns of the minimum accepting run. Below we briefly describe the involved discarding criterion.

In order to determine that a run r' cannot be a subrun of the minimum accepting run, we prove that any run r having r' as a subrun can be transformed into a smaller run reaching the same state. Due to the equality constraints, such a transformation requires replacing several identical subruns of r by other smaller subruns. The fact that equalities of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ are implicitly forced by identity of states provides a great advantage to reason about such replacements: two positions p_1, p_2 are forced by the automaton to have the same pending term whenever the same sequence of states $q_1, q_2, \ldots, q_n$ is found while traversing the run from the root to p_1 and p_2. This leads us to introduce the concept of abstract position as a string of states $q_1 q_2 \ldots q_n$ that implicitly represents the set of all positions that find $q_1, q_2, \ldots, q_n$ while traversing the run from the root to them. We adapt typical notation of subterm location and replacement to abstract positions. For example, $r|_{q_1 q_2 \ldots q_n}$ denotes the subrun pending at any of the positions referred by $q_1 q_2 \ldots q_n$, and $r[r']_{q_1 q_2 \ldots q_n}$ denotes the simultaneous replacement of all such subruns by a new subrun r'. It is easy to satisfy the implicit equality constraints using such replacements, but in general the result is not a correct run since the disequality constraints of a rule applied above the replaced subruns might become unsatisfied. We extend an argument from [8] to prove that, for each $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A, there exists a natural number K exponentially bounded by $|A|$ satisfying the following property: given a run r, an abstract position $P = q_1 \ldots q_n$, and runs $r_1, \ldots, r_K$ smaller than $r|_P$, if all the replacements $r[r_1]_P, \ldots, r[r_K]_P$ falsify different disequalities, then there exists a prefix P' of P and a run r' smaller than $r|_{P'}$ such that $r[r']_{P'}$ does not falsify any disequality.

In general, finding such runs $r_1, \ldots, r_K$ satisfying the above conditions is not an easy task. The following notion of independence, given in [8], helps to overcome this problem: the runs $r_1, \ldots, r_K$ on terms $t_1, \ldots, t_K$ are independent with respect to a set of positions $\mathcal{P}$ if, for each position $p \in \mathcal{P}$, either all the terms $t_1|_p, \ldots, t_K|_p$ are identical, or they are pairwise different. We prove that it is safe to discard a new generated run $\bar{r}$ if there exist certain $r_1, \ldots, r_K$ chosen among the previously generated runs and such that $r_1, \ldots, r_K, \bar{r}$ are independent (with respect to the set of positions $\mathcal{P}$ that can be affected by constraints), under certain additional conditions. To efficiently detect those $r_1, \ldots, r_K$, we define a new notion on sets of runs and natural numbers K, namely, K-smallness. With this notion, when a set S is K-small, but the addition of a new run $\bar{r}$ gives rise to a non-K-small set $S \cup \{\bar{r}\}$, it follows the existence of an independent subset $\tilde{S}$ of $S \cup \{\bar{r}\}$ with size K and containing $\bar{r}$.

1.2. Organization of the paper. In section 2 we introduce basic notation and concepts related to terms, tree automata, and tree homomorphisms. In section 3 we define $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ and recall some results of [21]. In section 4 we introduce the notions of independence and K-smallness and the corresponding results. In section 5 we define abstract positions and replacements and analyze disequality constraints. In section 6, we give the algorithm deciding emptiness of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ in exponential time. In section 7 we show the consequences of this result by reducing other problems to it. In particular, we prove that HOM is EXPTIME-complete. In section 8 we conclude.

2. Preliminaries.

2.1. Terms. In this section we introduce notation for terms, positions, replacements, substitutions, and rewrite rules. For a survey see [1].

The size of a finite set S is denoted by $|S|$, and the powerset of S is denoted by 2^S. A *signature* consists of an alphabet Σ, i.e., a finite set of symbols, together with a mapping that assigns to each symbol in Σ a natural number, its *arity*. We denote by $\mathsf{arity}(f)$ the arity of symbol f, by $\Sigma^{(m)}$ the subset of symbols in Σ of arity m, and by $\mathsf{maxarity}(\Sigma)$ the maximum m such that $\Sigma^{(m)}$ is not empty. The *set of all terms over* Σ 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 Σ with variables in $\mathcal{X}$, denoted $\mathcal{T}(\Sigma, \mathcal{X})$, 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 $|f(t_1, \ldots, t_m)| = 1 + |t_1| + \cdots + |t_m|$ for each $f \in \Sigma^{(m)}$, $m \geq 0$, and $t_1, \ldots, t_m \in \mathcal{T}(\Sigma, \mathcal{X})$, and as $|x| = 1$ for each $x \in \mathcal{X}$. 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 $t_1, \ldots, t_m \in \mathcal{T}(\Sigma, \mathcal{X})$, as $\mathsf{height}(a) = 0$ for each $a \in \Sigma^{(0)}$, and as $\mathsf{height}(x) = 0$ for each $x \in \mathcal{X}$. A term t is called flat if its height is at most 1. Positions in terms are sequences of natural numbers. Given a term $t = f(t_1, \ldots, t_m) \in \mathcal{T}(\Sigma, \mathcal{X})$, its set of positions $\mathsf{Pos}(t)$ is defined recursively as $\{\lambda\} \cup \{i.p \mid i \in \{1, \ldots, m\} \wedge p \in \mathsf{Pos}(t_i)\}$. Here, λ 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 $p \in \mathsf{Pos}(t)$ 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 t, s and position $p \in \mathsf{Pos}(t)$, we denote by $t[s]_p$ the result of replacing the subterm at position p in t by the term s. More formally, $t[s]_\lambda$ is s and $f(t_1, \ldots, t_m)[s]_{i.p}$ is $f(t_1, \ldots, t_{i-1}, t_i[s]_p, t_{i+1}, \ldots, t_m)$. The symbol of t occurring at position p is denoted by $t(p)$. More formally, $f(t_1, \ldots, t_m)(\lambda) = f$ and $f(t_1, \ldots, t_m)(i.p) = t_i(p)$. We say that t at position p is labeled by f if $f = t(p)$. For a set Γ, we use $\mathsf{Pos}_\Gamma(t)$ to denote the set of positions of t that are labeled by symbols in Γ. 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 λ, then we say that p_1 is a strict 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* σ 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 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. A *rewrite rule* is a pair of terms $l \to r$. The application of a rewrite rule $l \to r$ to a term $s[\sigma(l)]_p$ at position p produces the term $s[\sigma(r)]_p$. If R is a set of rewrite rules, the application of a rule of R to a term s resulting in a term t is denoted by $s \to_R t$, and the reflexive-transitive closure of this relation is denoted by $\to_R^*$.

2.2. Tree automata. Tree automata and regular tree languages are well-known concepts of theoretical computer science [16, 17, 7]. 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 2.1. *A* tree automaton (TA) *is a tuple* $A = \langle Q, \Sigma, F, \Delta \rangle$*, where* Q *is a finite set of states,* Σ *is a signature,* $F \subseteq Q$ *is the subset of final states, and* Δ *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* Δ*, 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* $t(p) \in \Sigma^{(m)}$*, then* $r(p)$ *is a rule of the form* $t(p)(q_1, \ldots, q_m) \to q$*, 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* $\mathcal{L}(A)$*. A language* L *is* regular *if there exists a* TA A *such that* $\mathcal{L}(A) = L$*.*

2.3. Tree homomorphisms.

DEFINITION 2.2. *Let* Σ_1, Σ_2 *be two signatures. A* tree homomorphism *is a function* $H : \mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ *that can be defined as follows. Let* $\mathcal{X}_m$ *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* $\mathcal{T}(\Sigma_2, \mathcal{X}_m)$*. After that,* $H(f(t_1, \ldots, t_m))$ *is defined recursively, 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 2.3. *The HOM problem is defined as follows:*
Input: *A* TA A *and a tree homomorphism* H*.*
Question: *Is* $H(\mathcal{L}(A))$ *regular?*

2.4. Hardness results. We list some known hardness results that are related to our setting.

PROPOSITION 2.4. *The following problems are EXPTIME-hard:*

- *The HOM problem (from* [20]*).*
- *The problems of deciding equivalence and inclusion between the languages recognized by two* TA *(since universality of* TA *is EXPTIME-hard and can be reduced to equivalence and to inclusion* [7]*) and finiteness of their difference (since inclusion can be reduced to finite difference).*

3. Tree automata with disequality and implicit HOM equality constraints. Decidability of HOM is tackled in [21] by reasoning on a new kind of automata, which is obtained by a straightforward application of a homomorphism to the rules of a TA (and adding equality and disequality constraints). Its main disadvantage is that dealing with constrained rules makes the presentation of technical proofs a laborious task. For this reason, we define an equivalent kind of automata in which equality constraints are implicitly encoded in the left-hand sides of the rules. Intuitively, a state that appears duplicated in the left-hand side l of a rule implicitly forces an equality test between all the positions of l where such state occurs.

DEFINITION 3.1. *A* tree automaton with disequality and implicit HOM equality constraints ($\mathtt{TA}_{\mathrm{ihom},\not\approx}$) *is a tuple* $A = \langle Q, \Sigma, F, \Delta \rangle$*, where* Q *is a finite set of states,* Σ *is a signature,* $F \subseteq Q$ *is the subset of final states, and* Δ *is a finite 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 nullary*

symbols, and c, called the disequality constraint of the rule, is a conjunction/set of unordered pairs of the form $\bar{p}_1 \not\approx \bar{p}_2$ for arbitrary positions $\bar{p}_1, \bar{p}_2$. When all the rules of A have an empty disequality constraint, we say that A is a $\mathtt{TA}_{\mathrm{ihom}}$.

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 $l(p')$ 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 Σ, then $l(p') = t(p.p')$. In addition,* (i) *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, and* (ii) *for each two positions $\bar{p}_1, \bar{p}_2$ satisfying $(\bar{p}_1 \not\approx \bar{p}_2) \in c$, either one of $p.\bar{p}_1, p.\bar{p}_2$ is not in $\mathsf{Pos}(t)$ or both of them are and $t|_{p.\bar{p}_1} \neq t|_{p.\bar{p}_2}$ holds. We say that r is a* weak run *when conditions* (i) *and* (ii) *are not enforced. Moreover, since t can be deduced from r, we often do not make explicit t and just say that r is a (weak) run of A.*

The state reached *by a weak run r is the right-hand side of $r(\lambda)$, and we say that 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. Given a weak run r of A on a term t, we define $\mathsf{Pos}(r)$ as $\mathsf{Pos}(t)$, $\mathsf{height}(r)$ as $\mathsf{height}(t)$, and $\mathsf{term}(r)$ as t. Moreover, given a position p such that $r(p)$ is defined, we define the weak* subrun *$r|_p$ as the weak run of A on $t|_p$ described by $r|_p(p') = r(p.p')$. Note that if r is a run, then so is $r|_p$. Given two weak runs r_1, r_2 and a position p such that $r_1(p)$ is defined and $r_1|_p, r_2$ reach the same state, we define the* replacement *$r_1[r_2]_p$ as the weak run r on $\mathsf{term}(r_1)[\mathsf{term}(r_2)]_p$ described as follows: $r(p') = r_2(\hat{p})$ if p' is of the form $p.\hat{p}$, and $r(p') = r_1(p')$ otherwise.*

The size of A, denoted $|A|$, is $|Q|$ plus the sum of sizes of all rules in Δ, 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. By $\mathsf{n}_{\not\approx}(A)$ we denote the number of distinct disequality atoms in the rules of A. By $\mathsf{h}_{\not\approx}(A)$ we denote the maximum among the lengths of the positions occurring in disequality atoms in the rules of A. By $\mathsf{Pos}_{\mathrm{lhs}}(A)$ we denote the set of positions of left-hand sides of rules of A, i.e., $\bigcup_{(l \xrightarrow{c} q) \in \Delta} \mathsf{Pos}(l)$. By $\mathsf{h}_{\mathrm{lhs}}(A)$ we denote the maximum among the heights of the left-hand sides of the rules of A, i.e., $\max\{|p| \mid p \in \mathsf{Pos}_{\mathrm{lhs}}(A)\}$. We just write $\mathsf{n}_{\not\approx}$, $\mathsf{h}_{\not\approx}$, $\mathsf{Pos}_{\mathrm{lhs}}$, $\mathsf{h}_{\mathrm{lhs}}$ when A is clear from the context and denote by $\|A\|$ the maximum among $\mathsf{n}_{\not\approx}$, $\mathsf{h}_{\not\approx}$, $|\mathsf{Pos}_{\mathrm{lhs}}|$, and $\mathsf{h}_{\mathrm{lhs}}$.

The implicit equality constraints of a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ can be assumed to ask for equality not only of subterms but also of subruns. Runs holding this property are called uniform.

DEFINITION 3.2. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. A run r of A is called* uniform *if, for each positions p, p_1, p_2 such that $r(p)$ is defined as a rule $l \xrightarrow{c} q$ satisfying $l(p_1) = l(p_2) \in Q$, $r|_{p.p_1} = r|_{p.p_2}$ holds.*

LEMMA 3.3. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Then, any run of A can be transformed into a uniform run on the same term and reaching the same state.*

With the previous lemma it is clear that the classes of languages recognizable by $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ with runs and with uniform runs coincide. We straightforwardly extend the notion of uniform runs to the setting of weak runs and call them uniform weak runs. Note that weak runs do not have to satisfy the disequality constraints or the implicit equality constraints occurring in the rules applied, whereas a uniform weak run guarantees that equality constraints are satisfied. For this reason, uniform weak runs do not allow an equivalent of Lemma 3.3, i.e., it is possible that a weak run cannot

be transformed into a uniform weak run recognizing the same term and reaching the same state.

In the remainder of this section we summarize results from [21] that are relevant to our goals. We cite them using the new formalism $\mathtt{TA}_{\mathrm{ihom},\not\approx}$, as it is equivalent to the model $\mathtt{TA}_{\mathrm{hom},\not\approx}$ used in [21]. (Such equivalence can easily be proved by means of straightforward PTIME transformations in both directions.) As a first ingredient, the following proposition establishes that the class $\mathtt{TA}_{\mathrm{ihom}}$ can be used to represent images of regular tree languages under tree homomorphisms.

PROPOSITION 3.4 (from Proposition 4.6 and section 4.1 of [21]). *There effectively exists a polynomial $\mathcal{P}$ satisfying the following condition. Let A be a $\mathtt{TA}$, and let H be a tree homomorphism. Then, a $\mathtt{TA}_{\mathrm{ihom}}$ B satisfying $\mathcal{L}(B) = H(\mathcal{L}(A))$ can be computed in time $\mathcal{P}(|A|, |H|)$.*

Now we state a technical construction that will be used to check inclusion between images of regular tree languages under tree homomorphisms.

PROPOSITION 3.5 (from Corollary 4.9, Proposition 4.13, and section 4.1 of [21]). *There effectively exist polynomials $\mathcal{P}_1, \mathcal{P}_2$ satisfying the following condition. Let A, B be $\mathtt{TA}_{\mathrm{ihom}}$ with signature Σ. Then, a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ C satisfying $\mathcal{L}(C) = \mathcal{L}(A) \cap \overline{\mathcal{L}(B)}$ can be computed in time $2^{\mathcal{P}_1(\mathsf{maxarity}(\Sigma), |\Sigma|, \|A\|, \log|A|, |B|)}$, and such that $\|C\| \leq \mathcal{P}_2(\|A\|, \|B\|)$.*

The decision of HOM presented in section 7.1 of [21] consists in reducing such a problem to a question of inclusion between two $\mathtt{TA}_{\mathrm{ihom}}$ A, A'. From the previous Proposition 3.5, it is clear that such a question can in turn be reduced to testing emptiness on a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ B recognizing $\mathcal{L}(A) \cap \overline{\mathcal{L}(A')}$. The construction of A' following the simple algorithm of Definition 7.1 in [21] takes triple exponential time, complexity that is good enough in the context of [21] as it does not increase their overall cost of testing emptiness of B. Nevertheless, such construction can be refined using the ideas presented in section 5.3 of [21], and combined with the complement construction of section 3 and Lemma 4.8 of [21], in order to reduce the cost of constructing an automaton recognizing $\overline{\mathcal{L}(A')}$ to a single exponential. Thus, we can obtain the following improved result.

PROPOSITION 3.6 (from the ideas on the constructions in sections 3, 5.3, and 7.1 and Lemma 4.8 for constructing an automaton recognizing $\overline{\mathcal{L}(A')}$, Proposition 4.13, section 4.1, and Lemma 7.3 of [21]). *There effectively exist polynomials $\mathcal{P}_1, \mathcal{P}_2$ satisfying the following condition. Let A be a $\mathtt{TA}_{\mathrm{ihom}}$ with signature Σ. Then, a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ B satisfying that $\mathcal{L}(B)$ is empty if and only if $\mathcal{L}(A)$ is regular can be computed in time $2^{\mathcal{P}_1(\mathsf{maxarity}(\Sigma), |\Sigma|, |A|)}$, and such that $\|B\| \leq \mathcal{P}_2(\|A\|)$.*

We recall one additional result from [21] that reasons about the finiteness of the language recognized by $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Intuitively, the authors reduce the question of finiteness to a question of emptiness on a new automaton.

PROPOSITION 3.7 (from the proof of Corollary 5.20 and section 5.3 of [21]). *There effectively exist polynomials $\mathcal{P}_1, \mathcal{P}_2$ satisfying the following condition. Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Then, a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ B satisfying that $\mathcal{L}(B)$ is empty if and only if $\mathcal{L}(A)$ is finite can be computed in time $2^{\mathcal{P}_1(\|A\|, \log|A|)}$, and such that $\|B\| \leq \mathcal{P}_2(\|A\|)$.*

4. Independent sets. The content presented in 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 [8], similar notions were needed to prove

EXPTIME-completeness of emptiness for tree automata with disequality constraints. We explain the differences with such notions after Definition 4.1.

Before entering into the details, we give some intuition about how these results are connected with the rest of the paper. Let P be the set of suffixes of positions occurring in disequality constraints, let r be a uniform run recognizing a term t, let P' be a set of positions of r that are forced by the automaton to have the same pending terms, and let $r_1, \ldots, r_K$ be runs recognizing terms $t_1, \ldots, t_K$ and reaching the same state as any of the identical runs $r|_{p'}$ for $p' \in P'$. We will define that the set $\{t_1, \ldots, t_K, t|_{p'}\}$ is P-independent if for any $p \in P$ either all $t_1|_p, \ldots, t_K|_p, t|_{p'}|_p$ coincide or are pairwise different. Note that P includes the positions p of $r|_{p'}$ that are involved in a disequality constraint tested just above p'. Intuitively, the fact that $\{t_1, \ldots, t_K, t|_{p'}\}$ is P-independent allows us to conclude that, for each of the positions p inside $r|_{p'}$ and involved in an atomic disequality constraint tested just above p', we can use any of the r_i's, except for at most one of them, to do a replacement at p' that still satisfies such constraint, as either all the t_i's coincide with $t|_{p'}$ at p or all of them are different at p.

In section 4.1, the notion of independence and how to detect it efficiently is presented in a more abstract setting with tuples to simplify the presentation and obtain more general results. In section 4.2, these results are adapted from tuples to terms.

4.1. Independent sets of tuples. 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 ith component e_i. We denote the set of all such possible tuples as T. For a given finite set of tuples, we are interested in finding a "large" subset which is independent according to the following definition.

DEFINITION 4.1. *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 [8], it is proved for a fixed natural number K that, given a set S with $K^n \cdot n!$ tuples, there effectively exists an independent subset $\tilde{S}$ of S with size K. This fact is used in [8] to decide emptiness of tree automata with disequality constraints in exponential time. 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 4.2. *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 $t' \in \mathsf{coincidents}(S, t, I)$, then $\mathsf{coincidents}(S, t', I) = \mathsf{coincidents}(S, t, I)$. For a natural number K, we define a counting property on sets of tuples, namely K-smallness, that will be useful to construct an independent set of size K containing a specific tuple t.

DEFINITION 4.3. *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:* $\forall t \in S : \forall I \subsetneq \{1, \ldots, n\} : |\mathsf{coincidents}(S, t, I)| < K^{n-|I|} \cdot (n - |I|)!$.

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

LEMMA 4.4. *Let K be a natural number. Let $S \subseteq T$ be a nonempty K-small set. Then, $|S| < K^n \cdot n!$.*

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

The following lemma holds by Definitions 4.2 and 4.3.

LEMMA 4.5. *Let K be a natural number. Let $S \subseteq T$ be a set of tuples. Then, checking whether S is K-small can be done with at most $|S|^2 \cdot 2^n \cdot n$ comparisons between elements occurring in the tuples of S.*

In order to show that detecting K-smallness is enough for our purposes, we prove in Lemma 4.10 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 4.6. *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 $\{(u, v) \mid u, v \in \tilde{V}\} \cap E = \emptyset$.*

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 4.7. *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 the undirected graph $G = \langle V, E \rangle$ with $V = \mathsf{coincidents}(S, t, I)$ and $E = \{(t_1, t_2) \in V^2 \mid t_1 \neq t_2 \wedge \exists i \in \{1, \ldots, n\} - I : t_1[i] = t_2[i]\}$.*

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

LEMMA 4.8. *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.*

Proof. Consider any i in $\{1, \ldots, n\}$. If $i \in I$, then for any $t' \in \mathsf{coincidents}(S, t, I)$ we have $t'[i] = t[i]$, and the notion of independence is satisfied for $\tilde{S}$ and such i. Otherwise, if $i \notin I$, then, since $\tilde{S}$ is edge-free in $\mathsf{graph}(S, t, I)$, for any $t_1, t_2 \in \tilde{S}$ we have that t_1 and t_2 are not connected in $\mathsf{graph}(S, t, I)$. Thus, by the definition of $\mathsf{graph}(S, t, I)$ and the fact that $i \notin I$, $t_1[i] \neq t_2[i]$ follows. Hence, the notion of independence is satisfied for $\tilde{S}$ and such i. ∎

In the proof of Lemma 4.10, 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. It was argued as a part of the proof of Theorem 2 in [8], and we include a proof here to keep the paper self-contained.

LEMMA 4.9. *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 $|\tilde{V}| = \lceil \frac{|V|}{\mathsf{maxdegree}(G)+1} \rceil$.*

Proof. Let $G_1 := G$, $u_1 := u$, and let G_2 be the graph obtained from G_1 by removing u_1 and all the nodes adjacent to u_1. Note that no more than $\mathsf{maxdegree}(G)+1$ nodes have been removed. Now, choose a node u_2 of G_2 and let G_3 be the graph obtained from G_2 by removing u_2 and all the nodes adjacent to u_2. Again, no more than $\mathsf{maxdegree}(G)+1$ nodes have been removed. We proceed analogously until the resulting graph is empty. In this process we have obtained $\lceil \frac{|V|}{\mathsf{maxdegree}(G)+1} \rceil$ or more nodes $u_1, u_2, \ldots$, including u, that form an edge-free set in G. ∎

LEMMA 4.10. *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. There exists an independent set $\tilde{S}$ of tuples such that $t \in \tilde{S} \subseteq S$ and $|\tilde{S}| \geq K$.*

Proof. Since S is not K-small, there exists a set $I \subsetneq \{1, \ldots, n\}$ and a tuple $t' \in S$ satisfying $|\mathsf{coincidents}(S, t', I)| \geq K^{n-|I|} \cdot (n - |I|)!$. 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, $|\mathsf{coincidents}(S - \{t\}, t', I)| \geq K^{n-|I|} \cdot (n - |I|)!$, which implies that $S - \{t\}$ is not K-small, contradicting the assumptions of the lemma. Therefore, $\mathsf{coincidents}(S, t', I) = \mathsf{coincidents}(S, t, I)$. In other words, the mentioned tuple t' can be assumed to be t.

In the case $|I| = n - 1$, we have $|\mathsf{coincidents}(S, t, I)| \geq K^{n-(n-1)} \cdot (n - (n-1))! = K$. 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 a 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:

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

Now, we analyze some properties of $G = \langle V, E \rangle = \mathsf{graph}(S, t, I)$. First, note that $|V| = |\mathsf{coincidents}(S, t, I)| \geq K^{n-|I|} \cdot (n - |I|)!$, 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:

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

Therefore, $\mathsf{maxdegree}(G) < K^{n-|I|-1} \cdot (n - |I|)! - 1$. By Lemma 4.9, there exists a subset $\tilde{S}$ of $\mathsf{coincidents}(S, t, I)$ which is edge-free in G, includes t, and satisfies

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

By Lemma 4.8, it follows that $\tilde{S}$ is an independent set. ∎

In the following corollary we restate the result from [8] as a particular consequence of Lemmas 4.4 and 4.10. This result is useful when it is not necessary to have a distinguished tuple t in the independent subset.

COROLLARY 4.11. *Let K be a natural number. Let $S \subseteq T$ be a set of tuples such that $|S| \geq K^n \cdot n!$. Then, there exists an independent subset of tuples $\tilde{S} \subseteq S$ satisfying $|\tilde{S}| \geq K$.*

4.2. Independent sets of terms. Recall from the introduction that, given a run r, a natural number K, and a set of positions P' that are forced by the automaton to have the same pending terms, we will need to find runs $r_1, \ldots, r_K$ satisfying that all the replacements $r[r_1]_{P'}, \ldots, r[r_K]_{P'}$ falsify different disequalities. The notion of independence of previous section adapted to terms helps to overcome this problem.

DEFINITION 4.12. *Let $t \in \mathcal{T}(\Sigma)$ be a term and let p be a position. We define $t{\downarrow_p}$ as $t|_p$ if $p \in \mathsf{Pos}(t)$ and as a fixed symbol $\perp \notin \Sigma$ otherwise. Let P be a set of positions. Let $p_1, \ldots, p_n$ be the positions in P, ordered lexicographically.*[1] *We define $\mathsf{Tuple}_P(t) = \langle t{\downarrow_{p_1}}, \ldots, t{\downarrow_{p_n}} \rangle$. Let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms. 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 number. We say that S is (K, P)-*small *if $\mathsf{Tuples}_P(S)$ is K-small.*

Under certain additional conditions, we will see that if the run r has $\bar{r}$ pending at any of the positions in P', and the terms recognized by $r_1, \ldots, r_K, \bar{r}$ are independent with respect to the set of positions P that can be affected by disequality constraints, then all the replacements $r[r_1]_{P'}, \ldots, r[r_K]_{P'}$ falsify different disequalities.

The results of the previous section adapted to terms show how such an independent set can be detected, and with which complexity. Nevertheless, the translation of such results from tuples to terms and positions requires the mapping Tuples_P to be injective, and this is not the case in general. In our concrete setting, this holds thanks to the fact that the set of positions P that we will consider contains λ. (In fact, P will be the set of positions that can be affected by the disequality constraints.) We will consider sets of terms $\{t_1, \ldots, t_m\} \subseteq \mathcal{T}(\Sigma)$ such that $P \cap \mathsf{Pos}(t_1) = \cdots = P \cap \mathsf{Pos}(t_m)$. Note that in this case, $\{t_1, \ldots, t_m\}$ is P-independent if and only if for each $p \in P$, either p is not in any of the sets $\mathsf{Pos}(t_1), \ldots, \mathsf{Pos}(t_m)$, or it is in all of them and either $t_1|_p = \cdots = t_m|_p$ or the subterms $t_1|_p, \ldots, t_m|_p$ are pairwise different.

The following facts are straightforwardly implied by Lemmas 4.4, 4.5, and 4.10 and Corollary 4.11, respectively.

LEMMA 4.13. *Let P be a set of positions including λ and let K be a natural number. Let $S \subseteq \mathcal{T}(\Sigma)$ be a nonempty (K, P)-small set of terms. Then, $|S| < K^{|P|} \cdot |P|!$.*

LEMMA 4.14. *Let P be a set of positions including λ and let K be a natural number. Let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms. Then, checking whether S is (K, P)-small can be done with at most $|S|^2 \cdot 2^{|P|} \cdot |P|$ comparisons between elements of $\{t|_p \mid t \in S \wedge p \in \mathsf{Pos}(t) \cap P\} \cup \{\perp\}$.*

LEMMA 4.15. *Let P be a set of positions including λ, 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. There exists a P-independent set $\tilde{S}$ of terms such that $t \in \tilde{S} \subseteq S$ and $|\tilde{S}| \geq K$.*

COROLLARY 4.16. *Let P be a set of positions including λ, let K be a natural number, and let $S \subseteq \mathcal{T}(\Sigma)$ be a set of terms such that $|S| \geq K^{|P|} \cdot |P|!$. Then, there exists a P-independent set of terms $\tilde{S} \subseteq S$ satisfying $|\tilde{S}| \geq K$.*

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

5. Constraint-satisfying replacements. In this section we study how to perform replacements on runs of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ in a way that guarantees that all constraints are satisfied, i.e., that the weak run resulting from the replacement is actually a run. We start in section 5.1 focusing on the implicit equality constraints. This is the simplest case since, in order to satisfy the equalities tested by the run, it can be proved that it suffices to perform the replacement simultaneously at several parallel positions. Moreover, these positions for the replacement can be easily defined by considering the positions of the run involved in equality tests. The remaining sections are devoted to disequality constraints. In section 5.2 we formalize a criterion to distinguish two different kinds of disequality constraints. Intuitively, this distinction depends on how "close" are the positions tested by the disequality constraint to the positions where the replacement is performed. The "closest" ones are studied in section 5.3 and the "furthest" ones in section 5.4.

Recall that the reason to study replacements on runs of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ is to be able to reason about the emptiness of the recognized language: if any "big enough" run can be reduced by means of a decreasing replacement, then emptiness can be decided by checking only "small" runs. In order to formalize the notion of such decreasing replacements, 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 a Knuth–Bendix ordering [3] with the standard term size comparison as the first component satisfies these conditions. We consider this ordering extended to runs r, r' in a way such that $r \ll r'$ if $\mathsf{term}(r) \ll \mathsf{term}(r')$.

5.1. Equality constraints. We start by defining a kind of replacement that satisfies all the implicit equality constraints occurring in the rules applied in a run. Note that, in general, a simple replacement $r[r']_p$ is not enough, since equality tests checked at positions above p may become falsified after the replacement. To satisfy these equality tests it is necessary that such a replacement is done at the same time at all the subruns involved in an equality test, i.e., a replacement needs to be performed simultaneously at multiple parallel positions. In order to simplify the definition of these positions, we reason over uniform weak runs. Recall that uniform weak runs satisfy the implicit equality constraints occurring in the rules applied, and moreover, an implicit equality constraint asks for equality not only of subterms but also of weak subruns. Using these properties of uniform weak runs and the fact that equality constraints of $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ are implicitly defined by duplication of states, we can easily define the positions for the replacement with the following notion of abstract positions. Given a uniform weak run r and a position $p \in \mathsf{Pos}(r)$, we describe the abstract position of p in r as 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 5.1. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let r be a uniform weak run of A, and let p be a position in $\mathsf{Pos}(r)$. 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 $l \xrightarrow{c} q$. 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.*

We denote abstract positions as P, with possible subscripts, and say that P is a pure *abstract position when it is of the form $q_1 \dots q_n.\lambda$, for $q_1, \dots, q_n \in Q$.*

Note that, for a uniform weak run r and a position $p \in \mathsf{Pos}(r)$, $r(p)$ is defined if and only if $\mathsf{abstract}(p)$ is a pure abstract position. Furthermore, if two positions

$p_1, p_2 \in \mathsf{Pos}(r)$ satisfy that $\mathsf{abstract}(p_1), \mathsf{abstract}(p_2)$ are pure and identical, then $r(p_1)$ and $r(p_2)$ are defined and $r|_{p_1}$ and $r|_{p_2}$ are equal. Intuitively, according to the definition of uniform weak run of a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$, for positions sharing the same sequence of states $q_1 \dots q_n$ from the root, the corresponding uniform weak subruns must coincide. For this reason, with $r|_{q_1 \dots q_n.\lambda}$ we denote such a uniform weak subrun, and with $r(q_1 \dots q_n.\lambda)$ the rule applied at the root position of $r|_{q_1 \dots q_n.\lambda}$. In addition, given a uniform weak run r' reaching the same state as $r|_{q_1 \dots q_n.\lambda}$, we denote as $r[r']_{q_1 \dots q_n.\lambda}$ the result of replacing by r' the uniform weak subrun at each position p holding $\mathsf{abstract}_r(p) = q_1 \dots q_n.\lambda$. It is straightforward that such $r[r']_{q_1 \dots q_n.\lambda}$ is also a uniform weak run. The following lemma formally states this property.

LEMMA 5.2. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let r, r' be uniform weak runs of A. Let P be a pure abstract position of r such that $r|_P$ and r' reach the same state. Then, $r[r']_P$ is a uniform weak run.*

Proof. Since $r|_P$ and r' reach the same state, $r[r']_P$ is a weak run. To prove that $r[r']_P$ is uniform, we proceed by contradiction by assuming that there exist positions p, p_1, p_2 such that $r[r']_P(p)$ is defined as a rule whose left-hand side has the same state at the positions p_1, p_2 and $r[r']_P|_{p.p_1} \neq r[r']_P|_{p.p_2}$. This is only possible if $p.p_1$ and $p.p_2$ belong to $\mathsf{abstract}_r^{-1}(P')$ for some pure abstract position P' of r. Hence, the replacement at P modifies the corresponding subruns in the same way, and thus, $r[r']_P|_{p.p_1} = r[r']_P|_{p.p_2}$, a contradiction. ∎

The previous fact is equivalent to saying that a replacement $r[r']_P$ defined by means of a pure abstract position P necessarily satisfies the equality tests. However, note that nothing is guaranteed about the disequality constraints occurring in the rules applied in $r[r']_P$, even in the case where r and r' are uniform runs. Dealing with disequality constraints requires more complex arguments, and we present them in the following sections.

Before concluding this section, we give some additional definitions on abstract positions. This formalism, besides simplifying the previous Lemma 5.2, also helps in making the remaining reasonings of our work simpler and more accessible. For this reason, we are interested in adapting some typical operations on positions to the setting of abstract positions. In particular, we need to relax the conditions on abstract positions by allowing concatenations of the form $P.p$, where P is a pure abstract position and p is a position. We also need to compare abstract positions between them by means of a prefix relation. Such a relation is more complex for abstract positions than for positions, since an abstract position implicitly represents a set of positions.

DEFINITION 5.3. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let r be a uniform weak run of A, and let $P, \bar{P}$ be abstract positions of r more explicitly written of the form $q_1 \dots q_n.p$ and $\bar{q}_1 \dots \bar{q}_m.\bar{p}$, respectively. We say that P is a* prefix *of $\bar{P}$, denoted $P \leq \bar{P}$, if $n \leq m$, $q_1 \dots q_n = \bar{q}_1 \dots \bar{q}_n$, and the following conditions hold:*

- *If $n = m$, then $p \leq \bar{p}$.*
- *If $n < m$, then the left-hand side l of the rule $r(q_1 \dots q_n.\lambda)$ has an occurrence of state $\bar{q}_{n+1}$ in the subterm $l|_p$.*

Moreover, we say that P is $m - n$ steps above *$\bar{P}$. We say that P is a* strict *prefix of $\bar{P}$, denoted $P < \bar{P}$, if $P \leq \bar{P}$ and $P \neq \bar{P}$. We say that P and $\bar{P}$ are parallel, denoted $P \parallel \bar{P}$, if neither $P \leq \bar{P}$ nor $\bar{P} \leq P$ hold.*

Let P be a pure abstract position of r more explicitly written of the form $q_1 \dots q_n.\lambda$. Let p be a position in $\mathsf{Pos}(r|_P)$. By the concatenation *$P.p$ we denote the abstract position $q_1 \dots q_{n-1}.\mathsf{abstract}_{r|_P}(p)$.*

It is important to remark that the previous definitions on abstract positions contradict a usual intuition on positions. Consider two parallel abstract positions P_1 and P_2 of a uniform weak run r and note that, even though they are parallel, it is possible that both of them are prefix of a common abstract position P of r, i.e., that $P_1, P_2 \leq P$. It is easy to see that this is only possible in the case where P_1, P_2, and P can be written of the form $q_1 \dots q_n.p_1$, $q_1 \dots q_n.p_2$ and $q_1 \dots q_n.q_{n+1} \dots q_m.p$, respectively, with p_1 and p_2 being parallel positions and $n < m$, and moreover, the left-hand side of the rule $r(q_1 \dots q_n.\lambda)$ has an occurrence of the state q_{n+1} below the positions p_1 and p_2. The following technical lemma proves that, in the particular case where one of P_1, P_2 is a pure abstract position, it is not possible that both of them are prefix of P.

LEMMA 5.4. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let r be a uniform weak run of A. Let P_1, P_2 be parallel abstract positions of r such that at least one of them is pure. Then, there is no abstract position P of r such that $P_1 \leq P$ and $P_2 \leq P$.*

Proof. Assume without loss of generality that P_1 is pure, and consider any abstract position P of r such that $P_1 \leq P$. In order to conclude, it suffices to prove $P_2 \not\leq P$. Let P_1 and P_2 be more explicitly written of the form $q_1 \dots q_n.\lambda$ and $q_1 \dots q_i.\bar{q}_{i+1} \dots \bar{q}_m.p_2$, respectively, where $q_1 \dots q_i$ is the maximal common prefix of P_1 and P_2. Note that since $P_1 \not\leq P_2$ and P_1 is pure, necessarily $i < n$. In the case where $i = m$, the left-hand side of the rule $r(q_1 \dots q_i.\lambda)$ does not have any occurrence of the state q_{i+1} below position p_2: otherwise $P_2 \leq P_1$ contradicting the assumption that P_1 and P_2 are parallel. Hence, in this case $P_2 \not\leq P$. In the case where $i < m$, we have that $q_{i+1} \neq \bar{q}_{i+1}$, and hence, $P_2 \not\leq P$ follows again. This concludes the proof. ∎

As a final remark, note that an abstract position is defined with respect to a concrete uniform weak run, which leads to some counterintuitive cases when comparing abstract positions of different uniform weak runs. For example, consider two uniform weak runs r_1 and r_2 and positions $p_1 \in \mathsf{Pos}(r_1)$ and $p_2 \in \mathsf{Pos}(r_2)$. Clearly, it is possible for p_1 and p_2 to be equal and yet $\mathsf{abstract}_{r_1}(p_1) \neq \mathsf{abstract}_{r_2}(p_2)$, and it is also possible that $P = \mathsf{abstract}_{r_1}(p_1) = \mathsf{abstract}_{r_2}(p_2)$ and yet $\mathsf{abstract}_{r_1}^{-1}(P) \neq \mathsf{abstract}_{r_2}^{-1}(P)$. For these reasons, comparing abstract positions of r_1 and r_2 can only be done when r_1 and r_2 are "similar." In our setting, we are interested in the case where r_1 and r_2 can be written of the form $r[r'_1]_P$ and $r[r'_2]_P$, respectively, for some uniform weak runs r, r'_1, r'_2 and pure abstract position P of r. Note that in such case, an abstract position P_1 of r_1 and an abstract position P_2 of r_2 can be compared if P is not a strict prefix of P_1 or P_2.

5.2. Classifying disequality constraints. We now consider the disequality constraints of the rules applied in $r[r']_P$, where r and r' are uniform runs and P is a pure abstract position of r such that $r|_P$ and r' reach the same state. Recall that $r[r']_P$ is necessarily a uniform weak run as stated in Lemma 5.2. Moreover, since r and r' are uniform runs, for $r[r']_P$ to satisfy all the constraints—and thus be a run—it only remains to prove that the disequality constraints of rules applied at prefixes of P are satisfied. That is, we have to show that $\mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_1} \neq \mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_2}$ holds for each triplet of positions $\langle \bar{p}, \bar{p}_1, \bar{p}_2 \rangle$ satisfying the following conditions: $r(\bar{p})$ is defined, the atom $\bar{p}_1 \not\approx \bar{p}_2$ occurs in the disequality constraint of the rule $r(\bar{p})$, $\mathsf{abstract}_r(\bar{p}) < P$, and $\bar{p}.\bar{p}_1, \bar{p}.\bar{p}_2 \in \mathsf{Pos}(r[r']_P)$. We can generalize this idea to abstract positions in order to simplify further reasonings. Consider any two such triplets $\langle \bar{p}, \bar{p}_1, \bar{p}_2 \rangle$ and $\langle \bar{p}', \bar{p}_1, \bar{p}_2 \rangle$ such that $\mathsf{abstract}_r(\bar{p}) = \mathsf{abstract}_r(\bar{p}')$. Note that, since r is a uniform run, it follows that $r|_{\bar{p}} = r|_{\bar{p}'}$. Therefore, a replacement at P satisfies $\mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_1} \neq$

$\mathsf{term}(r[r']_P)|_{\bar{p}.\bar{p}_2}$ if and only if it also satisfies $\mathsf{term}(r[r']_P)|_{\bar{p}'.\bar{p}_1} \neq \mathsf{term}(r[r']_P)|_{\bar{p}'.\bar{p}_2}$. In other words, different triplets with the same abstract positions are actually equivalent and we only need to reason about one of them. The following definition formalizes these triplets with abstract positions. Moreover, it also distinguishes the case where the positions are "close" to P, i.e., the test involves subterms of $\mathsf{term}(r')$, from the one where the positions are "far" from P.

DEFINITION 5.5. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let r be a uniform weak run of A, and let P be a pure abstract position of r. Let $\bar{P}$ be a pure abstract position of r such that $\bar{P} < P$, and let $\bar{p}_1, \bar{p}_2$ be positions such that the atom $\bar{p}_1 \not\approx \bar{p}_2$ occurs in the disequality constraint of the rule $r(\bar{P})$. We say that a* disequality is tested *(by r) at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. Moreover, we say that it is a* close disequality *(of r) with respect to P if $P \leq \bar{P}.\bar{p}_1$ or $P \leq \bar{P}.\bar{p}_2$, and otherwise, we say that it is a* far disequality *(of r) with respect to P. We say that it is* falsified *if $\bar{p}_1, \bar{p}_2 \in \mathsf{Pos}(r|_{\bar{P}})$ and $\mathsf{term}(r|_{\bar{P}})|_{\bar{p}_1} = \mathsf{term}(r|_{\bar{P}})|_{\bar{p}_2}$.*

When r and P are clear from the context, we just say that a disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is close/far and omit that the distinction is done with respect to P. In our setting, since we want to reason about the disequality tests falsified when performing a replacement, such an implicit P corresponds to where the replacement takes place. In some cases, when $\bar{p}_1, \bar{p}_2$ are clear from the context or not relevant, we just say that a disequality is tested at $\bar{P}$. Finally, we say that a disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is tested d steps above P if $\bar{P}$ is d steps above P. We deal with close and far disequalities separately in the following sections.

5.3. Close disequalities. We first tackle the falsified close disequalities in the replacement $r[r']_P$. Recall that, in this case, such disequalities are necessarily tested at triplets $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$, where $\bar{P} < P$ and $P \leq \bar{P}.\bar{p}_1 \vee P \leq \bar{P}.\bar{p}_2$. Note that these conditions imply that the tests involve a subterm of $\mathsf{term}(r')$ pending at some position p', where p' is suffix of $\bar{p}_1$ or $\bar{p}_2$. We define the set of such suffixes as follows.

DEFINITION 5.6. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. We define the set of positions $\mathsf{suff}_{\not\approx}(A)$ as the set of suffixes of the positions occurring in the disequality constraints of the rules of A, i.e., $\{p \mid \exists (l \xrightarrow{c} q) \in \Delta, \exists p_1, p_2 : (p_1.p \not\approx p_2) \in c\}$, where Δ is the set of rules of A. (Recall that disequality atoms are unordered pairs.) We just write $\mathsf{suff}_{\not\approx}$ when A is clear from the context.*

In order to define replacements that do not falsify any close disequality, we first introduce an equivalence relation $\sim^A$ on terms, induced by a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A. Intuitively, two terms are equivalent if they share the same set of positions among the positions in $\mathsf{suff}_{\not\approx}$, and moreover, they satisfy the same equality and disequality relations among subterms at such positions.

DEFINITION 5.7. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. We define the equivalence relation $\sim^A$ on $\mathcal{T}(\Sigma)$ as $t \sim^A t'$ if and only if*

- $\mathsf{Pos}(t) \cap \mathsf{suff}_{\not\approx} = \mathsf{Pos}(t') \cap \mathsf{suff}_{\not\approx}$ *and*
- $\forall p_1, p_2 \in \mathsf{Pos}(t) \cap \mathsf{suff}_{\not\approx} : (t|_{p_1} = t|_{p_2} \Leftrightarrow t'|_{p_1} = t'|_{p_2})$.

Note that reflexivity, symmetry, and transitivity of $\sim^A$ are straightforward.

Now, consider a specific disequality tested by the uniform run r at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$, and assume that it is a close disequality with respect to a pure abstract position P of r. Given some candidate uniform runs $r_1, \ldots, r_n$ for replacements of the form $r[r_i]_P$, we prove that at most one of those replacements can falsify the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ if $r|_P, r_1, \ldots, r_n$ recognize distinct terms that are $\sim^A$-equivalent and form a $\mathsf{suff}_{\not\approx}$-independent set.

LEMMA 5.8. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let a close disequality with respect to P be tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. Let $r_1, \ldots, r_n$ be uniform runs of A reaching the same state as $r|_P$ and such that the terms $\mathsf{term}(r|_P), \mathsf{term}(r_1), \ldots, \mathsf{term}(r_n)$ are pairwise different and $\sim^A$-equivalent and form a $\mathsf{suff}_{\not\approx}$-independent set. Then, for at most one $i \in \{1, \ldots, n\}$, the replacement $r[r_i]_P$ falsifies the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$.*

Proof. We start considering the case $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r|_{\bar{P}})$. This is straightforward, since the assumptions that $\mathsf{term}(r|_P), \mathsf{term}(r_1), \ldots, \mathsf{term}(r_n)$ are $\sim^A$-equivalent and $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is close with respect to P guarantee that $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ for each $i \in \{1, \ldots, n\}$, and thus, in all the replacements the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is trivially satisfied. Hence, from now on we assume $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r|_{\bar{P}})$ and note that $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ follows for each $i \in \{1, \ldots, n\}$.

We reason on the underlying terms. Let $t = \mathsf{term}(r|_{\bar{P}})$, and let $s_i = \mathsf{term}(r_i)$ for each $i \in \{1, \ldots, n\}$. Note that $t|_{\bar{p}_1} \neq t|_{\bar{p}_2}$ holds. Let S be the set of positions of t where the replacements take place, i.e., $S = \{p \in \mathsf{Pos}(t) \mid \exists \bar{p} \in \mathsf{abstract}_r^{-1}(\bar{P}) : \mathsf{abstract}_r(\bar{p}.p) = P\}$. (Note that the set could be equivalently defined changing the existential quantifier in the condition by a universal quantifier.) By definition, S is a nonempty set of parallel positions, and moreover, the subterms of t pending at the positions in S are all identical. To ease the presentation, we denote by $t|_S$ the subterm of t pending at any of the positions in S, and by $t[s_i]_S$ the simultaneous replacement in t of all the subterms pending at positions in S by s_i. Note that, by the assumptions of the lemma, $t|_S, s_1, \ldots, s_n$ are distinct terms, $\sim^A$-equivalent, and form a $\mathsf{suff}_{\not\approx}$-independent set. In order to conclude, it suffices to show that at most one $i \in \{1, \ldots, n\}$ satisfies $t[s_i]_S|_{\bar{p}_1} = t[s_i]_S|_{\bar{p}_2}$.

By the assumption that $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is close with respect to P, it follows that there exists $p \in S$ such that $p \leq \bar{p}_1$ or $p \leq \bar{p}_2$. Since both cases are symmetric, without loss of generality we assume $p \leq \bar{p}_1$. Let $\bar{p}'_1$ be $\bar{p}_1 - p$ and note that $\bar{p}'_1 \in \mathsf{suff}_{\not\approx}$. Now, we distinguish cases depending on $\bar{p}_2$ and the positions in S. First, assume that there exists $p' \in S$ such that $p' \leq \bar{p}_2$. Let $\bar{p}'_2$ be $\bar{p}_2 - p'$ and note that $\bar{p}'_2 \in \mathsf{suff}_{\not\approx}$. In this case, it suffices to prove that at most one $i \in \{1, \ldots, n\}$ satisfies $s_i|_{\bar{p}'_1} = s_i|_{\bar{p}'_2}$. But, since $t|_S \sim^A s_i$ and $t|_S|_{\bar{p}'_1} = t|_p|_{\bar{p}'_1} = t|_{\bar{p}_1} \neq t|_{\bar{p}_2} = t|_{p'}|_{\bar{p}'_2} = t|_S|_{\bar{p}'_2}$, it follows that $s_i|_{\bar{p}'_1} \neq s_i|_{\bar{p}'_2}$ for all $i \in \{1, \ldots, n\}$. Second, assume that $\bar{p}_2$ is parallel to all positions in S, and consider any $i, j \in \{1, \ldots, n\}$ such that $t[s_i]_S|_{\bar{p}_1} = t[s_i]_S|_{\bar{p}_2}$ and $t[s_j]_S|_{\bar{p}_1} = t[s_j]_S|_{\bar{p}_2}$. Note that, in this case, this condition is equivalent to saying $s_i|_{\bar{p}'_1} = t|_{\bar{p}_2}$ and $s_j|_{\bar{p}'_1} = t|_{\bar{p}_2}$. It suffices to note that necessarily $i = j$: otherwise, $t|_S|_{\bar{p}'_1} = s_i|_{\bar{p}'_1} = s_j|_{\bar{p}'_1}$ since $\{t|_S, s_1, \ldots, s_n\}$ is $\mathsf{suff}_{\not\approx}$-independent, and thus, $t|_{\bar{p}_1} = t|_{\bar{p}_2}$ since $t|_S|_{\bar{p}'_1} = t|_p|_{\bar{p}'_1} = t|_{\bar{p}_1}$, contradicting $t|_{\bar{p}_1} \neq t|_{\bar{p}_2}$. Third, assume that there exists $p' \in S$ satisfying $\bar{p}_2 < p'$. Then, for all $i \in \{1, \ldots, n\}$ it follows $\mathsf{height}(t[s_i]_S|_{\bar{p}_1}) < \mathsf{height}(t[s_i]_S|_{\bar{p}_2})$, and thus, $t[s_i]_S|_{\bar{p}_1} \neq t[s_i]_S|_{\bar{p}_2}$. ∎

Now we are ready to construct a replacement $r[r']_P$ that does not falsify any close disequality. From the previous result, it is clear that a single candidate r' for the replacement at P might not suffice, and instead, we require some uniform runs $r_1, \ldots, r_n$ such that n is greater than the number of different close disequalities in r. Moreover, these $r_1, \ldots, r_n$ must reach the same state as $r|_P$ and satisfy that $\mathsf{term}(r|_P), \mathsf{term}(r_1), \ldots, \mathsf{term}(r_n)$ are pairwise different and $\sim^A$-equivalent and form a $\mathsf{suff}_{\not\approx}$-independent set. These conditions are enough to guarantee that if $r[r_i]_P$ falsifies a close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$, then no other $r[r_j]_P$ with $i \neq j$ can falsify the close disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. The following lemma states the number

n of needed candidates to construct m replacements that do not falsify any close disequality.

LEMMA 5.9. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let m be a natural number, and let $n = \mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let $r_1, \ldots, r_n$ be uniform runs of A reaching the same state as $r|_P$ and such that the terms $\mathsf{term}(r|_P), \mathsf{term}(r_1), \ldots, \mathsf{term}(r_n)$ are pairwise different and $\sim^A$-equivalent and form a $\mathsf{suff}_{\not\approx}$-independent set. Then, there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, n\}$ such that the replacements $r[r_{i_1}]_P, \ldots, r[r_{i_m}]_P$ do not falsify any close disequality.*

Proof. Note that by Lemma 5.8, each close disequality can be falsified in at most one of the replacements $r[r_1]_P, \ldots, r[r_n]_P$. Also note that, since a close disequality can be tested at most $\mathsf{h}_{\not\approx}$ steps above P and there are $\mathsf{n}_{\not\approx}$ different disequality atoms in the rules of A, it follows that there are at most $\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx}$ different close disequalities that we need to consider. Therefore, $n - \mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} = m$ of the replacements $r[r_1]_P, \ldots, r[r_n]_P$, say, $r[r_{i_1}]_P, \ldots, r[r_{i_m}]_P$, do not falsify any close disequality. ∎

The previous result is not enough for our purposes, since the arguments in section 5.4 need a bound for the case where the candidate terms are not assumed to be $\sim^A$-equivalent or form a $\mathsf{suff}_{\not\approx}$-independent set. These assumptions are necessary when the replacement must be performed at a fixed pure abstract position P of the uniform run r, but not when such P can be chosen among several possibilities. Hence, consider some pure abstract positions $P_1, \ldots, P_n$ of r such that $r|_{P_1}, \ldots, r|_{P_n}$ reach the same state and recognize distinct terms. We prove that, when n is "big enough," there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, n\}$ such that the replacements $r[r|_{P_{i_1}}]_{P_{i_m}}, \ldots, r[r|_{P_{i_{m-1}}}]_{P_{i_m}}$ do not falsify any close disequality with respect to P_{i_m}. The value of such n is given by means of the function $\mathcal{B}_{\mathrm{close}}$ of Definition 5.12, which uses as intermediate result the following bound for the number of equivalence classes induced by the relation $\sim^A$.

DEFINITION 5.10. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. We define $\mathcal{B}_{\mathrm{eq}}(A)$ as $2^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|^{|\mathsf{suff}_{\not\approx}|}$.*

LEMMA 5.11. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. The number of different equivalence classes induced by $\sim^A$ is bounded by $\mathcal{B}_{\mathrm{eq}}(A)$.*

Proof. The first condition of the definition of $\sim^A$ induces as many equivalence classes as there are subsets of $\mathsf{suff}_{\not\approx}$, and this is bounded by $2^{|\mathsf{suff}_{\not\approx}|}$. The second condition of the definition of $\sim^A$ depends on which subterms pending at positions in $\mathsf{suff}_{\not\approx}$ are equal or different. This condition induces as many equivalence classes as the number of partitions of the set $\mathsf{suff}_{\not\approx}$, and this is bounded by $|\mathsf{suff}_{\not\approx}|^{|\mathsf{suff}_{\not\approx}|}$. The statement follows by combining both bounds. ∎

We now give the concrete definition of $\mathcal{B}_{\mathrm{close}}$ for the number of needed candidates.

DEFINITION 5.12. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let m be a natural number. We define $\mathcal{B}_{\mathrm{close}}(A, m)$ as $(\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m)^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|! \cdot \mathcal{B}_{\mathrm{eq}}(A)$.*

LEMMA 5.13. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let m be a natural number, and let $n = \mathcal{B}_{\mathrm{close}}(A, m)$. Let r be a uniform run of A. Let $P_1, \ldots, P_n$ be pure abstract positions of r such that the subruns $r|_{P_1}, \ldots, r|_{P_n}$ reach the same state and recognize pairwise different terms. Then, there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, n\}$ such that $r|_{P_{i_1}} \ll \cdots \ll r|_{P_{i_m}}$ and the replacements $r[r|_{P_{i_1}}]_{P_{i_m}}, \ldots, r[r|_{P_{i_{m-1}}}]_{P_{i_m}}$ do not falsify any close disequality.*

Proof. By Lemma 5.11, there are $n' := n/\mathcal{B}_{\mathrm{eq}}(A)$ pure abstract positions among $P_1, \ldots, P_n$ satisfying that the terms recognized by the subruns at such positions

are $\sim^A$-equivalent. Without loss of generality, we assume that these n' pure abstract positions are the first ones, i.e., that the terms $\mathsf{term}(r|_{P_1}), \ldots, \mathsf{term}(r|_{P_{n'}})$ are $\sim^A$-equivalent. Since n' is $(\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m)^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|!$, by Corollary 4.16 there exists a $\mathsf{suff}_{\not\approx}$-independent subset of $\{\mathsf{term}(r|_{P_1}), \ldots, \mathsf{term}(r|_{P_{n'}})\}$ with size $n'' := \mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + m$. Without loss of generality, we assume that this subset is formed by the terms recognized by the subruns at the n'' first pure abstract positions, i.e., that $\{\mathsf{term}(r|_{P_1}), \ldots, \mathsf{term}(r|_{P_{n''}})\}$ is $\mathsf{suff}_{\not\approx}$-independent. We also assume without loss of generality that $\mathsf{term}(r|_{P_1}) \ll \cdots \ll \mathsf{term}(r|_{P_{n''}})$. Let the i_m of the statement be defined as n''. By Lemma 5.9 applied on r, P_{i_m}, and $r|_{P_1}, \ldots, r|_{P_{n''-1}}$, it follows that there exists a subset $\{i_1, \ldots, i_{m-1}\}$ of $\{1, \ldots, n'' - 1\}$ such that the replacements $r[r|_{P_{i_1}}]_{P_{i_m}}, \ldots, r[r|_{P_{i_{m-1}}}]_{P_{i_m}}$ do not falsify any close disequality. We conclude the proof by assuming without loss of generality that $i_1 < \cdots < i_{m-1}$, and thus, $r|_{P_{i_1}} \ll \cdots \ll r|_{P_{i_{m-1}}} \ll r|_{P_{i_m}}$. ∎

5.4. Far disequalities. We start with a result on far disequalities analogous to the statement on close disequalities of Lemma 5.8. More precisely, consider a specific disequality tested by the uniform run r at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$, and assume that it is a far disequality with respect to a pure abstract position P of r. Given some candidate uniform runs $r_1, \ldots, r_n$ for replacements of the form $r[r_i]_P$, we prove that at most one of those replacements can falsify the far disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. In contrast with Lemma 5.8, in this case the only needed assumptions on $r_1, \ldots, r_n$ are that they reach the same state as $r|_P$ and recognize pairwise different terms.

LEMMA 5.14. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let a far disequality with respect to P be tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$. Let $r_1, \ldots, r_n$ be uniform runs of A on distinct terms and reaching the same state as $r|_P$. Then, for at most one $i \in \{1, \ldots, n\}$, the replacement $r[r_i]_P$ falsifies the far disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$.*

Proof. We start considering the case $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r|_{\bar{P}})$. This is straightforward, since the assumption that $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is far with respect to P guarantees that $\{\bar{p}_1, \bar{p}_2\} \not\subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ for each $i \in \{1, \ldots, n\}$, and thus, in all the replacements the far disequality tested at $\langle \bar{P}, \bar{p}_1, \bar{p}_2 \rangle$ is trivially satisfied. Hence, from now on we assume $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r|_{\bar{P}})$ and note that $\{\bar{p}_1, \bar{p}_2\} \subseteq \mathsf{Pos}(r[r_i]_P|_{\bar{P}})$ follows for each $i \in \{1, \ldots, n\}$.

We reason on the underlying terms. Let $t_i = \mathsf{term}(r|_{\bar{P}.\bar{p}_i})$ for $i \in \{1, 2\}$, and note that $t_1 \neq t_2$ holds. For $i \in \{1, 2\}$, let S_i be the sets of positions of t_i where the replacements take place, i.e., $S_i = \{p \in \mathsf{Pos}(t_i) \mid \exists \bar{p} \in \mathsf{abstract}_r^{-1}(\bar{P}) : \mathsf{abstract}_r(\bar{p}.\bar{p}_i.p) = P\}$. By definition, S_i is a (maybe empty) set of parallel positions, and moreover, the subterms of t_i pending at the positions in S_i are all identical. As in the proof of Lemma 5.8, we denote by $t_i[s]_{S_i}$ the simultaneous replacement in t_i of all the subterms pending at positions in S_i by a term s. In order to conclude, it suffices to show that at most one term s satisfies $t_1[s]_{S_1} = t_2[s]_{S_2}$.

We assume that there exists a term s satisfying $t_1[s]_{S_1} = t_2[s]_{S_2}$ and prove that it is unique. Note that, for each position $p_1 \in S_1$, there is no position $p_2 \in S_2$ such that $p_1 < p_2$ or $p_2 < p_1$: otherwise, the condition $t_1[s]_{S_1} = t_2[s]_{S_2}$ would be false for any s. Also, note that $S_1 \neq S_2$: otherwise, the condition $t_1[s]_{S_1} = t_2[s]_{S_2}$ would imply $t_1 = t_2$ since all the replaced subterms of t_1 and t_2 are identical by definition. Hence, there exists a position $p \in (S_1 - S_2) \cup (S_2 - S_1)$. Without loss of generality, assume that such a p is in $S_1 - S_2$. The condition $t_1[s]_{S_1} = t_2[s]_{S_2}$ and the fact that any position in S_2 is parallel with p implies $s = t_2|_p$, and we are done. ∎

It is clear from the previous result that a single candidate r' for the replacement $r[r']_P$ is not enough to guarantee that no far disequality is falsified. In particular, given some candidates $r_1, \ldots, r_n$ recognizing distinct terms, each specific far disequality can only be falsified by one of the r_i's when performing the replacement at P. In contrast to our arguments in section 5.3 dealing with close disequalities, note that the number of far disequalities is not bounded. For this reason, the definition of the number n of needed candidates is more complex. We start with the following intermediate lemma where we only consider far disequalities that are "near" P, i.e., far disequalities tested at a bounded distance from the pure abstract position where the replacement is performed.

LEMMA 5.15. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let m, k be natural numbers, and let $n = k \cdot \mathsf{n}_{\not\approx} + m$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let $r_1, \ldots, r_n$ be uniform runs of A on distinct terms reaching the same state as $r|_P$ and such that, for a given natural number d, the replacements $r[r_1]_P, \ldots, r[r_n]_P$ do not falsify any far disequality tested at most d steps above P. Then, there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, n\}$ such that the replacements $r[r_{i_1}]_P, \ldots, r[r_{i_m}]_P$ do not falsify any far disequality tested at most $d + k$ steps above P.*

Proof. Note that by Lemma 5.14, each far disequality can be falsified in at most one of the replacements $r[r_1]_P, \ldots, r[r_n]_P$. Also note that, among the far disequalities that are tested at most $d+k$ steps above P, we do not need to consider the ones that are tested at most d steps above P since they are already satisfied by assumption. Hence, since there are $\mathsf{n}_{\not\approx}$ different disequality atoms in the rules of A, it follows that there are at most $k \cdot \mathsf{n}_{\not\approx}$ different far disequalities that we need to consider. Therefore, $n - k \cdot \mathsf{n}_{\not\approx} = m$ of the replacements $r[r_1]_P, \ldots, r[r_n]_P$, say, $r[r_{i_1}]_P, \ldots, r[r_{i_m}]_P$, do not falsify any far disequality tested at most $d + k$ steps above P. ∎

Now we are ready to tackle the far disequalities that are not "near" P. Consider that we have candidates $r_1, \ldots, r_n$ such that the replacements $r[r_1]_P, \ldots, r[r_n]_P$ do not falsify any close disequality. We assume that all of them falsify some far disequality, since otherwise, no further arguments would be needed. We define an n "big enough" to guarantee that we are able to construct from subruns of r and from $r_1, \ldots, r_n$ new candidates $r'_1, \ldots, r'_n$ for replacements at a pure abstract position $P' < P$ such that, again, $r[r'_1]_{P'}, \ldots, r[r'_n]_{P'}$ do not falsify any close disequality. Note that in the case where all of them falsify some far disequality, this argument can be iterated to obtain new candidates $r''_1, \ldots, r''_n$ to perform replacements at a $P'' < P' < P$, i.e., at a pure abstract position closer to the root. Hence, we are guaranteed to eventually find a replacement that does not falsify any far disequality. The number n of needed candidates is given by means of the function $\mathcal{B}$ of Definition 5.16 and the proof of this fact is given in Lemma 5.17. The function $\mathcal{B}$ takes two natural numbers M and N for which we do not give a concrete definition until Lemma 5.19. At this point it suffices to assume that they satisfy $M \cdot N \geq \mathcal{B}(A, M, N) = n$. In order to illustrate the definition of $\mathcal{B}$, we sketch the steps that we perform in the proof of Lemma 5.17 to construct the new candidates $r'_1, \ldots, r'_n$ and to find the new pure abstract position P' for the replacement. We start by noting that, since all the replacements $r[r_1]_P, \ldots, r[r_n]_P$ falsify far disequalities, we can consider the maximal pure abstract positions $\bar{P}_1, \ldots, \bar{P}_n$ such that, for $i \in \{1, \ldots, n\}$, the replacement $r[r_i]_P$ falsifies a far disequality tested at $\bar{P}_i$. We also assume without loss of generality that $\bar{P}_1 \geq \bar{P}_2 \geq \cdots \geq \bar{P}_n$ by reordering the runs r_i if necessary (see Figure 2). Now, the proof proceeds as follows:

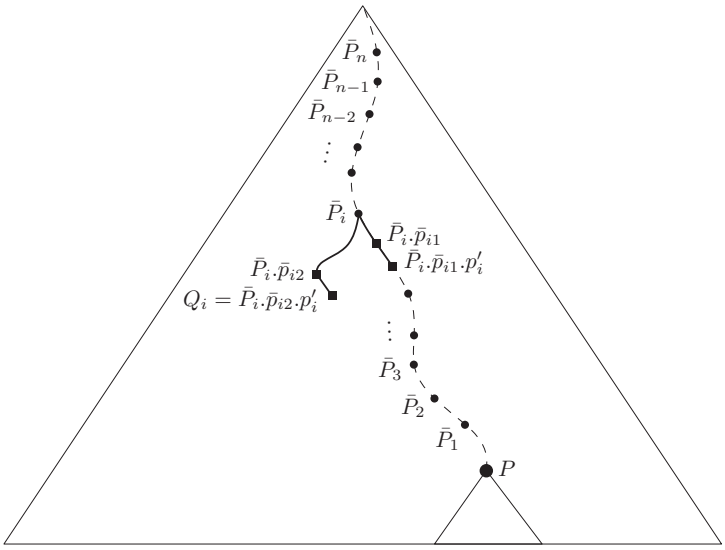

FIG. 2. *Abstract positions of the uniform run r that are considered in Lemma* 5.17. *(The single path depicted should be understood as all paths in r with abstract position P.)*

- As an initial step, we need the pure abstract positions $\bar{P}_1, \ldots, \bar{P}_n$ to be spaced between them. In particular, we want any two $\bar{P}_i, \bar{P}_j$ to be more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps away from each other. To this end, we first remove from $\bar{P}_1, \ldots, \bar{P}_n$ any repetition of pure abstract positions. Recall that, by Lemma 5.14, the far disequalities falsified by a candidate r_i are necessarily different from the far disequalities falsified by any other candidate r_j. However, since a rule may have several different disequality atoms, it follows that we can have several occurrences of identical $\bar{P}_i$'s. But there are at most $\mathsf{n}_{\not\approx}$ occurrences of the same element, and hence, it is possible to take from $\bar{P}_1, \ldots, \bar{P}_n$ a selection with $n_1 := n/\mathsf{n}_{\not\approx}$ distinct pure abstract positions, say, $\bar{P}_1, \ldots, \bar{P}_{n_1}$. Finally, we can take a new selection with $n_2 := n_1/(1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}})$ pure abstract positions from $\bar{P}_1, \ldots, \bar{P}_{n_1}$, say, $\bar{P}_1, \ldots, \bar{P}_{n_2}$, that are more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps away from each other.
- We now consider each selected $\bar{P}_i$ and their corresponding positions $\bar{p}_{i1}, \bar{p}_{i2}$ of the far disequality tested at $\bar{P}_i$ and falsified by $r[r_i]_P$, and consider common extensions of $\bar{P}_i.\bar{p}_{i1}, \bar{P}_i.\bar{p}_{i2}$ defined by positions p'_i satisfying that either $\bar{P}_i.\bar{p}_{i1}.p'_i$ or $\bar{P}_i.\bar{p}_{i2}.p'_i$ is pure in r and the terms pending at such abstract positions in r are still different (and note that the terms pending at such abstract positions in $r[r_i]_P$ must coincide). We prove that for each of such p'_i, one of the extensions, say, $\bar{P}_i.\bar{p}_{i1}.p'_i$, is a prefix of P, and that the other one, i.e., $\bar{P}_i.\bar{p}_{i2}.p'_i$, is parallel to P. Among all the possible p'_i, we choose a minimal one in size such that the extension $\bar{P}_i.\bar{p}_{i2}.p'_i$ parallel to P is pure. Thanks to the fact that the selected $\bar{P}_1, \ldots, \bar{P}_{n_2}$ are spaced between them by more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps, this extension can be done for each $\bar{P}_i$ without reaching any larger $\bar{P}_j$. Let Q_i be the extension $\bar{P}_i.\bar{p}_{i2}.p'_i$. We prove that the subruns $r|_{Q_1}, \ldots, r|_{Q_{n_2}}$ recognize distinct terms.
- At this point, we split $\{1, \ldots, n_2\}$ into two subsets depending on how close the corresponding $\bar{P}_i$'s are to P: one subset with the n_3 closest ones and the other with the $n_4 := n_2 - n_3$ remaining ones. Say they are $\{1, \ldots, n_3\}$

and $\{n_3+1,\dots,n_2\}$, respectively. Recall that each $\bar{P}_i$ is the maximal pure abstract position where a far disequality is falsified by the replacement $r[r_i]_P$. This means that the $\bar{P}_i$'s that are furthest from P necessarily correspond to the replacements where all the falsified far disequalities are "very far" from P. We now extract $M+1$ indexes from $\{1,\dots,n_3\}$, say, $\{1,\dots,M+1\}$, such that, for each $i \in \{1,\dots,M\}$, the replacement $r[r|_{Q_i}]_{Q_{M+1}}$ does not falsify any close disequality with respect to $\bar{P}_{M+1}$. Moreover, for each of such i, we extract a set of N indexes from $\{n_3+1,\dots,n_2\}$ such that, for each of such indexes j, the simultaneous replacement $r[r|_{Q_i}]_{Q_{M+1}}[r_j]_P$ does not falsify any close disequality with respect to $\bar{P}_{M+1}$. We prove that $n_3 := |Q|\cdot\mathcal{B}_{\text{close}}(A, M+1+(2\cdot \mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})\cdot \mathsf{n}_{\not\approx})$ suffices to guarantee the existence of such $M+1$ indexes, where the factor $|Q|$ is required in order to guarantee that $r|_{Q_1},\dots,r|_{Q_M}$ reach the same state as $r|_{Q_{M+1}}$. For each of such $i \in \{1,\dots,M\}$ the generation of the subset of size N from $\{n_3+1,\dots,n_2\}$ must be done depending on i since, even though for all $j \in \{n_3+1,\dots,n_2\}$ the replacement $r[r_j]_P$ does not falsify any disequality below $\bar{P}_{M+1}$, it might be the case that it falsifies some such disequality when combined with the replacement at Q_{M+1}. We prove that $n_4 := N + (2\cdot \mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})\cdot \mathsf{n}_{\not\approx}$ suffices to guarantee the existence of such N indexes.

To summarize, it is possible to combine each of the M replacements at Q_{M+1} with the N corresponding replacements at P, and thus we can define the $M\cdot N \geq \mathcal{B}(A,M,N) = n$ needed candidates as runs of the form $r[r|_{Q_i}]_{Q_{M+1}}[r_j]_P|_{\bar{P}_{M+1}}$ and the abstract position P' as $\bar{P}_{M+1}$.

By considering the values given to n_1, n_2, n_3, and n_4 in the previous explanation, we can finally define the global bound $\mathcal{B}(A,M,N)$ and prove the main result of this section.

DEFINITION 5.16. *Let A be a $\mathtt{TA}_{\text{ihom},\not\approx}$. Let M,N be natural numbers. We define*

$$\mathcal{B}(A,M,N) = \mathsf{n}_{\not\approx}\cdot(1+\mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})\cdot\big(|Q|\cdot\mathcal{B}_{\text{close}}(A,M+1+(2\cdot\mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})\cdot\mathsf{n}_{\not\approx}) + N + (2\cdot\mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})\cdot\mathsf{n}_{\not\approx}\big).$$

LEMMA 5.17. *Let $A=\langle Q,\Sigma,F,\Delta\rangle$ be a $\mathtt{TA}_{\text{ihom},\not\approx}$. Let M,N be natural numbers satisfying $M\cdot N \geq \mathcal{B}(A,M,N)$. Let $n=\mathcal{B}(A,M,N)$. Let r be a uniform run of A, and let P be a pure abstract position of r. Let $r_1,\dots,r_n$ be uniform runs of A on distinct terms reaching the same state as $r|_P$ and such that $r_1,\dots,r_n \ll r|_P$ and each one of the replacements $r[r_1]_P,\dots,r[r_n]_P$ falsifies at least one far disequality but does not falsify any close disequality. Then, there exists a pure abstract position $P'<P$ of r and uniform runs $r'_1,\dots,r'_n$ of A on distinct terms reaching the same state as $r|_{P'}$, and such that $r'_1,\dots,r'_n \ll r|_{P'}$ and each one of the replacements $r[r'_1]_{P'},\dots,r[r'_n]_{P'}$ does not falsify any close disequality.*

Proof. For each $i\in\{1,\dots,n\}$, let $\bar{P}_i$ be the maximal pure abstract position such that the replacement $r[r_i]_P$ falsifies a far disequality tested at $\bar{P}_i$. Without loss of generality, we assume that $\bar{P}_1 \geq \bar{P}_2 \geq \cdots \geq \bar{P}_n$ by reordering the runs r_i if necessary. Note that $|\mathsf{term}(r|_{\bar{P}_1})| \leq |\mathsf{term}(r|_{\bar{P}_2})| \leq \cdots \leq |\mathsf{term}(r|_{\bar{P}_n})|$. By Lemma 5.14 and the fact that there are $\mathsf{n}_{\not\approx}$ different disequality atoms in the rules of A, it follows that for each pure abstract position $P'<P$ at most $\mathsf{n}_{\not\approx}$ of the pure abstract positions $\bar{P}_i$ coincide with P'. Thus, we can choose a subset S of $\{1,\dots,n\}$ with size

$$n' := n/(\mathsf{n}_{\not\approx}\cdot(1+\mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})) = |Q|\cdot\mathcal{B}_{\text{close}}(A,M+1+(2\cdot\mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})\cdot\mathsf{n}_{\not\approx}) + N + (2\cdot\mathsf{h}_{\not\approx}+\mathsf{h}_{\text{lhs}})\cdot\mathsf{n}_{\not\approx}$$

satisfying that $\bar{P}_j$ is more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps above $\bar{P}_i$ for each $i, j \in S$ with $i < j$ and that $\bar{P}_i$ is more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps above P for each $i \in S$. Without loss of generality, we assume that S is $\{1, \ldots, n'\}$. Note that $\bar{P}_1 > \bar{P}_2 > \cdots > \bar{P}_{n'}$ and $|\mathsf{term}(r|_{\bar{P}_1})| < |\mathsf{term}(r|_{\bar{P}_2})| < \cdots < |\mathsf{term}(r|_{\bar{P}_{n'}})|$.

Consider any k in S. Since the replacement $r[r_k]_P$ falsifies a disequality tested at $\bar{P}_k$, it follows that there exist positions $\bar{p}_{k1}, \bar{p}_{k2}$ such that the atom $\bar{p}_{k1} \not\approx \bar{p}_{k2}$ occurs in the disequality constraint of the rule $r(\bar{P}_k)$, and moreover, $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}}$ and $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}}$. Let p_k be the shortest position such that at least one of $\bar{P}_k.\bar{p}_{k1}.p_k, \bar{P}_k.\bar{p}_{k2}.p_k$ is defined in r (i.e., corresponds to a pure abstract position of r), and moreover, $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$ and $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$. Note that $|\bar{p}_{k1}|, |\bar{p}_{k2}| \leq \mathsf{h}_{\not\approx}$ and $|p_k| \leq \mathsf{h}_{\mathrm{lhs}}$, which implies that $\bar{P}_i$ is not a prefix of $\bar{P}_k.\bar{p}_{k1}.p_k$ or $\bar{P}_k.\bar{p}_{k2}.p_k$ for each $i \in S$ with $i < k$, and neither P is prefix of $\bar{P}_k.\bar{p}_{k1}.p_k$ or $\bar{P}_k.\bar{p}_{k2}.p_k$. Also note that $\bar{P}_k.\bar{p}_{k1}.p_k$ and $\bar{P}_k.\bar{p}_{k2}.p_k$ are necessarily parallel and at least one of them is a prefix of P, since otherwise, either $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$ or $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}.p_k} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}.p_k}$ would be impossible. By Lemma 5.4, without loss of generality, we can assume that $\bar{P}_k.\bar{p}_{k1}.p_k$ is a prefix of P and that $\bar{P}_k.\bar{p}_{k2}.p_k$ is not a prefix of P. Let p'_k be the shortest extension of p_k such that $\bar{P}_k.\bar{p}_{k2}.p'_k$ is defined in r (i.e., corresponds to a pure abstract position of r), and moreover, $\mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k1}.p'_k} \neq \mathsf{term}(r|_{\bar{P}_k})|_{\bar{p}_{k2}.p'_k}$ and $\mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k1}.p'_k} = \mathsf{term}(r[r_k]_P|_{\bar{P}_k})|_{\bar{p}_{k2}.p'_k}$. Note that in the case where $\bar{P}_k.\bar{p}_{k2}.p_k$ already corresponds to a pure abstract position of r, p'_k is just p_k. In any case, we have $|p'_k| \leq \mathsf{h}_{\mathrm{lhs}}$. Let Q_k be the pure abstract position $\bar{P}_k.\bar{p}_{k2}.p'_k$. Observe that all of such Q_k are parallel with P.

We prove that the terms $\mathsf{term}(r|_{Q_1}), \ldots, \mathsf{term}(r|_{Q_{n'}})$ are distinct by showing that $\mathsf{term}(r|_{Q_i})$ is a strict subterm of $\mathsf{term}(r|_{Q_j})$ for each $1 \leq i < j \leq n'$. From the fact that $\bar{P}_j$ is more than $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps above $\bar{P}_i$, it follows that $\bar{P}_j.\bar{p}_{j1}.p'_j$ is a strict prefix of $\bar{P}_i$, and thus also of Q_i. Hence, $\mathsf{term}(r|_{Q_i})$ is a strict subterm of $\mathsf{term}(r|_{\bar{P}_j})|_{\bar{p}_{j1}.p'_j}$. Moreover, since Q_i is parallel with P, $\mathsf{term}(r|_{Q_i})$ is also a strict subterm of $\mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j1}.p'_j}$. Since $\mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j1}.p'_j} = \mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j2}.p'_j}$, it follows that $\mathsf{term}(r|_{Q_i})$ is also a strict subterm of $\mathsf{term}(r[r_j]_P|_{\bar{P}_j})|_{\bar{p}_{j2}.p'_j}$, i.e., of $\mathsf{term}(r[r_j]_P|_{Q_j})$. Finally, since Q_j is parallel with P, $\mathsf{term}(r|_{Q_i})$ is also a strict subterm of $\mathsf{term}(r|_{Q_j})$.

Let $m = M + 1 + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) \cdot \mathsf{n}_{\not\approx}$, and consider the first $|Q| \cdot \mathcal{B}_{\mathrm{close}}(A, m)$ elements of S, i.e., $\{1, \ldots, |Q| \cdot \mathcal{B}_{\mathrm{close}}(A, m)\}$. Necessarily, there exists $\mathcal{B}_{\mathrm{close}}(A, m)$ elements among them, say, $\{1, \ldots, \mathcal{B}_{\mathrm{close}}(A, m)\}$ without loss of generality, such that the subruns of r at the pure abstract positions $Q_1, \ldots, Q_{\mathcal{B}_{\mathrm{close}}(A,m)}$ reach the same state. By Lemma 5.13, there exists a subset $\{i_1, \ldots, i_m\}$ of $\{1, \ldots, \mathcal{B}_{\mathrm{close}}(A, m)\}$ such that $r|_{Q_{i_1}} \ll \cdots \ll r|_{Q_{i_m}}$ and the replacements $r[r|_{Q_{i_1}}]_{Q_{i_m}}, \ldots, r[r|_{Q_{i_{m-1}}}]_{Q_{i_m}}$ do not falsify any close disequality. Moreover, by Lemma 5.15, there exists a subset $\{j_1, \ldots, j_M\}$ of $\{i_1, \ldots, i_{m-1}\}$ such that the replacements $r[r|_{Q_{j_1}}]_{Q_{i_m}}, \ldots, r[r|_{Q_{j_M}}]_{Q_{i_m}}$ do not falsify any far disequality tested at most $2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps above Q_{i_m}. Note that, since $\bar{P}_{i_m}$ is at most $\mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}$ steps above Q_{i_m}, the replacements do not falsify any disequality tested at most $\mathsf{h}_{\not\approx}$ steps above $\bar{P}_{i_m}$, and hence, they do not falsify any close disequality with respect to $\bar{P}_{i_m}$.

Now, consider the last $|S| - |Q| \cdot \mathcal{B}_{\mathrm{close}}(A, m)$ remaining elements of S. Observe that $S' := S - \{1, \ldots, |Q| \cdot \mathcal{B}_{\mathrm{close}}(A, m)\} = \{|Q| \cdot \mathcal{B}_{\mathrm{close}}(A, m) + 1, \ldots, n'\} = \{n' - (N + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) \cdot \mathsf{n}_{\not\approx}) + 1, \ldots, n'\}$. Also, note that, for each $i \in S'$, the replacement $r[r_i]_P$ does not falsify any disequality tested below or at $\bar{P}_{i_m}$. Thus, for each $i \in S'$ and each $k \in \{1, \ldots, M\}$, since $\bar{P}_{i_m}.\bar{p}_{i_m 1}.p'_{i_m}$ and $\bar{P}_{i_m}.\bar{p}_{i_m 2}.p'_{i_m}$ are parallel, it follows

that the replacement $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P$ does not falsify any disequality tested below or at $\bar{P}_{i_m}.\bar{p}_{i_m 1}.p'_{i_m}$. Recall that $|\bar{p}_{i_m 1}| \leq \mathsf{h}_{\not\approx}$ and $|p'_{i_m}| \leq \mathsf{h}_{\mathrm{lhs}}$. By Lemma 5.15, for each fixed $k \in \{1, \ldots, M\}$, we can choose a subset S_k of S' with size N such that, for each $i \in S_k$, the replacement $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P$ does not falsify any disequality tested below or at $\bar{P}_{i_m}$, and moreover, it does not falsify any disequality tested at most $\mathsf{h}_{\not\approx}$ steps above $\bar{P}_{i_m}$, i.e., it does not falsify any close disequality with respect to $\bar{P}_{i_m}$. Let r_{ki} be $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P|_{\bar{P}_{i_m}}$ for each $k \in \{1, \ldots, M\}$ and each $i \in S_k$. Note that all such r_{ki}'s are uniform runs of A on distinct terms reaching the same state as $r|_{\bar{P}_{i_m}}$. Moreover, each of such r_{ki} satisfies $r_{ki} \ll r|_{\bar{P}_{i_m}}$, and the replacement $r[r_{ki}]_{\bar{P}_{i_m}}$, which in fact produces $r[r|_{Q_{j_k}}]_{Q_{i_m}}[r_i]_P$, does not falsify any close disequality with respect to $\bar{P}_{i_m}$. Observe that there are $M \cdot N \geq n$ of such r_{ki}'s. Thus, by defining P' as $\bar{P}_{i_m}$ and $r'_1, \ldots, r'_n$ as n of such r_{ki}'s, the lemma follows. ∎

At this point it is clear that, by iterative applications of Lemma 5.17, we can construct a replacement that does not falsify any disequality or implicit equality constraint—i.e., a replacement that produces a run—whenever we have $\mathcal{B}(A, M, N)$ candidates for the replacement that do not falsify any close disequality. Moreover, note that since the candidates considered are smaller than the subrun being replaced, such replacement necessarily decreases the size of the starting run. The following corollary is an immediate consequence of this fact stating that, when the starting run is accepting, then it is not a minimum accepting run since we can decrease its size by performing such a replacement.

COROLLARY 5.18. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let M, N be natural numbers satisfying $M \cdot N \geq \mathcal{B}(A, M, N)$. Let $n = \mathcal{B}(A, M, N)$. Let r be an accepting uniform run of A, and let P be a pure abstract position of r. Let $r_1, \ldots, r_n$ be uniform runs of A on distinct terms reaching the same state as $r|_P$ and such that $r_1, \ldots, r_n \ll r|_P$ and each one of the replacements $r[r_1]_P, \ldots, r[r_n]_P$ does not falsify any close disequality. Then, r is not a minimum accepting run.*

In order to conclude, it only remains to prove that there exist M and N satisfying $M \cdot N \geq \mathcal{B}(A, M, N)$. In the following lemma we give concrete values for M and N that satisfy that condition. Note that there exist alternative definitions, but the ones we use are rather straightforward and lead to a simple proof.

LEMMA 5.19. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let $M = \mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) + 1$ and $N = \mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) \cdot (|Q| \cdot \mathcal{B}_{\mathrm{close}}(A, M + 1 + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) \cdot \mathsf{n}_{\not\approx}) + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) \cdot \mathsf{n}_{\not\approx})$. Then, $M \cdot N = \mathcal{B}(A, M, N)$.*

Proof. It follows by replacing the N in the definition of $\mathcal{B}$ by the definition of N in the statement and factoring the result. More precisely, let $X = \mathsf{n}_{\not\approx} \cdot (1 + \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}})$ and $Y = (|Q| \cdot \mathcal{B}_{\mathrm{close}}(A, M + 1 + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) \cdot \mathsf{n}_{\not\approx}) + (2 \cdot \mathsf{h}_{\not\approx} + \mathsf{h}_{\mathrm{lhs}}) \cdot \mathsf{n}_{\not\approx})$ and note that $M = X + 1$ and $N = X \cdot Y$. Then, $\mathcal{B}(A, M, N) = X \cdot (Y + N) = X \cdot (Y + X \cdot Y) = X \cdot ((1 + X) \cdot Y) = (1 + X) \cdot X \cdot Y = M \cdot N$. ∎

6. Emptiness decision algorithm. In this section we introduce an algorithm that decides the emptiness of the language recognized by $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ in exponential time. In contrast to the previous section, we can now refrain from reasoning on runs since most of the information they provide is superfluous in our current setting. In particular, the only relevant data that the algorithm needs from a run is the term it recognizes and the state it reaches. For this reason, we focus on a formalism simpler than runs, namely, (term, state)-pairs defined as follows.

DEFINITION 6.1. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. A (term, state)-pair $\langle t, q\rangle \in \mathcal{T}(\Sigma) \times Q$ is called a* feasible pair *of A if there exists a run of A on t reaching q, and moreover, it is called* accepting *if such run is accepting, i.e., if $q \in F$.*

We compare (term, state)-pairs by the lexicographic extension of $\ll$ and an arbitrary total ordering on states. We also use $\ll$ to denote the ordering on (term, state)-pairs.

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 there exists a run with only one applied rule on t and reaching 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 cannot be used to construct the minimum term with respect to $\ll$ of $\mathcal{L}(A)$. This fact can be detected using the results of the previous section. Corollary 6.4 is a consequence of Corollary 5.18 and Lemma 5.9.

DEFINITION 6.2. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let $\langle t, q\rangle, \langle t', q'\rangle \in \mathcal{T}(\Sigma)\times Q$ be feasible (term, state)-pairs. 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 $t|_p = t'$.*

DEFINITION 6.3. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let M, N be natural numbers defined as in Lemma* 5.19. *We define $K(A)$ as $\mathsf{h}_{\not\approx} \cdot \mathsf{n}_{\not\approx} + \mathcal{B}(A, M, N)$.*

COROLLARY 6.4. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let $t, s_1, \ldots, s_{K(A)}$ be distinct terms in $\mathcal{T}(\Sigma)$ with runs of A on them reaching a state $q \in Q$ and such that $\{t, s_1, \ldots, s_{K(A)}\}$ is $\mathsf{suff}_{\not\approx}$-independent, $t \sim^A s_1, \ldots, s_{K(A)}$, and $s_1, \ldots, s_{K(A)} \ll t$. Then, $\langle t, q\rangle$ is not a piece of the minimum accepting pair of A.*

According to the previous corollary, 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 check whether it has a subset $\{s_1, \ldots, s_{K(A)}\}$ such that $\{t, s_1, \ldots, s_{K(A)}\}$ is $\mathsf{suff}_{\not\approx}$-independent. The time complexity of searching for such subset is too high for our goals. Fortunately, section 4.2 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 $(K(A)+1, \mathsf{suff}_{\not\approx})$-small set of terms. If the addition of t to this set makes it $\mathrm{non}(K(A)+1, \mathsf{suff}_{\not\approx})$-small, then, by Lemma 4.15, it follows the existence of the subset $\{s_1, \ldots, s_{K(A)}\}$ mentioned above. Thus, in this case we must discard the pair $\langle t, q\rangle$, since it is not a piece of the minimum accepting pair of A.

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 (term, state)-pairs, which are added to Candidates. This generation is performed (i) using the left-hand sides of rules in Δ to determine the symbols in the top-most positions of the new terms, (ii) using the feasible (term, state)-pairs 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 6.5. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let $S \subseteq \mathcal{T}(\Sigma) \times Q$ be a set of feasible (term, state)-pairs. Let $\langle t, q\rangle \in S$ be a feasible (term, state)-pair. We*

define the set of instantiations *of* Δ *with* S *and* $\langle t, q\rangle$ *as the set of feasible pairs:*

$$\big\{\langle t' = l[t_1]_{p_1}\dots[t_n]_{p_n}, q'\rangle \mid \exists (l \xrightarrow{c} q') \in \Delta,\ \{p_1,\dots,p_n\} = \mathsf{Pos}_Q(l),\ \forall i \in \{1,\dots,n\} : \langle t_i, l(p_i)\rangle \in S,\ \exists i \in \{1,\dots,n\} : \langle t_i, l(p_i)\rangle = \langle t, q\rangle,\ \forall i,j \in \{1,\dots,n\} : (l(p_i) = l(p_j) \Rightarrow t_i = t_j),\ \forall (\bar{p}_1 \not\approx \bar{p}_2) \in c : (\bar{p}_1, \bar{p}_2 \in \mathsf{Pos}(t') \Rightarrow t'|_{\bar{p}_1} \neq t'|_{\bar{p}_2})\big\}.$$

When there are no more pairs in Candidates to be considered, the algorithm stops and states nonemptiness if there is a (term, state)-pair in Definitive where the state is final. We present in Algorithm 6.1 a formalization of the previous explanations.

Algorithm 6.1 Emptiness decision for the language recognized by a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A.

Input: a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ $A = \langle Q, \Sigma, F, \Delta\rangle$.
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 there exists a rule $(l \xrightarrow{c} q) \in \Delta$ with $l \in \mathcal{T}(\Sigma)$ and satisfying $\forall (\bar{p}_1 \not\approx \bar{p}_2) \in c : (\bar{p}_1, \bar{p}_2 \in \mathsf{Pos}(l) \Rightarrow l|_{\bar{p}_1} \neq l|_{\bar{p}_2})$.
(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 $\{s \mid \langle s, q\rangle \in \mathsf{Definitive} \wedge s \sim^A t\} \cup \{t\}$ is $(K(A)+1, \mathsf{suff}_{\not\approx})$-small:
    (i) Insert $\langle t, q\rangle$ in Definitive.
    (ii) Insert in Candidates all the elements in the set of instantiations of Δ with Definitive and $\langle t, q\rangle$.
(3) If there is a pair $\langle t, q\rangle \in$ Definitive with $q \in F$, then output "NON-EMPTY," else output "EMPTY."

Example 6.6. Consider the signature Σ with binary symbols h, g and a nullary symbol a, and the language of terms over Σ of the form $h(t_1, t_2)$ satisfying that t_1, t_2 are different complete trees over g and a. Such language is recognized by the $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ $A = \langle \{q, q', q_{\mathtt{accept}}\}, \Sigma, \{q_{\mathtt{accept}}\}, \{a \to q|q',\ g(q,q) \to q|q',\ h(q,q') \xrightarrow{1\not\approx 2} q_{\mathtt{accept}}\}\rangle$, where we use $l \to q|q'$ to simultaneously denote the rules $l \to q$ and $l \to q'$.

In order to apply Algorithm 6.1, we first need to fix an ordering $\ll$ for terms. We choose a natural recursive definition: for distinct terms t, t', if $|t| < |t'|$, then $t \ll t'$, and in the case $|t| = |t'|$ (which implies that the sizes are at least 2 since $t \neq t'$), $t \ll t'$ if $(t(\lambda) = g \wedge t'(\lambda) = h) \vee (t(\lambda) = t'(\lambda) \wedge t|_1 \ll t'|_1) \vee (t(\lambda) = t'(\lambda) \wedge t|_1 = t'|_1 \wedge t|_2 \ll t'|_2)$. To lexicographically extend the ordering to (term, state)-pairs we simply assume $\langle t, q\rangle \ll \langle t, q'\rangle \ll \langle t, q_{\mathtt{accept}}\rangle$.

We execute Algorithm 6.1 step by step. First, Candidates $:= \{\langle a, q\rangle,\ \langle a, q'\rangle\}$ is set at step 1. Second, step 2 is executed repeatedly, where the first four iterations proceed as follows (to ease the presentation, we do not discuss $(K(A)+1, \mathsf{suff}_{\not\approx})$-smallness in detail):

1. The $\ll$-minimum pair extracted from Candidates is $\langle a, q\rangle$. Since Definitive is still empty, step 2(c) is satisfied with a trivial $(K(A)+1, \mathsf{suff}_{\not\approx})$-smallness test. Hence, Definitive $:= \{\langle a, q\rangle\}$, and the pair $\langle a, q\rangle$ and the current Definitive are used to instantiate two new feasible pairs for Candidates:

$$\mathsf{Candidates} := \{\langle a, q'\rangle\} \uplus \{\langle g(a,a), q\rangle,\ \langle g(a,a), q'\rangle\}.$$

2. The $\ll$-minimum pair extracted from Candidates is $\langle a, q'\rangle$. Now, Definitive is nonempty, but its subset $\{s \mid \langle s, q'\rangle \in \mathsf{Definitive}\}$ is, and thus, step 2(c) is

satisfied. Hence, $\mathsf{Definitive} := \{\langle a, q\rangle,\ \langle a, q'\rangle\}$. In this case, the instantiation does not produce any feasible pair: since it is required that the current pair $\langle a, q'\rangle$ is used in the instantiation, the only transition rule that we can use is $h(q, q') \xrightarrow{1 \not\approx 2} q_{\mathtt{accept}}$, but it is easy to see that the disequality constraint $1 \not\approx 2$ cannot be satisfied with the current feasible pairs in Definitive. Hence

$$\mathsf{Candidates} = \{\langle g(a, a), q\rangle,\ \langle g(a, a), q'\rangle\}.$$

3. The $\ll$-minimum pair extracted from Candidates is $\langle g(a, a), q\rangle$. Now, note that $\{s \mid \langle s, q\rangle \in \mathsf{Definitive} \wedge s \sim^A g(a, a)\}$ is empty since $\mathsf{suff}_{\not\approx} = \{\lambda, 1, 2\}$. Thus, again step 2(c) is satisfied. Hence, $\mathsf{Definitive} := \{\langle a, q\rangle, \langle a, q'\rangle, \langle g(a, a), q\rangle\}$, and the instantiation produces three new feasible pairs:

$$\mathsf{Candidates} := \{\langle g(a, a), q'\rangle\} \uplus \{\langle g(g(a, a), g(a, a)), q\rangle,\ \langle g(g(a, a), g(a, a)), q'\rangle,\ \langle h(g(a, a), a), q_{\mathtt{accept}}\rangle\}.$$

   At this point it would be possible to conclude $\mathcal{L}(A) \neq \emptyset$ due to the presence of the feasible pair $\langle h(g(a, a), a), q_{\mathtt{accept}}\rangle$ in Candidates. Nevertheless, the algorithm does not stop its execution until Candidates is empty.
4. The $\ll$-minimum pair extracted from Candidates is $\langle g(a, a), q'\rangle$. As before, step 2(c) is satisfied since $\{s \mid \langle s, q'\rangle \in \mathsf{Definitive} \wedge s \sim^A g(a, a)\}$ is empty. Hence, $\mathsf{Definitive} := \{\langle a, q\rangle,\ \langle a, q'\rangle,\ \langle g(a, a), q\rangle,\ \langle g(a, a), q'\rangle\}$, and the instantiation produces one new feasible pair:

$$\mathsf{Candidates} := \{\langle g(g(a, a), g(a, a)), q\rangle,\ \langle g(g(a, a), g(a, a)), q'\rangle,\ \langle h(g(a, a), a), q_{\mathtt{accept}}\rangle\} \uplus \{\langle h(a, g(a, a)), q_{\mathtt{accept}}\rangle\}.$$

In the next iteration of step 2, the algorithm will extract from Candidates the pair $\langle h(a, g(a, a)), q_{\mathtt{accept}}\rangle$ generated in the fourth iteration (which is, in fact, the $\ll$-minimum accepting pair of A) and insert it into Definitive. Hence, once Candidates is completely emptied, the algorithm will halt with output "NON-EMPTY," as expected. Note that, in the detailed iterations, the feasible pairs instantiated are all new, even though this is in general not the case: the instantiation might produce feasible pairs that are already in Candidates. (The new pairs are only guaranteed to be $\ll$-greater than all the feasible pairs in Definitive.)

Now it remains to prove that our algorithm is correct and terminates in the desired time.

LEMMA 6.7. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Algorithm 6.1 on input A takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$.*

Proof. Let A be $\langle Q, \Sigma, F, \Delta\rangle$ more explicitly written. First note that, by Definitions 5.10, 5.12, 5.16 and 6.3 and Lemma 5.19, $K(A)$ is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}| \cdot \log |A|)}$. Now, consider any maximal subset of Definitive with (term, state)-pairs having the same state and with all the terms belonging to the same equivalence class of $\sim^A$ and note that, since Algorithm 6.1 guarantees that such subset is $(K(A)+1, \mathsf{suff}_{\not\approx})$-small, by Lemma 4.13 it follows that its size is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot \log |A|)}$. By Lemma 5.11, there are $|Q| \cdot \mathcal{B}_{\mathrm{eq}}(A)$ of such subsets, and thus, we have that the size of Definitive is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot \log |A|)}$. Each time a pair is added to Definitive, new pairs are generated and added to Candidates. When this happens, the maximum number of new pairs that can be generated in the instantiation is bounded by $|\Delta| \cdot |\mathsf{Definitive}|^{|\mathsf{Pos}_{\mathrm{lhs}}|}$. Hence, the number of pairs inserted in Candidates during the whole execution is in

$2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$. Since each iteration of the algorithm removes a pair from Candidates, it follows that the number of iterations is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$.

It remains to prove that each iteration takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$. To avoid double exponential blowup, we consider a directed acyclic graph (DAG) representation for terms. More precisely, the algorithm uses a DAG as an internal data structure, where each node is labeled by a symbol $f \in \Sigma$ and has $\mathsf{arity}(f)$ ordered outedges. In this way, each node of the DAG implicitly represents a term over Σ, and thus, each of the terms considered by the algorithm is simply a reference to the appropriate node in the DAG. Additionally, we consider that the DAG is minimum, meaning that each two distinct nodes of the DAG represent different terms. (Note that each time a new node has to be inserted into the DAG, it can be checked in linear time with respect to the size of the DAG whether the DAG already contains a node representing the desired term.) This implies that the total size of the representation, i.e., the size of the DAG, equals the number of distinct terms generated by the algorithm, which is in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$. Now, we analyze the cost of performing each of the steps in the loop when using such an internal data structure. For step 2(a), it is necessary to select the minimum pair in Candidates with respect to $\ll$. To this end, it suffices to compute, for each two distinct nodes of the DAG representing terms t, t', whether $t \ll t'$. The cost of this computation is polynomial with respect to the size of the DAG when using a dynamic programming scheme, and hence, this step takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$. For step 2(c), $\sim^A$-equivalence with a term t must be checked: this consists in testing, for each node of the DAG representing a term t' and each two positions $p_1, p_2 \in \mathsf{suff}_{\not\approx}$, whether the subterms of t, t' pending at positions p_1, p_2 exist, and when they do, whether $t|_{p_1}, t|_{p_2}$ are represented by the same node of the DAG (i.e., are equal terms) if and only if $t'|_{p_1}, t'|_{p_2}$ are represented by the same node of the DAG (i.e., are equal terms). Also for step 2(c), it is necessary to test $(K(A)+1, \mathsf{suff}_{\not\approx})$-smallness: by Lemma 4.14, this requires at most $|\mathsf{Definitive}|^2 \cdot 2^{|\mathsf{suff}_{\not\approx}|} \cdot |\mathsf{suff}_{\not\approx}|$ equality comparisons between subterms, and such comparisons consist simply in checking whether the nodes of the DAG representing the involved subterms coincide. Thus, this step also takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$. Finally, for step 2(c)(ii), we need to insert into the DAG the newly instantiated terms, and also avoid the insertion of repeated elements in Candidates. As justified before, inserting a node into the DAG takes linear time with respect to the size of the DAG, as we need to keep the DAG minimum. Avoiding repeated elements in Candidates is straightforward: it suffices to check whether Candidates already contains a (term, state)-pair with the same state and referencing the same node of the DAG as the (term, state)-pair to be inserted. Hence, this also takes time in $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$, and we are done. ∎

We now prove Algorithm 6.1 to be correct.

LEMMA 6.8. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Let $\langle t, q \rangle \in \mathcal{T}(\Sigma) \times Q$ be a feasible (term, state)-pair of A satisfying that it is not generated by Algorithm* 6.1 *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* 6.1 *on input A.*

Proof. We proceed by induction on $\mathsf{height}(t)$. Let r be a run of A on t reaching q and let $l \xrightarrow{c} q$ be the rule applied at root position in r. Let $p_1, \ldots, p_m$ be the positions in $\mathsf{Pos}_Q(l)$. Note that necessarily $m > 0$, since otherwise the pair $\langle t, q \rangle$ is generated at step 1 of Algorithm 6.1. Let $t_1, \ldots, t_m$ be the terms $t|_{p_1}, \ldots, t|_{p_m}$ and $q_1, \ldots, q_m$ be the states $l(p_1), \ldots, l(p_m)$, respectively. Note that $\mathsf{height}(t_i) < \mathsf{height}(t)$ since $p_i \neq \lambda$ and that q_i is the state reached by the subrun $r|_{p_i}$ for all $i \in \{1, \ldots, m\}$.

Since $\langle t, q\rangle$ is not generated by Algorithm 6.1 on input A, there exists $i \in \{1, \ldots, m\}$ satisfying that $\langle t_i, q_i\rangle$ is either discarded or not generated. Note that such i must exist, since otherwise, $\langle t, q\rangle$ is generated at step 2(c)(ii) of Algorithm 6.1, contradicting the definition of $\langle t, q\rangle$.

In order to conclude, first assume that $\langle t_i, q_i\rangle$ is discarded. In this case, the statement holds by defining the pair $\langle t', q'\rangle$ of the lemma as $\langle t_i, q_i\rangle$. Next, assume that $\langle t_i, q_i\rangle$ is not generated. In this case, by induction hypothesis, there exists a feasible pair $\langle \hat{t}, \hat{q}\rangle \in \mathcal{T}(\Sigma) \times Q$ of A such that it is a piece of $\langle t_i, q_i\rangle$ and is discarded by Algorithm 6.1 on input A. Hence, the statement holds by defining the pair $\langle t', q'\rangle$ of the lemma as $\langle \hat{t}, \hat{q}\rangle$. ∎

We are finally ready to prove the soundness and completeness of Algorithm 6.1.

LEMMA 6.9. *Let A be a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$. Then, $\mathcal{L}(A)$ is empty if and only if Algorithm* 6.1 *on input A outputs "*`EMPTY`*."*

Proof. Let $A = \langle Q, \Sigma, F, \Delta\rangle$. The left-to-right direction follows trivially, since in this case it is not possible to generate a feasible pair $\langle t, q\rangle \in \mathcal{T}(\Sigma) \times Q$ of A satisfying that $q \in F$, and hence, the algorithm necessarily outputs "`EMPTY`." For the other direction, we proceed by contradiction by assuming that there exists an accepted term, but the algorithm cannot find any accepting pair of A and outputs "`EMPTY`." Let $\langle t, q\rangle \in \mathcal{T}(\Sigma) \times Q$ be the minimum accepting pair of A with respect to $\ll$. By assumption, $\langle t, q\rangle$ is either discarded or not generated by the algorithm. The former case is not possible, since it implies that $\mathsf{Definitive}$ contains at least one pair $\langle t', q\rangle$ such that $t' \sim^A t$, and therefore $\langle t', q\rangle$ is an accepting pair of A, contradicting the fact that the algorithm finds no accepting pairs of A. Hence, assume that $\langle t, q\rangle$ is not generated by the algorithm. By Lemma 6.8, it follows that there exists a feasible pair $\langle t', q'\rangle$ of A such that it is a piece of $\langle t, q\rangle$ 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 $\{s \mid \langle s, q'\rangle \in \mathsf{Definitive} \ \wedge \ s \sim^A t'\} \cup \{t'\}$. We know that $S - \{t'\}$ is $(K(A)+1, \mathsf{suff}_{\not\approx})$-small, but S is not. Hence, by Lemma 4.15, it follows that there exists a $\mathsf{suff}_{\not\approx}$-independent set of terms $\tilde{S} \subseteq S$ including t' and satisfying $|\tilde{S}| \geq K(A) + 1$. By definition, all the terms in $\tilde{S} - \{t'\}$ also have runs of A on them reaching the state q', are $\sim^A$-equivalent to t', and are smaller than t' with respect to $\ll$. By Corollary 6.4, it follows that $\langle t', q'\rangle$ is not a piece of the minimum accepting pair of A, contradicting the selection of $\langle t, q\rangle$. ∎

The next corollary follows from Lemmas 6.7 and 6.9.

COROLLARY 6.10. *The emptiness of the language recognized by a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A can be decided with time complexity $2^{\mathcal{O}(|\mathsf{suff}_{\not\approx}|^2 \cdot |\mathsf{Pos}_{\mathrm{lhs}}| \cdot \log |A|)}$.*

7. Consequences. The following theorem just states the same as Corollary 6.10 but without the detailed time complexity.

THEOREM 7.1. *Deciding the emptiness of the language recognized by a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A is in EXPTIME.*

THEOREM 7.2. *Deciding the finiteness of the language recognized by a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A is in EXPTIME.*

Proof. By Proposition 3.7, a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A' such that $\mathcal{L}(A')$ is empty if and only if $\mathcal{L}(A)$ is finite can be computed in exponential time with respect to $|A|$. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A')$, $\mathsf{n}_{\not\approx}(A')$, $|\mathsf{Pos}_{\mathrm{lhs}}(A')|$, and $|A'|$ are such that, by Corollary 6.10, the emptiness of $\mathcal{L}(A')$ can be decided in exponential time with respect to $|A|$. ∎

Now we give decidability results on the images of regular tree languages under tree homomorphisms.

THEOREM 7.3. *The inclusion problem for images of regular tree languages under tree homomorphisms, i.e., deciding* $H_1(\mathcal{L}(A_1)) \subseteq H_2(\mathcal{L}(A_2))$ *for* $\mathtt{TA}$ A_1, A_2 *and tree homomorphisms* H_1, H_2 *given as input, is EXPTIME-complete.*

Proof. By Proposition 3.4, two $\mathtt{TA}_{\mathrm{ihom}}$ A_1' and A_2' recognizing $H_1(\mathcal{L}(A_1))$ and $H_2(\mathcal{L}(A_2))$, respectively, can be computed in polynomial time with respect to the size of the input. By Proposition 3.5, a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A recognizing $\mathcal{L}(A_1') \cap \overline{\mathcal{L}(A_2')}$ can be computed in exponential time with respect to the size of the input. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A)$, $\mathsf{n}_{\not\approx}(A)$, $|\mathsf{Pos}_{\mathrm{lhs}}(A)|$, and $|A|$ are such that, by Corollary 6.10, the emptiness of $\mathcal{L}(A)$ can be decided in exponential time with respect to the size of the input. Thus, we conclude by noting that emptiness of $\mathcal{L}(A_1') \cap \overline{\mathcal{L}(A_2')}$ is equivalent to $H_1(\mathcal{L}(A_1)) \subseteq H_2(\mathcal{L}(A_2))$ and that the problem is EXPTIME-hard by Proposition 2.4. ∎

COROLLARY 7.4. *The equivalence problem for images of regular tree languages under tree homomorphisms, i.e., deciding* $H_1(\mathcal{L}(A_1)) = H_2(\mathcal{L}(A_2))$ *for* $\mathtt{TA}$ A_1, A_2 *and tree homomorphisms* H_1, H_2 *given as input, is EXPTIME-complete.*

THEOREM 7.5. *The finite difference problem for images of regular tree languages under tree homomorphisms, i.e., deciding the finiteness of* $H_1(\mathcal{L}(A_1)) - H_2(\mathcal{L}(A_2))$ *for* $\mathtt{TA}$ A_1, A_2 *and tree homomorphisms* H_1, H_2 *given as input, is EXPTIME-complete.*

Proof. By Propositions 3.4, 3.5, and 3.7, a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A such that $\mathcal{L}(A)$ is empty if and only if $H_1(\mathcal{L}(A_1)) \cap \overline{H_2(\mathcal{L}(A_2))}$ is finite can be computed in exponential time with respect to the size of the input. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A)$, $\mathsf{n}_{\not\approx}(A)$, $|\mathsf{Pos}_{\mathrm{lhs}}(A)|$, and $|A|$ are such that, by Corollary 6.10, the emptiness of $\mathcal{L}(A)$ can be decided in exponential time with respect to the size of the input. Thus, we conclude by noting that the finiteness of $H_1(\mathcal{L}(A_1)) \cap \overline{H_2(\mathcal{L}(A_2))}$ is equivalent to the finiteness of $H_1(\mathcal{L}(A_1)) - H_2(\mathcal{L}(A_2))$, and that the problem is EXPTIME-hard by Proposition 2.4. ∎

THEOREM 7.6. *The HOM problem is EXPTIME-complete.*

Proof. Assume a given $\mathtt{TA}$ A and a tree homomorphism H. By Propositions 3.4 and 3.6, a $\mathtt{TA}_{\mathrm{ihom},\not\approx}$ A' such that $\mathcal{L}(A')$ is empty if and only if $H(\mathcal{L}(A))$ is regular can be computed in exponential time with respect to $|A|$ and $|H|$. Moreover, the bounds on $\mathsf{h}_{\not\approx}(A')$, $\mathsf{n}_{\not\approx}(A')$, $|\mathsf{Pos}_{\mathrm{lhs}}(A')|$, and $|A'|$ are such that, by Corollary 6.10, the emptiness of $\mathcal{L}(A')$ can be decided in exponential time with respect to $|A|$ and $|H|$. Thus, we conclude by noting that the problem is EXPTIME-hard by Proposition 2.4. ∎

Our results have also implications in the context of term rewriting. The set of reducible terms of a term rewrite system can be described as the image of a regular tree language under a tree homomorphism, and the set of normal forms, i.e., the set of terms for which no rule can be applied, is just its complement. Thus, we can decide the inclusion and equality of such sets with respect to two given term rewrite systems in exponential time. Since ground reducibility is a particular case of such problems and it is shown EXPTIME-hard in [8], we conclude that these problems are EXPTIME-complete.

COROLLARY 7.7. *For any given term rewrite system* R, *let* $\mathsf{Red}(R)$ *and* $\mathsf{NF}(R)$ *be the set of reducible terms and the set of normal forms, respectively, with respect to* R. *Deciding* $\mathsf{Red}(R_1) = \mathsf{Red}(R_2)$, $\mathsf{Red}(R_1) \subseteq \mathsf{Red}(R_2)$, $\mathsf{NF}(R_1) = \mathsf{NF}(R_2)$, *and* $\mathsf{NF}(R_1) \subseteq \mathsf{NF}(R_2)$ *for given term rewrite systems* R_1, R_2 *is EXPTIME-complete.*

In [18], the question $\mathsf{Rel}(L_1) \subseteq L_2$ is shown to be decidable for given regular tree languages L_1, L_2 and where the relation Rel is defined in several ways according to a given term rewrite system R. Tree homomorphisms are used to describe the image of L_1 through this relation: two tree homomorphisms H_l and H_r and a tree language R_c are defined satisfying $\mathsf{Rel}(L_1) = H_r(H_l^{-1}(L_1) \cap R_c)$, so that deciding $\mathsf{Rel}(L_1) \subseteq L_2$ is done by testing $H_r(H_l^{-1}(L_1) \cap R_c) \subseteq L_2$. The tree homomorphisms H_l, H_r depend only on the rewrite system R. The tree language R_c depends also on the relation Rel. Our results allow us to improve the results in [18] where R_c is a regular tree language. These are when Rel is one of the following relations: the one rewriting step, the one parallel rewriting step, the one-pass innermost-outermost step for left-linear term rewrite systems, and the one-pass outermost-innermost step for right-linear term rewrite systems (see [18] for details). In those cases, we are able to extend the results to decide the question $\mathsf{Rel}_1(L_1) \subseteq \mathsf{Rel}_2(L_2)$. Analogously, tree homomorphisms $H_{1,l}, H_{1,r}, H_{2,l}, H_{2,r}$ and regular tree languages $R_{1,c}, R_{2,c}$ can be defined such that $\mathsf{Rel}_1(L_1) = H_{1,r}(H_{1,l}^{-1}(L_1) \cap R_{1,c})$ and $\mathsf{Rel}_2(L_2) = H_{2,r}(H_{2,l}^{-1}(L_2) \cap R_{2,c})$, so that deciding $\mathsf{Rel}_1(L_1) \subseteq \mathsf{Rel}_2(L_2)$ is done by testing $H_{1,r}(H_{1,l}^{-1}(L_1) \cap R_{1,c}) \subseteq H_{2,r}(H_{2,l}^{-1}(L_2) \cap R_{2,c})$. Under the given assumptions, $H_{1,l}^{-1}(L_1) \cap R_{1,c}$ and $H_{2,l}^{-1}(L_2) \cap R_{2,c}$ are regular languages. Thus, the above inclusion relates two images of regular tree languages under tree homomorphisms.

COROLLARY 7.8. *Deciding* $H_{1,r}(H_{1,l}^{-1}(L_1) \cap R_{1,c}) \subseteq H_{2,r}(H_{2,l}^{-1}(L_2) \cap R_{2,c})$ *is EXPTIME-complete for given tree homomorphisms* $H_{1,l}, H_{1,r}, H_{2,l}, H_{2,r}$ *and given regular tree languages* $L_1, L_2, R_{1,c}, R_{2,c}$.

COROLLARY 7.9. *Deciding* $\mathsf{Rel}_1(L_1) = \mathsf{Rel}_2(L_2)$ *and* $\mathsf{Rel}_1(L_1) \subseteq \mathsf{Rel}_2(L_2)$ *is EXPTIME-complete for given regular tree languages* L_1, L_2 *and a given term rewrite system* R, *where* $\mathsf{Rel}_1, \mathsf{Rel}_2$ *are defined as either the one rewriting step, the one parallel rewriting step, the one-pass innermost-outermost step if* R *is left-linear, or the one-pass outermost-innermost step if* R *is right-linear.*

8. Conclusion. We have proved EXPTIME-completeness of set inclusion, regularity (HOM problem), and finiteness of set difference for languages defined as images of regular tree languages under tree homomorphisms. Hence, we have determined the exact complexity of HOM and other problems that were already proved decidable in [21]. To this end, we have used some intermediate results from [21]. It would be interesting to study whether such intermediate results can be obtained in a simpler and clearer manner using the new class of tree automata presented here. Also, we have obtained simpler combinatoric arguments than the ones used in [8]. That paper proves decidability in exponential time of emptiness for the particular case of tree automata with disequality constraints. Hence, it could be interesting to study whether those proofs can be rewriten with the present approach in order to make them more accessible. In [10], the emptiness of deterministic and complete reduction automata is proved decidable. It could be also interesting to study whether our techniques can be applied to this problem in order to improve the obtained time complexity.

REFERENCES

[1] F. BAADER AND T. NIPKOW, *Term Rewriting and All That*, Cambridge University Press, New York, 1998.

[2] L. BARGUÑÓ, C. CREUS, G. GODOY, F. JACQUEMARD, AND C. VACHER, *Decidable classes of tree automata mixing local and global constraints modulo flat theories*, Log. Methods Comput. Sci., 9 (2013).

[3] P. B. Bendix and D. E. Knuth, *Simple word problems in universal algebras*, in Computational Problems in Abstract Algebra, Pergamon Press, Elmsford, NY, 1970, pp. 263–297.
[4] V. Benzaken, G. Castagna, H. Hosoya, B. C. Pierce, and S. Vansummeren, *XML type-checking*, in Encyclopedia of Database Systems, Springer, New York, 2009, pp. 3646–3650.
[5] B. Bogaert, F. Seynhaeve, and S. Tison, *The recognizability problem for tree automata with comparison between brothers*, in Proceedings of Foundations of Software Science and Computation Structures (FOSSACS), 1999, pp. 150–164.
[6] A.-C. Caron, F. Seynhaeve, S. Tison, and M. Tommasi, *Deciding the satisfiability of quantifier free formulae on one-step rewriting*, in Proceedings of Rewriting Techniques and Applications (RTA), 1999, pp. 103–117.
[7] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, *Tree Automata Techniques and Applications*, http://www.grappa.univ-lille3.fr/tata, 2007.
[8] H. Comon and F. Jacquemard, *Ground reducibility is EXPTIME-complete*, Inform. and Comput., 187 (2003), pp. 123–153.
[9] C. Creus, A. Gascón, G. Godoy, and L. Ramos, *The HOM problem is EXPTIME-complete*, in Proceedings of the IEEE Symposium on Logic in Computer Science (LICS), IEEE, 2012, pp. 255–264.
[10] M. Dauchet, A.-C. Caron, and J.-L. Coquidé, *Automata for reduction properties solving*, J. Symbolic Comput., 20 (1995), pp. 215–233.
[11] M. Dauchet, S. Tison, and M. Tommasi, *Réduction de la non-linéarité des morphismes d'arbres Recognizable tree-languages and non-linear morphisms*, Theoret. Comput. Sci., 281 (2002), pp. 219–233.
[12] J. Doner, *Tree acceptors and some of their applications*, J. Computer Syst. Sci., 4 (1970), pp. 406–451.
[13] J. Engelfriet, *Bottom-up and top-down tree transformations—A comparison*, Math. Syst. Theory, 9 (1975), pp. 198–231.
[14] E. Filiot, J.-M. Talbot, and S. Tison, *Tree automata with global constraints*, Internat. J. Found. Comput. Sci., 21 (2010), pp. 571–596.
[15] Z. Fülöp, *Undecidable properties of deterministic top-down tree transducers*, Theoret. Comput. Sci., 134 (1994), pp. 311–328.
[16] F. Gécseg and M. Steinby, *Tree Automata*, Akadémiai Kiadó, 1984.
[17] F. Gécseg and M. Steinby, *Tree languages*, in Handbook of Formal Languages, G. Rozenberg and A. Salomaa, eds., Vol. 3, Springer, New York, 1997, pp. 1–68.
[18] R. Gilleron and S. Tison, *Regular tree languages and rewrite systems*, Fund. Inform., 24 (1995), pp. 157–174.
[19] R. Gilleron, S. Tison, and M. Tommasi, *Set constraints and automata*, Inform. and Comput., 149 (1999), pp. 1–41.
[20] 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 J. Comput., 40 (2011), pp. 446–464.
[21] G. Godoy and O. Giménez, *The HOM problem is decidable*, J. ACM, 60 (2013), 23.
[22] G. Godoy, S. Maneth, and S. Tison, *Classes of tree homomorphisms with decidable preservation of regularity*, in Proceedings of Foundations of Software Science and Computation Structures (FOSSACS), 2008, pp. 127–141.
[23] D. Hofbauer and M. Huber, *Computing linearizations using test sets*, in Proceedings of the Third International Workshop on Conditional Term Rewriting Systems (CTRS), 1992, pp. 287–301.
[24] G. Kucherov and M. Tajine, *Decidability of regularity and related properties of ground normal form languages*, Inform. and Comput., 118 (1995), pp. 91–100.
[25] J. Mezei and J. B. Wright, *Algebraic automata and context-free sets*, Inform. Control, 11 (1967), pp. 3–29.
[26] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, *Taxonomy of XML schema languages using formal language theory*, ACM Trans. Internet Technol., 5 (2005), pp. 660–704.
[27] J. W. Thatcher, *Transformations and translations from the point of view of generalized finite automata theory*, in Proceedings of the Symposium on Theory of Computing (STOC), 1969, pp. 129–142.
[28] S. Vágvölgyi and R. Gilleron, *For a rewrite system it is decidable whether the set of irreducible, ground terms is recognizable*, Bull. EATCS, 48 (1992), pp. 197–209.