Logical Methods in Computer Science
Vol. 10(1:7)2014, pp. 1–33
www.lmcs-online.org

Submitted Oct. 30, 2012
Published Feb. 12, 2014

THE COMPLEXITY OF THE FIRST-ORDER THEORY OF GROUND TREE REWRITE GRAPHS

STEFAN GÖLLER [a] AND MARKUS LOHREY [b]

[a] Universität Bremen, Germany
e-mail address: goeller@informatik.uni-bremen.de

[b] Universität Siegen, Germany
e-mail address: lohrey@eti.uni-siegen.de

Abstract. The uniform first-order theory of ground tree rewrite graphs is the set of all pairs consisting of a ground tree rewrite system and a first-order sentence that holds in the graph defined by the ground tree rewrite system. We prove that the complexity of the uniform first-order theory of ground tree rewrite graphs is in $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$. Providing a matching lower bound, we show that there is some fixed ground tree rewrite graph whose first-order theory is hard for $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$ with respect to logspace reductions. Finally, we prove that there exists a fixed ground tree rewrite graph together with a single unary predicate in form of a regular tree language such that the resulting structure has a non-elementary first-order theory.

1. Introduction

A ground tree rewrite system is a term rewrite system where rules do not contain variables (neither on the left-hand side nor on the right-hand side). So, rules replace subtrees by subtrees. Ground tree rewrite systems were first studied in the term rewriting community [7, 12, 13], where they are also known as ground term rewrite systems.

Recently, ground tree rewrite systems were also studied in the context of verification of infinite state systems [30]. The main motivation for this is that ground tree rewrite systems can be seen as a generalization of pushdown systems. These are a natural abstraction of sequential recursive programs. Rules of a ground tree rewrite system can be applied concurrently at different positions of a tree. This allows to model recursive progams with the additional ability to spawn new subthreads that are hierarchically structured, which in turn may terminate and return some values to their parents.

One of the most important and oldest decidability results for ground tree rewrite systems was shown more than 20 years ago by Dauchet and Tison [13]: The transition graph of a ground tree rewrite system (called a ground tree rewrite graph in the following) has a

2012 ACM CCS: [**Theory of computation**]: Formal languages and automata theory—Formalisms—Rewrite systems; Formal languages and automata theory—Tree languages.

Key words and phrases: Ground tree rewriting, complexity of first-order theories, alternating complexity classes.

DOI:10.2168/LMCS-10(1:7)2014

decidable first-order theory. Actually, Dauchet and Tison even showed that the first-order theory of a ground tree rewrite graph extended by the transitive closure of the edge relation is decidable (one also says that first-order logic with reachability is decidable for ground tree rewrite graphs). The proof of Dauchet and Tison uses a tree automata construction, which yields a non-elementary algorithm. This leads to the question of complexity. While the first-order theory of a ground tree rewrite graph extended by the transitive closure of the edge relation may have non-elementary complexity (this holds already for the infinite binary tree, which is a pushdown graph [42]), the precise complexity of the first-order theories of ground tree rewrite graphs remained open. As the main contribution of this paper we solve this problem. We prove the following:

- The first-order theory of every ground tree rewrite graph belongs to the complexity class $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$ (doubly exponential alternating time, where the number of alternations is bounded linearly), where n is the length of the input formula.
- There exists a fixed ground tree rewrite graph with an $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$-complete first-order theory.

The upper bound of $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$ even holds uniformly, which means that the ground tree rewrite system may be part of the input, i.e., n is the sum of the length of the input formula and the length of the description of the ground tree rewrite system. Let us remark that the complexity class $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$ appears also in other contexts. For instance, Presburger Arithmetic (the first-order theory of $(\mathbb{N}, +)$) is known to be complete for $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$ [2], see [11] for similar results.

The upper bound of $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$ is shown by the method of Ferrante and Rackoff [16]. Basically, the idea is to show the existence of a winning strategy of the duplicator in an Ehrenfeucht-Fraïssé game, where the duplicator chooses "small" (w.r.t. to a predefined norm) elements. This method is one of the main tools for proving upper bounds for FO-theories. We divide the upper bound proof into two steps. In a first step, we will reduce the FO-theory for a ground tree rewrite graph to the FO-theory for a very simple word rewrite graph, where all word rewrite rules replace one symbol by another symbol. The alphabet will consist of all trees, whose size is bounded by a singly exponential function in the input size (hence, the alphabet size is doubly exponential in the input size; this is the reason for the doubly exponential time bound). Basically, we obtain a word over this alphabet from a tree t by cutting off some upward-closed set C in the tree and taking the resulting sequence of trees. Intuitively, the set C consists of all nodes u of t such that the subtree rooted in u is "large". Here, "large" has to be replaced by a concrete value $m \in \mathbb{N}$ such that a sequence of n rewrite steps applied to a tree t cannot touch a node from the upward-closed set C. Clearly, m depends on n. In our context, n will be exponential in the input size and so will m. In a second step, we provide an upper bound for the FO-theory of a word rewrite graph of the above form.

Perhaps it is worth mentioning that for proving our upper bound result one cannot make use of Gaifman's locality theorem [18] since the resulting formulas in Gaifman normal form can become non-elementary in the size of the original first-order formula [14]. An elementary upper bound on the size of Gaifman normal formulas was shown for structures of bounded degree in [14]. However, ground tree rewrite graphs have unbounded degree. This also the reason why Hanf's theorem [23] does not seem to be of any use for our problem.

For the lower bound, we prove in a first step hardness for $\mathsf{2NEXP}$ (doubly exponential nondeterministic time). This is achieved by an encoding of a $(2^{2^n} \times 2^{2^n})$ tiling problem.

In this tiling problem, we are given a word w of length n over some fixed set of tiles, and it is asked, whether this word can be completed to a tiling of an array of size $(2^{2^n} \times 2^{2^n})$, where the word w is an initial part of the first row. There exists a fixed set of tiles, for which this problem is $\mathsf{2NEXP}$-complete. From this fixed set of tiles, we construct a fixed ground tree rewrite graph such that the following holds: From a given word w of length n over the tiles, one can construct (in logspace) a first-order formula that evaluates to true in our fixed ground tree rewrite graph if and only if the word w is a positive instance of the $(2^{2^n} \times 2^{2^n})$ tiling problem. Our construction is inspired by [20], where it is shown that the model checking problem for a fragment of the logic EF (consisting of those EF-formulas, where on every path of the syntax tree at most one EF-operator occurs) over ground tree rewrite graphs is complete for the class $\mathsf{P}^{\mathsf{NEXP}}$. In a second step, we show that our $\mathsf{2NEXP}$ lower bound can easily be lifted to $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$. For this, we have to consider an alternating version of the $(2^{2^n} \times 2^{2^n})$ tiling problem.

We conclude the paper with a proof sketch for the following result: There exists a fixed ground tree rewrite graph together with a single unary predicate in form of a regular tree language such that the resulting structure has a non-elementary first-order theory. This result is shown by a reduction from first-order satisfiability of finite binary words, which is non-elementary [42]. It should be noted that the first-order theory of a pushdown graph extended by regular unary predicates still has an elementary first-order theory (it is an automatic structure of bounded degree, hence its first-order theory belongs to $\mathsf{2EXPSPACE}$ by a result from [27]).

A short version of this paper appeared in [22].

2. Related work

2.1. **Other decidability and complexity results for ground tree rewrite systems.**

Other important algorithmic problems that are decidable for ground tree rewrite systems are:

- confluence [12, 37], which in fact can be decided in polynomial time [10, 19],
- reachability [7, 15],[1] recurrent reachability [30, 31], and recurrent reachability with multiple regular fairness constraints [44],
- fair termination [43], and
- model checking certain fragments of LTL [45, 44].

The decidability of first-order logic with reachability for ground tree rewrite graphs implies that model checking of the CTL-fragment EF is decidable for ground tree rewrite graphs; the precise complexity was recently shown to be non-elementary [20].

[1] Actually, Brainerd [7] showed that a set of trees is regular if and only if it is the set of trees that can be reached from a single tree via a ground tree rewriting system, where both translations are effective. This generalizes a result of Büchi for strings.

2.2. **Pushdown graphs.** As remarked above, ground tree rewrite systems generalize pushdown systems. Muller and Schupp proved that every pushdown graph (the transition graph of a pushdown system) has a decidable monadic second-order (MSO) theory [36]. MSO extends first-order logic by the ability to quantify over subsets of the universe. Most temporal logics (e.g. LTL, CTL, modal μ-calculus) can be translated into MSO and are therefore decidable over pushdown graphs. Precise complexity results can be found in [5, 36, 49, 50].

Löding proved in [29] that a ground tree rewrite graph has bounded tree width if and only if it is a pushdown graph.

2.3. **Algorithmic limitations.** Ground tree rewrite graphs do not share all the nice algorithmic properties of pushdown graphs. For instance, the infinite grid is easily seen to be (embeddable into) a ground tree rewrite graph, which implies that ground tree rewrite graphs with an undecidable MSO-theory exist. In fact, most linear-time and branching-time temporal logics such as LTL and CTL have undecidable model checking problems over ground tree rewrite graphs (cf. [30, 44]).

Concerning the first-order theory, mild generalizations of ground tree rewrite systems lead to undecidable first-order theories. Undecidability holds for linear and non-erasing term rewrite systems [46], right ground Noetherian rewrite systems [33], and linear canonical rewrite systems [48]. In all these papers, undecidability is shown for fragments of first-order logic with only one quantifier alternation.

2.4. **Formalisms related to ground tree rewrite systems.** Several other extensions of pushdown systems with multithreading capabilities have been considered in [6, 24, 34, 39]. Among these extensions, the class of process rewrite systems [34], which generalize both Petri nets and pushdown systems by providing hierarchical structures to threads, seem to have tight connections with ground tree rewrite systems. Lugiez and Schnoebelen proved decidability of various first-order logics on PA-processes by using tree-automata techniques [32]. Mayr's process rewrite systems hierarchy [35] was recently refined via ground tree rewrite systems [21].

Recently, Lin extended ground-tree rewrite systems with a finite control unit that is acyclic but with possible self-loops, so called weakly-extended ground tree rewrite systems [28]. It is shown that reachability, recurrent reachability and (the complement of) model checking deterministic LTL is NP-complete for this extension.

The class of ground tree rewrite graphs is contained in the class of tree automatic structures [3], whose FO-theories are (non-elementarily) decidable. In [27], it is shown that (i) for every tree automatic structure of bounded degree (which means that the Gaifman-graph has bounded degree) the FO-theory belongs to $\mathsf{3EXPTIME}$ and that there is a fixed tree automatic structure of bounded degree with a $\mathsf{3EXPTIME}$-complete FO-theory. Note that in general, ground tree rewrite graphs are *not* of bounded degree.

2.5. **Applications of the method of Ferrante and Rackoff.** Recall that the method of Ferrante and Rackoff is the main technical tool in our proof that the first-order theory of every ground tree rewrite graph belongs to the complexity class $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$. Further applications of this technique in computer science can be found in [40] (for the theory of queues) and in [25] (for nested pushdown trees).

3. Preliminaries

By $\mathbb{Z}$ we denote the *integers* and by $\mathbb{N} = \{0, 1, \ldots\}$ the set of *non-negative integers*. For $i, j \in \mathbb{Z}$ we define the interval $[i, j] = \{i, i+1, \ldots, j\}$ and $[j] = [0, j]$.

For an alphabet A (possibly infinite), we denote with $A^+ = A^* \setminus \{\varepsilon\}$ the set of all non-empty words over A. The length of the word $w \in A^*$ is denoted by $|w|$. For $B \subseteq A$, we denote with $|w|_B$ the number of occurrences of symbols from B in the word w.

Let $f : A \to B$ be a mapping. For $A' \subseteq A$, we denote with $f{\restriction}A' : A' \to B$ the restriction of f to A'. For sets A, B, C (where A and B may have a non-empty intersection) and two mappings $f : A \to C$ and $g : B \to C$, we say that f and g are *compatible* if $f{\restriction}(A \cap B) = g{\restriction}(A \cap B)$. Finally, for mappings $f : A \to C$ and $g : B \to C$ with $A \cap B = \emptyset$, we define $f \uplus g : A \cup B \to C$ as the mapping with $(f \uplus g)(a) = f(a)$ for $a \in A$ and $(f \uplus g)(b) = g(b)$ for $b \in B$.

3.1. Complexity theory.

We will deal with alternating complexity classes, see [8, 38] for more details. An *alternating Turing-machine* is a nondeterministic Turing-machine, where the set of states is partitioned into existential and universal states. A configuration with a universal (resp. existential) state is *accepting* if every (resp. some) successor configuration is accepting. An *alternation* in a computation of an alternating Turing-machine is a transition from a universal state to an existential state or vice versa. For functions $t(n)$ and $a(n)$ with $a(n) \leq t(n)$ for all $n \geq 0$ let $\mathsf{ATIME}(t(n), a(n))$ denote the class of all problems that can be decided on an alternating Turing-machine in time $t(n)$ with at most $a(n)$ alternations. It is known that $\mathsf{ATIME}(t(n), t(n))$ is contained in $\mathrm{DSPACE}(t(n))$ if $t(n) \geq n$ [8].

3.2. Labelled graphs.

A (directed) *graph* is a pair $(V, \to)$, where V is a set of *nodes* and $\to \subseteq V \times V$ is a binary relation. A *labelled graph* is a tuple $\mathfrak{G} = (V, \Sigma, \{\xrightarrow{a} \mid a \in \Sigma\})$, where V is a set of *nodes*, Σ is a finite set of *actions*, and $\xrightarrow{a}$ is a binary relation on V for all $a \in \Sigma$. We note that (labelled) graphs may have infinitely many nodes. For $u, v \in V$, we define $d_{\mathfrak{G}}(u, v)$ as the length of a shortest undirected path between u and v in the graph $(V, \bigcup_{a \in \Sigma} \xrightarrow{a})$. For $n \in \mathbb{N}$ and $u \in V$ let $S_n(\mathfrak{G}, u) = \{v \in V \mid d_{\mathfrak{G}}(u, v) \leq n\}$ be the *sphere* of radius n around u. Moreover, for $u_1, \ldots, u_k \in V$ let $S_n(\mathfrak{G}, u_1, \ldots, u_k) = \bigcup_{1 \leq i \leq k} S_n(\mathfrak{G}, u_i)$. We identify $S_n(\mathfrak{G}, u_1, \ldots, u_k)$ with the substructure of $\mathfrak{G}$ induced by the set $S_n(\mathfrak{G}, u_1, \ldots, u_k)$, where in addition every u_i ($1 \leq i \leq k$) is added as a constant. For two labelled graphs $\mathfrak{G}_1$ and $\mathfrak{G}_2$ with node set V_1 and V_2, respectively, and nodes $u_1, \ldots, u_k \in V_1$, $v_1, \ldots, v_k \in V_2$, we will consider isomorphisms $f : S_n(\mathfrak{G}_1, u_1, \ldots, u_k) \to S_n(\mathfrak{G}_2, v_1, \ldots, v_k)$. Such an isomorphism has to map u_i to v_i. We write $S_n(\mathfrak{G}_1, u_1, \ldots, u_k) \cong S_n(\mathfrak{G}_2, v_1, \ldots, v_k)$ if there is an isomorphism $f : S_n(\mathfrak{G}_1, u_1, \ldots, u_k) \to S_n(\mathfrak{G}_2, v_1, \ldots, v_k)$.

Lemma 3.1. *Let $\mathfrak{G}_1, \mathfrak{G}_2$ be labelled graphs with the same set of actions and node sets V_1 and V_2, respectively. Let $\overline{u} \in V_1^k$, $\overline{v} \in V_2^k$, $u \in V_1$, and $v \in V_2$ such that $u \notin S_{2n+1}(\mathfrak{G}_1, \overline{u})$ and $v \notin S_{2n+1}(\mathfrak{G}_2, \overline{v})$. Finally, let $f : S_n(\mathfrak{G}_1, \overline{u}) \to S_n(\mathfrak{G}_2, \overline{v})$ and $f' : S_n(\mathfrak{G}_1, u) \to S_n(\mathfrak{G}_2, v)$ be isomorphisms. Then $f \uplus f' : S_n(\mathfrak{G}_1, u, \overline{u}) \to S_n(\mathfrak{G}_2, v, \overline{v})$ is an isomorphism as well.*

Proof. The lemma is obvious, once one realizes that the condition $u \notin S_{2n+1}(\mathfrak{G}_1, \overline{u})$ implies that the spheres $S_n(\mathfrak{G}_1, \overline{u})$ and $S_n(\mathfrak{G}_1, u)$ are disjoint and that there is no edge between the two spheres (and similarly for the spheres $S_n(\mathfrak{G}_2, \overline{v})$ and $S_n(\mathfrak{G}_2, v)$). $\square$

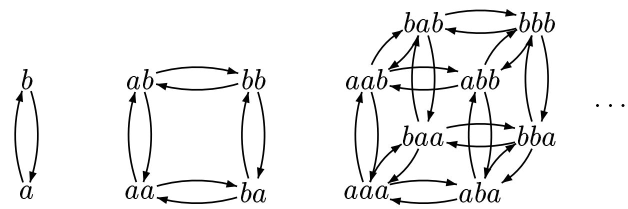


Figure 1: A finite portion of the relation $\to$ from Example 3.2 extended to $\{a, b\}^+$.

Later, we have to lift a relation $\to$ on a set A to a larger set. We will denote this new relation again by $\to$. Two constructions will be needed. Assume that $\to$ is a binary relation on a set A and let $A \subseteq B$. We lift $\to$ to the set B^+ of non-empty words over B as follows: For all $u, v \in B^+$, we have $u \to v$ if and only if there are $x, y \in B^*$ and $a, b \in A$ such that $a \to b$ and $u = xay$, $v = xby$. Note that this implies $|u| = |v|$. The second construction lifts $\to \subseteq A \times A$ from A to $\mathbb{N} \times A$ as follows: For $a, b \in A$ and $m, n \in \mathbb{N}$ let $(m, a) \to (n, b)$ if and only if $m = n$ and $a \to b$. Note that $(\mathbb{N} \times A, \to)$ consists of $\aleph_0$ many disjoint copies of $(A, \to)$. Moreover, $((A \cup \{\$\})^+ \setminus \{\$\}^+, \to)$ (where $\$ \notin A$ is a new symbol) is isomorphic to $(\mathbb{N} \times A^+, \to)$.

Example 3.2. For the relation $\to = \{(a, b), (b, a)\}$ the corresponding relation on $\{a, b\}^+$ is shown in Figure 1. The relation $\to$ lifted to $\mathbb{N} \times \{a, b\}$ is simply the disjoint union of all 2-cycles

$$(a, n) \rightleftarrows (b, n)$$

for all $n \in \mathbb{N}$.

For a labelled graph $\mathfrak{G} = (V, \Sigma, \{\xrightarrow{a} \mid a \in \Sigma\})$, we define the labelled graph

$$\mathfrak{G}^+ = (V^+, \Sigma, \{\xrightarrow{a} \mid a \in \Sigma\}). \tag{3.1}$$

Note that by the above definition, $\xrightarrow{a}$ is lifted to a relation on V^+.

3.3. **First-order logic.** We will consider first-order logic (briefly FO) with equality over labelled graphs. Thus, for a set Σ of actions, we have for each $a \in \Sigma$ a binary relation symbol $a(x, y)$ in our signature. The meaning of $a(x, y)$ is of course $x \xrightarrow{a} y$. If $\varphi(x_1, \ldots, x_n)$ is a first-order formula with free variables $x_1, \ldots, x_n$, $\mathfrak{G} = (V, \Sigma, \{\xrightarrow{a} \mid a \in \Sigma\})$ is a labelled graph, and $v_1, \ldots, v_n \in V$, then we write $\mathfrak{G} \models \varphi(v_1, \ldots, v_n)$ if φ evaluates to true in $\mathfrak{G}$, when variable x_i is instantiated by v_i ($1 \le i \le n$). The *first-order theory* (briefly *FO-theory*) of a labelled transition graph $\mathfrak{G}$ is the set of all first-order sentences (i.e., first-order formulas without free variables) φ with $\mathfrak{G} \models \varphi$. In the final Section 6, we will consider the first-order theory of a labelled graph with an additional unary predicate. The *quantifier rank* of a first-order formula is the maximal number of nested quantifiers in φ. We will need the following well known lemma, which goes back to work of Fischer and Rabin [17].

Lemma 3.3. *Let Σ be a set of actions. Given a first-order formula $\theta(x, y)$ of quantifier rank $\mathsf{qr}(\theta)$ and a binary-coded integer j (let m be the number of 1-bits in the binary representation of j), one can compute in logspace a first-order formula $\theta^j(x, y)$ of quantifier rank $O(\log(j) + \mathsf{qr}(\theta))$ and size $O(m \cdot \log(j) + m \cdot |\theta|)$ such that for every labelled graph $\mathfrak{G} = (V, \Sigma, \{\xrightarrow{a} \mid a \in \Sigma\})$ and all nodes $u, v \in V$ we have: $\mathfrak{G} \models \theta^j(u, v)$ if and only if there is a directed path of length j from u to v in the graph $(V, \{(s, t) \mid \mathfrak{G} \models \theta(s, t)\})$.*

Proof. Before we define $\theta^j(x,y)$, let us inductively define for each $k \in \mathbb{N}$ a formula $\psi_k(x,y)$ such that for all $u, v \in V$ we have $\mathfrak{G} \models \psi_k(u,v)$ if and only if there is a directed path of length 2^k from u to v in the graph $(V, \{(s,t) \mid \mathfrak{G} \models \theta(s,t)\})$. We define

$$\begin{aligned}\psi_0(x,y) &= \theta(x,y), \text{ and}\\ \psi_k(x,y) &= \exists z \forall u, v \Big(\big((u = x \wedge v = z) \vee (u = z \wedge v = y)\big) \to \psi_{k-1}(u,v) \Big) \text{ for } k \geq 1.\end{aligned}$$

Note that the size of $\psi_k(x,y)$ is $O(k + |\theta|)$ and the quantifier rank is $3k + \mathsf{qr}(\theta)$.

Let $U \subseteq \mathbb{N}$ be the set of all positions of the binary representation of j whose bit is set to 1, i.e., $j = \sum_{i \in U} 2^i$. Let $m = |U|$ and let $h_1, \ldots, h_m$ be some enumeration of U. We can now define $\theta^j(x,y)$ as

$$\theta^j(x,y) \quad = \quad \exists x_1, \ldots, x_{m+1} \Big(x_1 = x \ \wedge \ x_{m+1} = y \ \wedge \bigwedge_{i \in [1,m]} \psi_{h_i}(x_i, x_{i+1}) \Big).$$

From the binary representation of j, we can easily compute $\theta^j(x,y)$. Moreover, the size of $\theta^j(x,y)$ is bounded by $O(m \cdot \log(j) + m \cdot |\theta|)$ and the quantifier rank is bounded by $O(\log(j) + \mathsf{qr}(\theta))$. ☐

One of most successful techniques for proving upper bounds for the complexity of first-order theories is the method of Ferrante and Rackoff [16]. We will apply this method in Section 4.2. The following result is shown in [16].[2]

Theorem 3.4. *Let $\mathfrak{G}$ be a labelled graph, and let V be the set of nodes of $\mathfrak{G}$. Assume that for every node $v \in V$ we have a norm $|v| \in \mathbb{N}$ (in our application, V will be a set of words and the norm of a word will be its length). Let $V_n = \{v \in V \mid |v| \leq n\}$. Moreover, for $k, \ell \geq 0$, let $\equiv_{k,\ell}$ be an equivalence relation on the set V^k and let $H : \mathbb{N}^2 \to \mathbb{N}$ be a function such that the following properties hold for all $k, \ell \in \mathbb{N}$, $\overline{u}, \overline{v} \in V^k$:*

(a) *If $\overline{u} \equiv_{k,0} \overline{v}$, then $\overline{u}$ and $\overline{v}$ satisfy the same quantifier-free formulas in the structure $\mathfrak{G}$.*
(b) *If $\overline{u} \equiv_{k,\ell} \overline{v}$ and $\ell > 0$, then for all $u \in V$ there exists $v \in V_{H(k,\ell)}$ with $(\overline{u}, u) \equiv_{k+1,\ell-1} (\overline{v}, v)$.*

Then, for every quantifier-free formula $\psi(x_0, \ldots, x_\ell)$ and all quantifiers $Q_0, \ldots, Q_\ell \in \{\exists, \forall\}$ we have that $\mathfrak{G} \models Q_0 x_0 \cdots Q_\ell x_\ell : \psi(x_0, \ldots, x_\ell)$ if and only if

$$\mathfrak{G} \models Q_0 x_0 \in V_{H(0,\ell)} Q_1 x_1 \in V_{H(1,\ell-1)} \cdots Q_\ell x_\ell \in V_{H(\ell,0)} : \psi(x_0, \ldots, x_\ell).$$

We will use Theorem 3.4 in Section 4.2, where the function $H(k,\ell)$ will be exponential in $k + \ell$.

3.4. **Trees.** Let $\preceq$ denote the prefix order on $\mathbb{N}^*$, i.e., $x \preceq y$ for $x, y \in \mathbb{N}^*$ if there is some $z \in \mathbb{N}^*$ such that $y = xz$. A set $D \subseteq \mathbb{N}^*$ is called *prefix-closed* if for all $x, y \in \mathbb{N}^*$, $x \preceq y \in D$ implies $x \in D$. A *ranked alphabet* is a collection of finite and pairwise disjoint alphabets $A = (A_i)_{i \in [k]}$ for some $k \geq 0$ such that $A_0 \neq \emptyset$. For simplicity we identify A with $\bigcup_{i \in [k]} A_i$. A *ranked tree* (over the ranked alphabet A) is a mapping $t : D_t \to A$, where $D_t \subseteq [1,k]^*$ satisfies the following:

- D_t is non-empty, finite, and prefix-closed, and

[2]The actual statement in [16] is stronger, but for our purpose the weaker statement in Theorem 3.4 is sufficient.

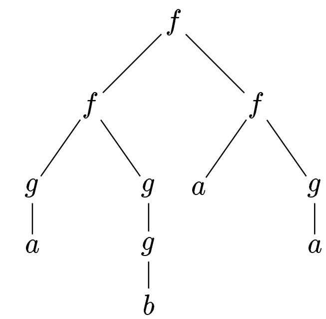


Figure 2: A tree s

- for each $x \in D_t$ with $t(x) \in A_i$ we have $x1, \ldots, xi \in D_t$ and $xj \notin D_t$ for each $j > i$.

We say that D_t is the *domain* of t and call its elements *nodes*. In case $t(x) \in A_2$ for some node x, then $x1$ is the *left child* and $x2$ the *right child* of x. A *leaf* of t is a node x with $t(x) \in A_0$. An *internal node* of t is a node, which is not a leaf. We also refer to $\varepsilon \in D_t$ as the *root* of t. By Trees_A we denote the set of all ranked trees over the ranked alphabet A. Define $\mathsf{size}(t)$ as the number of nodes in a tree t. It is easy to show that the number of all trees from Trees_A of size at most n is bounded by $|A|^n$.

Example 3.5. Assume $A_0 = \{a, b\}$, $A_1 = \{g\}$, and $A_2 = \{f\}$. Figure 2 shows a tree $s \in \mathsf{Trees}_A$ with $\mathsf{size}(s) = 11$. The domain D_s of this tree is

$$\{\varepsilon, 1, 2, 11, 12, 21, 22, 111, 121, 1211, 221\}.$$

Let t be a ranked tree and let x be a node of t. For each $x \in [1, k]^*$ we define $xD_t = \{xy \in [1, k]^* \mid y \in D_t\}$ and $x^{-1}D_t = \{y \in [1, k]^* \mid xy \in D_t\}$. By $t^{\downarrow x}$ we denote the *subtree of* t with root x, i.e., the tree with domain $D_{t^{\downarrow x}} = x^{-1}D_t$ defined as $t^{\downarrow x}(y) = t(xy)$. Let $s, t \in \mathsf{Trees}_A$ and let x be a node of t. We define $t[x/s]$ to be the tree that is obtained by replacing $t^{\downarrow x}$ in t by s, more formally $D_{t[x/s]} = (D_t \setminus xD_{t^{\downarrow x}}) \cup xD_s$ with

$$t[x/s](y) = \begin{cases} t(y) & \text{if } y \in D_t \setminus xD_{t^{\downarrow x}} \\ s(z) & \text{if } y = xz \text{ with } z \in D_s. \end{cases}$$

For two ranked trees s and t, let $\mathsf{diff}(s, t) = |D_s \setminus D_t|$. Thus $\mathsf{diff}(s, t)$ is the number of nodes that belong to the tree s but not to the tree t.

Example 3.6. Consider the tree s from Figure 2 and the tree t from Figure 3. We have

$$D_s \setminus D_t = \{11, 12, 22, 111, 121, 1211, 221\}$$

and hence $\mathsf{diff}(s, t) = 7$.

Let C be a prefix-closed subset of D_t. We define the string of subtrees $t \setminus C$ as follows: If $C = \emptyset$, then $t \setminus C = t$. If $C \neq \emptyset$, then $t \setminus C = t^{\downarrow v_1} \cdots t^{\downarrow v_m}$, where $v_1, \ldots, v_m$ is a list of all nodes from $((C \cdot \mathbb{N}) \cap D_t) \setminus C$ in lexicographic order. Intuitively, we remove from the tree t the prefix-closed subset C and list all remaining maximal subtrees. For $n \in \mathbb{N}$ and a tree t we define the prefix-closed subset $\mathsf{up}(t, n) \subseteq D_t$ as

$$\mathsf{up}(t, n) = \{v \in D_t \mid \mathsf{size}(t^{\downarrow v}) > n\}.$$

Note that $t \setminus \mathsf{up}(t, n)$ is a list of all maximal subtrees of size at most n in t, listed in lexicographic order.

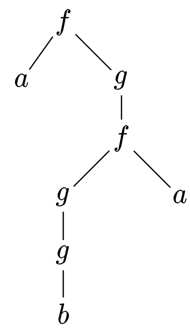


Figure 3: A chain t

Example 3.7. Consider the tree s from Figure 2. Then

$$C = \{\varepsilon, 1, 2, 12\} \subseteq D_s$$

is prefix-closed. We have

$$s \setminus C = g(a), g(b), a, g(a)$$

(here, we denote trees by their corresponding term expressions, and we separate the trees in the sequence $s \setminus C$ with the symbol ","). Moreover, we have $C = \mathsf{up}(s, 2)$.

A tree $t \in \mathsf{Trees}_A$ is a *chain* if $D_t \neq \{\varepsilon\}$ and for every internal node $u \in D_t$, there is at most one child ui of u such that ui is internal. Hence, a chain t has a unique maximal (with respect to the prefix relation) internal node $\max(t) \in \mathbb{N}^*$. Note that a chain consists of at least two nodes.

Example 3.8. The tree t in Figure 3 is a chain with $\max(t) = 2111$.

Lemma 3.9. *Let A be a ranked alphabet and let $\mathsf{ranks} = \{m \in \mathbb{N} \mid m \geq 1, A_m \neq \emptyset\}$. Then, for all $n \geq 1$, the following are equivalent:*

(a) *There is a chain $t \in \mathsf{Trees}_A$ with exactly n leaves.*
(b) *There is a tree $t \in \mathsf{Trees}_A$ with exactly n leaves.*
(c) *There exist numbers $d_m \in \mathbb{N}$ (for each $m \in \mathsf{ranks}$) such that $n = 1 + \sum_{m \in \mathsf{ranks}} d_m \cdot (m-1)$.*

Proof. Implication $(a) \Rightarrow (b)$ is trivial. Now, assume (b) and let $t \in \mathsf{Trees}_A$ has exactly n leaves. We show (c) by induction on the size of t. We distinguish two cases. The case $n = 1$ is clear; set $d_m = 0$ for all $m \in \mathsf{ranks}$. Now, assume that t has $n \geq 2$ leaves. Then, there must exist an internal node $u \in D_t$ such that all children of u are leaves. Let $1 \leq a \leq n$ be the rank of the symbol $t(u)$. By replacing u by a leaf (labelled with an arbitrary constant from A_0), we get a strictly smaller tree with $n - (a-1)$ many leaves (note that $a = 1$ is possible). Since $a \leq n$ we have $n - (a-1) \geq 1$. By induction, there exist $d_m \in \mathbb{N}$ ($m \in \mathsf{ranks}$) such that $n - (a-1) = 1 + \sum_{m \in M} d_m \cdot (m-1)$. Thus, we have $n = 1 + (d_a + 1) \cdot (a-1) + \sum_{m \in \mathsf{ranks} \setminus \{a\}} d_m \cdot (m-1)$.

Finally, for the implication $(c) \Rightarrow (a)$, assume that $n = 1 + \sum_{m \in \mathsf{ranks}} d_m \cdot (m-1)$. Take a chain t that consists of $\sum_{m \in \mathsf{ranks}} d_m$ internal nodes, d_m of which are labelled with a symbol of rank m. All other nodes are leaves. It is a simple observation that t has exactly n leaves. $\square$

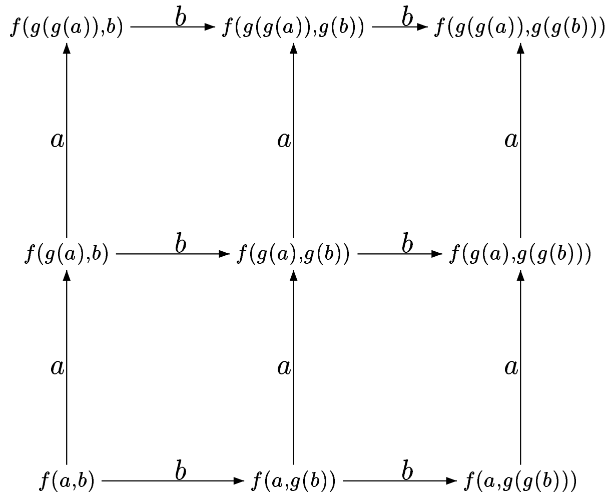


Figure 4: A finite part of the graph $\mathfrak{G}(\mathcal{R})$

The following lemma follows directly from Lemma 3.9.

Lemma 3.10. *Let A be a ranked alphabet and let $\mathsf{ranks} = \{m \in \mathbb{N} \mid m \geq 1, A_m \neq \emptyset\}$. Then, for every tree $t \in \mathsf{Trees}_A$ and every prefix-closed subset C of D_t the following holds, where n is the length of the string $t \setminus C$: There exist numbers $d_m \in \mathbb{N}$ (for each $m \in \mathsf{ranks}$) such that $n = 1 + \sum_{m \in \mathsf{ranks}} d_m \cdot (m-1)$.*

3.5. **Ground tree rewrite graphs.** A *ground tree rewrite system (GTRS)* is tuple $\mathcal{R} = (A, \Sigma, R)$, where A is a ranked alphabet, Σ is finite set of actions, and $\mathcal{R} \subseteq \mathsf{Trees}_A \times \Sigma \times \mathsf{Trees}_A$ is a finite set of rewrite rules. A rule (s, a, s') is also written as $s \overset{a}{\longmapsto} s'$. The *ground tree rewrite graph* defined by $\mathcal{R}$ is

$$\mathfrak{G}(\mathcal{R}) = (\mathsf{Trees}_A, \Sigma, \{\xrightarrow{a} \mid a \in \Sigma\}),$$

where for each $a \in \Sigma$, we have $t \xrightarrow{a} t'$ if and only if there exist a rule $(s \overset{a}{\longmapsto} s') \in R$ and $x \in D_t$ such that $t^{\downarrow x} = s$ and $t' = t[x/s']$.

Example 3.11. We define a GTRS $\mathcal{R} = (A, \Sigma, R)$ as follows. Let $A_0 = \{a, b\}$, $A_1 = \{g\}$, and $A_2 = \{f\}$, $\Sigma = \{a, b\}$, and let R consist of the following two rules:

$$a \overset{a}{\longmapsto} g(a), \qquad b \overset{b}{\longmapsto} g(b).$$

Take a tree $t(a_1, a_2, \ldots, a_n)$, where $a_1, \ldots, a_n \in \{a, b\}$, that does not contain a subtree of the form $g(a)$ or $g(b)$. Then, the (weakly) connected component of $\mathfrak{G}(\mathcal{R})$ that contains $t(a_1, a_2, \ldots, a_n)$ consists of all trees of the form $t(g^{i_1}(a_1), g^{i_2}(a_2), \ldots, g^{i_n}(a_n))$ for $i_1, \ldots, i_n \geq 0$. These trees form an n-dimensional grid, where edges in dimension $1 \leq j \leq k$ are labelled with a_j. Figure 4 shows the connected component of $\mathfrak{G}(\mathcal{R})$ that contains $f(a, b)$.

The next two lemmas are obvious:

Lemma 3.12. *Let $\mathcal{R} = (A, \Sigma, R)$ be a GTRS and let r be the maximal size of a tree that appears in R. Let s and t be ranked trees such that $d_{\mathfrak{G}(\mathcal{R})}(s, t) \leq n$. Then $\mathsf{size}(t) \leq \mathsf{size}(s) + r \cdot n$.*

Lemma 3.13. *Let $\mathcal{R} = (A, \Sigma, R)$ be a GTRS and let r be the maximal size of a tree that appears in R. Let s and t be ranked trees such that $\mathsf{diff}(s,t) > r \cdot n$. Then $d_{\mathfrak{G}(\mathcal{R})}(s,t) > n$.*

Recall the definition of the graph $\mathfrak{G}^+$ from (3.1).

Lemma 3.14. *Let $\mathcal{R} = (A, \Sigma, R)$ be a GTRS and let r be the maximal size of a tree that appears in R. Let t be a ranked tree, $n \in \mathbb{N}$, and let $C \subseteq \mathsf{up}(t, r \cdot n)$ be prefix-closed. Then we have*

$$S_n(\mathfrak{G}(\mathcal{R}), t) \cong S_n(\mathfrak{G}(\mathcal{R})^+, t \setminus C).$$

Proof. Let $t \setminus C = t_1 \cdots t_m$. Hence, there is a tree s with m leaves such that t results from s by replacing the i^{th} leaf of s by t_i ($1 \le i \le m$), let us write $t = s[t_1, t_2, \dots, t_m]$ for this. Recall that the subtree rooted in a node from $C \subseteq \mathsf{up}(t, r \cdot n)$ has size strictly larger than $r \cdot n$. Therefore, a node from C cannot be accessed by doing at most n rewrite steps. Hence, every tree $t' \in S_n(\mathfrak{G}(\mathcal{R}), t)$ can be written (uniquely) as $t' = s[t'_1, t'_2, \dots, t'_m]$. Moreover, the mapping $t' \mapsto t'_1 t'_2 \cdots t'_m$ defines an isomorphism from $S_n(\mathfrak{G}(\mathcal{R}), t)$ to $S_n(\mathfrak{G}(\mathcal{R})^+, t \setminus C)$. □

Remark 3.15. Note that if the word $w \in \mathsf{Trees}_A^+$ results from the string $t \setminus C$ by permuting the trees in the string, then we still have $S_n(\mathfrak{G}(\mathcal{R}), t) \cong S_n(\mathfrak{G}(\mathcal{R})^+, w)$.

The main goal of this paper is to study the complexity of the following set that we call the *uniform first-order theory of ground tree rewrite graphs*:

$$\{(\mathcal{R}, \varphi) \mid \mathcal{R} = (A, \Sigma, R) \text{ is a GTRS, } \varphi \text{ is an FO-sentence over the signature of } \mathfrak{G}(\mathcal{R}),\ \mathfrak{G}(\mathcal{R}) \models \varphi\}.$$

4. An $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$ upper bound

In this section we will prove the following result:

Theorem 4.1. *The uniform first-order theory of ground tree rewrite graphs belongs to the complexity class $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$.*

It suffices to prove Theorem 4.1 for the case that the underlying ranked alphabet A contains a symbol of rank at least two. A ground tree rewrite graph, where all symbols have rank at most 1 is in fact a suffix rewrite graph on words. Such a graph is first-order interpretable in a full $|\Gamma|$-ary tree Γ^* (with Γ finite), where the defining first-order formulas can be easily computed from the suffix rewrite system. Finally, the first-order theory of a full tree Γ^* (with $|\Gamma| \ge 2$) is complete for the class $\mathsf{ATIME}(2^{O(n)}, O(n))$ (under log-lin reductions) [11, 47].

The proof of Theorem 4.1 will be divided into two steps. In a first step, we will reduce the FO-theory for a given ground tree rewrite graph to the FO-theory for a very simple word rewrite graph of the form $\mathfrak{G}^+$, where $\mathfrak{G}$ is a finite labelled graph. Note that if V is the set of nodes of $\mathfrak{G}$, then V^+ is the set of nodes of $\mathfrak{G}^+$. Moreover, every edge in $\mathfrak{G}^+$ replaces a single symbol in a word by another symbol. In our reduction, the size of the set V will be doubly exponential in the input size (which is the size of the input formula plus the size of the input GTRS). In a second step, we will solve the FO-theory of a simple word structure $\mathfrak{G}^+$ on an alternating Turing machine. More precisely, we will show the following result:

Theorem 4.2. *There exists an alternating Turing-machine M, which accepts precisely those pairs $(\mathfrak{G}, \varphi)$, where $\mathfrak{G}$ is a finite labelled graph and φ is an FO-sentence over the signature of $\mathfrak{G}$ with $\mathfrak{G}^+ \models \varphi$. Moreover, M runs in time $O(n^{\ell+1} \cdot |\varphi|)$, where n is the number of nodes of $\mathfrak{G}$ and ℓ is the quantifier rank of φ. Finally, the number of alternations is bounded by $O(\ell)$.*

We prove Theorem 4.2 in Section 4.2. Together with our first reduction, Theorem 4.2 yields Theorem 4.1.

4.1. **Proof of Theorem 4.1.** In this section, we will prove Theorem 4.1. Let $\mathcal{R} = (A, \Sigma, R)$ be a GTRS over the ranked alphabet A and let r be the maximal size of a tree that appears in R. Let $\mathfrak{G} = \mathfrak{G}(\mathcal{R})$ and let φ be an FO-sentence of quantifier rank $\ell + 1$ over the signature of $\mathfrak{G}$. We want to check, whether $\mathfrak{G} \models \varphi$. Define the sets

$$\begin{aligned} \mathsf{ranks} &= \{m \in \mathbb{N} \mid m \geq 1, A_m \neq \emptyset\}, \\ M &= \{1 + \sum_{m \in \mathsf{ranks}} d_m \cdot (m-1) \mid d_m \in \mathbb{N} \text{ for } m \in \mathsf{ranks}\}. \end{aligned}$$

Note that by Lemma 3.9, we have $n \in M$ if and only if there exists a tree (or chain) $t \in \mathsf{Trees}_A$ with exactly n leaves. Also note that $M = \mathbb{N} \setminus \{0\}$ in case $A_2 \neq \emptyset$. Let

$$p = \max(\mathsf{ranks}) \geq 2$$

denote the maximal rank of a symbol from A. We define a function

$$\mathsf{int} : M \to \mathbb{N} \cup \{\infty\}$$

as follows: Let $m \in M$. If $A_1 \neq \emptyset$ (i.e., there exists a unary symbol), then we set $\mathsf{int}(m) = \infty$. If $A_1 = \emptyset$, then let $\mathsf{int}(m)$ be the maximal number of internal nodes in a tree $t \in \mathsf{Trees}_A$ with exactly m leaves (this maximum exists if $A_1 = \emptyset$; in fact $\mathsf{int}(m) \leq m - 1$). The intuition behind setting $\mathsf{int}(m) = \infty$ in case $A_1 \neq \emptyset$ is that there exist arbitrarily large trees with m leaves. Note that $\mathsf{int}(1) = 0$.

Lemma 4.3. *For every $m \in M$ we have $\mathsf{int}(m) \geq \frac{m-1}{p-1}$.*

Proof. It suffices to show the lemma for the case $A_1 = \emptyset$. In this case, the lemma can be shown by induction on m. The case $m = 1$ is clear. Let $m \in M \setminus \{1\}$ and let $t \in \mathsf{Trees}_A$ be a tree with m leaves and $\mathsf{int}(m)$ many internal nodes. Let $u \in D_t$ be an internal node such that all children of u are leaves. Let $t(u) \in A_q$ with $q \geq 2$. If we replace u by a leaf, we obtain a tree with $\mathsf{int}(m) - 1$ many internal nodes and $m - q + 1 \in M$ many leaves. We must have $\mathsf{int}(m - q + 1) = \mathsf{int}(m) - 1$ (if there would be a tree with $m - q + 1$ leaves and more than $\mathsf{int}(m) - 1$ many internal nodes, then we would obtain a tree with m leaves and more than $\mathsf{int}(m)$ many internal nodes by replacing an arbitrary leaf by a node with q children). Moreover, by induction (note that $q \geq 2$), we have $\mathsf{int}(m - q + 1) \geq \frac{m-q}{p-1}$. Hence, we get $\mathsf{int}(m) \geq \frac{m-q}{p-1} + 1 = \frac{m-q+p-1}{p-1} \geq \frac{m-1}{p-1}$ (since $p \geq q$). $\square$

Lemma 4.4. *Assume that $A_1 = \emptyset$. For every $m \in M$ there exists a chain with m leaves and $\mathsf{int}(m)$ many internal nodes.*

Proof. Let $m \in M$. By definition, there exists a tree $t \in \mathsf{Trees}_A$ with m leaves and $\mathsf{int}(m)$ internal nodes. It is easy to restructure t into a chain so that the number of leaves and the number of internal nodes is not changed. More precisely, take a tree $t = f(t_1, \dots, t_n)$ (in term notation) with m leaves and $\mathsf{int}(m)$ internal nodes, which is not a chain. By induction, we can assume that every t_i ($1 \le i \le n$) is either a chain or the constant $a \in A_0$ (for some arbitrarily chosen $a \in A_0$). Since t is not a chain there exist $1 \le i < j \le n$ such that t_i and t_j are chains. Choose an arbitrary child x of the the maximal internal node $\max(t_i)$ of the chain t_i; hence x is a leaf of t_i. Take the tree

$$t' = f(t_1, \dots, t_{i-1}, t_i[x/t_j], t_{i+1}, \dots, t_{j-1}, a, t_{j+1}, \dots, t_n).$$

This tree has the same number of leaves and internal nodes as t. Continuing this way, we finally obtain a chain. □

For numbers $1 \le i \le j$ let

$$T[i,j] = \{t \in \mathsf{Trees}_A \mid i \le \mathsf{size}(t) \le j\}.$$

For $0 \le i \le \ell$ let

$$\sigma(i) = \ell \cdot r \cdot 7 \cdot 4^i \cdot \big((p-1) \cdot r \cdot 4^i + 1\big) + p \cdot r \cdot 4^i \le r^2 \cdot p \cdot 2^{O(\ell)}. \tag{4.1}$$

Note that we have

$$\sigma(i+1) \ge \sigma(i) + p \cdot r \cdot 3 \cdot 4^i \ge \sigma(i) + r \cdot 3 \cdot 4^i \tag{4.2}$$

for all $0 \le i \le \ell$. Let

$$U = T[1, \sigma(\ell) + r \cdot p \cdot 4^\ell].$$

Moreover, for every $0 \le i \le \ell$ let

$$\begin{aligned} U_i &= T[1, \sigma(i)] \subseteq U, \\ V_i &= T[1, r \cdot 4^i] \subseteq U, && (4.3) \\ W_i &= \{\alpha(u_1, \dots, u_q) \mid q \ge 1, \alpha \in A_q, u_1, \dots, u_q \in V_i\} \setminus V_i \subseteq U. && (4.4) \end{aligned}$$

Note that $\mathsf{size}(t) \le r \cdot p \cdot 4^i + 1$ for all $t \in W_i$ and $V_i \cap W_i = \emptyset$. We consider the set U as a finite alphabet and the sets U_i, V_i, and W_i as subalphabets. Note that

$$|U| \le |A|^{\sigma(\ell) + r \cdot p \cdot 4^\ell}. \tag{4.5}$$

Define the language

$$Z = \{w \in U^+ \mid |w| \in M\} \tag{4.6}$$

over the alphabet U. Note that $Z = U^+$ in case $A_2 \neq \emptyset$. On the set $(\mathbb{N} \times Z) \cup U$ we define a labelled graph $\mathfrak{G}_1$ with label set Σ as follows: Take an action $\sigma \in \Sigma$. By our general lifting constructions from Section 3.2, the binary relation $\xrightarrow{\sigma}$ on Trees_A is implicitly lifted to a binary relation on Trees_A^+ and $\mathbb{N} \times \mathsf{Trees}_A^+$. Since $(\mathbb{N} \times Z) \cap U = \emptyset$, $\xrightarrow{\sigma}$ can be viewed as a binary relation on $(\mathbb{N} \times Z) \cup U$; simply take the disjoint union of the relations on $(\mathbb{N} \times Z)$ and U. Finally, we define the Σ-labelled graph

$$\mathfrak{G}_1 = ((\mathbb{N} \times Z) \cup U,\ \Sigma,\ \{\xrightarrow{\sigma} \mid \sigma \in \Sigma\}). \tag{4.7}$$

For a word $w = u_1 u_2 \cdots u_n \in U^*$ with $u_1, \ldots, u_n \in U$ we define

$$\|w\| = \sum_{i=1}^{n} \mathsf{size}(u_i).$$

We define the sets

$$\begin{aligned} Z_i &= \{w \in V_i^* W_i V_i^* \cap Z \mid \|w\| + \mathsf{int}(|w|) > \sigma(i)\}, \\ L_i &= (\mathbb{N} \times Z_i) \cup U_i. \end{aligned} \tag{4.8}$$

Note that $Z_i = V_i^* W_i V_i^* \cap Z$ in case $A_1 \neq \emptyset$ (clearly, we set $n + \infty = \infty$ for every number n). Assume that the first-order sentence φ is of the form $Q_\ell x_\ell \, \cdots \, Q_1 x_1 \; Q_0 x_0 : \psi$, where $Q_0, \ldots, Q_\ell \in \{\forall, \exists\}$ and ψ is quantifier-free. For $0 \leq i \leq \ell - 1$ and elements $s_{i+1}, \ldots, s_\ell \in (\mathbb{N} \times Z) \cup U$ let us define the set

$$L_i(s_{i+1}, \ldots, s_\ell) = L_i \cup S_{3 \cdot 4^i}(\mathfrak{S}_1, s_{i+1}, \ldots, s_\ell).$$

We define a first-order sentence φ_1 (with quantifiers relativized to the sets $L_i(s_{i+1}, \ldots, s_\ell)$) over the signature of $\mathfrak{S}_1$ as

$$\varphi_1 \;=\; Q_\ell x_\ell \in L_\ell \; Q_{\ell-1} x_{\ell-1} \in L_{\ell-1}(x_\ell) \; \cdots \; Q_0 x_0 \in L_0(x_1, \ldots, x_\ell) : \psi. \tag{4.9}$$

We want to show that $\mathfrak{G} \models \varphi$ if and only if $\mathfrak{S}_1 \models \varphi_1$. For this, we need the following lemma, which is the main technical contribution in this section. The reader might skip the proof at first reading.

Lemma 4.5. *Assume that*

- $0 \leq i \leq \ell$,
- $\overline{s} = (s_{i+1}, \ldots, s_\ell) \in ((\mathbb{N} \times Z) \cup U)^{\ell - i}$ *with* $s_j \in L_j \cup S_{3 \cdot 4^j}(\mathfrak{S}_1, s_{j+1}, \ldots, s_\ell)$ *for all* $j \in [i+1, \ell]$,
- $\overline{t} = (t_{i+1}, \ldots, t_\ell) \in \mathsf{Trees}_A^{\ell - i}$, *and*
- $f : S_{4^{i+1}}(\mathfrak{S}_1, \overline{s}) \to S_{4^{i+1}}(\mathfrak{G}, \overline{t})$ *is an isomorphism such that* $f \restriction S_{4^{i+1}}(\mathfrak{S}_1, s_j)$ *is the identity for all* $j \in [i+1, \ell]$ *with* $t_j \in U_{i+1}$ *or* $s_j \in U_{i+1}$.

Then, the following holds:

(a) *For all* $t_i \in \mathsf{Trees}_A$ *there exists* $s_i \in L_i \cup S_{3 \cdot 4^i}(\mathfrak{S}_1, \overline{s})$ *and an isomorphism*

$$g : S_{4^i}(\mathfrak{S}_1, s_i, \overline{s}) \to S_{4^i}(\mathfrak{G}, t_i, \overline{t})$$

such that f *and* g *are compatible*[3] *and* $g \restriction S_{4^i}(\mathfrak{S}_1, s_j)$ *is the identity for all* $j \in [i, \ell]$ *with* $t_j \in U_i$ *or* $s_j \in U_i$.

(b) *For all* $s_i \in L_i \cup S_{3 \cdot 4^i}(\mathfrak{S}_1, \overline{s})$ *there exists* $t_i \in \mathsf{Trees}_A$ *and an isomorphism*

$$g : S_{4^i}(\mathfrak{S}_1, s_i, \overline{s}) \to S_{4^i}(\mathfrak{G}, t_i, \overline{t})$$

such that f *and* g *are compatible and* $g \restriction S_{4^i}(\mathfrak{S}_1, s_j)$ *is the identity for all* $j \in [i, \ell]$ *with* $t_j \in U_i$ *or* $s_j \in U_i$.

Before we prove the lemma, let us provide some intuition. For case (a) we will basically distinguish two cases: In case t_i is "close" to some tree in the tuple $\overline{t}$, then the simulating s_i can safely be chosen as t_i itself. In case t_i is "far" to all trees in $\overline{t}$, we distinguish two cases: Either the size of t_i exceeds $\sigma(i)$ from (4.1) or not. If $|t_i| > \sigma(i)$, then s_i will be chosen as a pair from $\{n\} \times Z_i$ for some fresh number n that does not appear as a first component of any element in $\overline{s}$, and where the second component of s_i consists basically

[3]Recall the definition of compatible functions from the beginning of Section 3.

of $t_i \setminus C$ for some prefix-closed subset C of t_i's nodes. Intuitively, this means that s_i does not have to be "too big" in order to simulate t_i: only "small" subtrees of t_i have to be accounted for. Lemma 3.14 will be crucial. In case $|t_i| \le \sigma(i)$, we can prove that we can set $s_i = t_i \in U_i$. For case (b) we can proceed similarly, but the main crux is that for each element $s_i \in \mathbb{N} \times Z_i$ we can build a tree $t_i \in \mathsf{Trees}_A$ such that the spheres of radius 4^i around s_i and t_i are isomorphic. For building the latter trees, we have to distinguish the case when $A_1 \neq \emptyset$ and the case when $A_1 = \emptyset$.

Proof. Let $f : S_{4^{i+1}}(\mathfrak{S}_1, \overline{s}) \to S_{4^{i+1}}(\mathfrak{G}, \overline{t})$ be an isomorphism such that $f \restriction S_{4^{i+1}}(\mathfrak{S}_1, s_j)$ is the identity for all $i+1 \le j \le \ell$ with $t_j \in U_{i+1}$ or $s_j \in U_{i+1}$. Let us first prove statement (a). Let $t_i \in \mathsf{Trees}_A$. We distinguish two cases:

Case 1. $t_i \in S_{3 \cdot 4^i}(\mathfrak{G}, \overline{t})$. Note that this implies that t_i belongs to the range of the isomorphism f and that

$$S_{4^i}(\mathfrak{G}, t_i, \overline{t}) \subseteq S_{4^{i+1}}(\mathfrak{G}, \overline{t}).$$

Then, we set $s_i = f^{-1}(t_i) \in S_{3 \cdot 4^i}(\mathfrak{S}_1, \overline{s})$. We define g as the restriction of f to the set $S_{4^i}(\mathfrak{S}_1, s_i, \overline{s}) \subseteq S_{4^{i+1}}(\mathfrak{S}_1, \overline{s})$. Now, assume that $t_i \in U_i$, i.e., $\mathsf{size}(t_i) \le \sigma(i)$. We have to show that $f \restriction S_{4^i}(\mathfrak{S}_1, s_i)$ is the identity. Let t_j $(i+1 \le j \le \ell)$ such that $d_{\mathfrak{G}}(t_i, t_j) \le 3 \cdot 4^i$. With Lemma 3.12 it follows

$$\mathsf{size}(t_j) \le \mathsf{size}(t_i) + r \cdot 3 \cdot 4^i \le \sigma(i) + r \cdot 3 \cdot 4^i \overset{(4.2)}{\le} \sigma(i+1).$$

Hence, $t_j \in U_{i+1}$ and $f \restriction S_{4^{i+1}}(\mathfrak{S}_1, s_j)$ is the identity. Since $d_{\mathfrak{S}_1}(s_i, s_j) = d_{\mathfrak{G}}(t_i, t_j) \le 3 \cdot 4^i$, we have $S_{4^i}(\mathfrak{S}_1, s_i) \subseteq S_{4^{i+1}}(\mathfrak{S}_1, s_j)$. It follows that $f \restriction S_{4^i}(\mathfrak{S}_1, s_i)$ is the identity. If $s_i \in U_i$, then we can argue analogously.

Case 2. $t_i \notin S_{3 \cdot 4^i}(\mathfrak{G}, \overline{t})$. We will find $s_i \in L_i$ and an isomorphism $f' : S_{4^i}(\mathfrak{S}_1, s_i) \to S_{4^i}(\mathfrak{G}, t_i)$ such that $s_i \notin S_{3 \cdot 4^i}(\mathfrak{S}_1, \overline{s})$. Then, Lemma 3.1 implies that $g = (f \restriction S_{4^i}(\mathfrak{S}_1, \overline{s})) \uplus f'$ is an isomorphism from $S_{4^i}(\mathfrak{S}_1, s_i, \overline{s})$ to $S_{4^i}(\mathfrak{G}, t_i, \overline{t})$, which is compatible with f. Moreover, we will show that if $t_i \in U_i$ or $s_i \in U_i$, then f' is the identity.

In order to find s_i, let $t_i \setminus \mathsf{up}(t_i, r \cdot 4^i) = u_1 \cdots u_m$. Recall that the latter string is the lexicographic order of all maximal subtrees of t_i whose size is at most $r \cdot 4^i$. Hence, $\mathsf{size}(u_j) \le r \cdot 4^i$ for each j, i.e., $u_j \in V_i$ (see (4.3)).

Case 2.1. $\mathsf{size}(t_i) > \sigma(i)$. We must have $t_i \neq u_1$, because otherwise $\mathsf{size}(t_i) \le r \cdot 4^i \le \sigma(i)$, which is a contradiction. Therefore, there must exist $1 \le j \le m$, a symbol $\alpha \in A$ of rank $q \ge 1$, and a prefix-closed subset $C \subseteq \mathsf{up}(t_i, r \cdot 4^i)$ such that $\alpha(u_j, \ldots, u_{j+q-1}) \in W_i$ (see (4.4)) and

$$t_i \setminus C = u_1 \cdots u_{j-1} \alpha(u_j, \ldots, u_{j+q-1}) u_{j+q} \cdots u_m.$$

Let $w = t_i \setminus C$. By Lemma 3.10, we have $|w| \in M$. By the definition of the mapping int, we have $\|w\| + \mathsf{int}(|w|) \ge \mathsf{size}(t_i)$ and hence $\|w\| + \mathsf{int}(|w|) > \sigma(i)$ by assumption. Thus, we get $w \in Z_i$ by definition of Z_i in (4.8). Choose a number $n \in \mathbb{N}$ such that n does not appear as a first component of a pair from $\{s_{i+1}, \ldots, s_\ell\} \cap (\mathbb{N} \times Z)$. Finally, we set

$$s_i = (n, w) \in \mathbb{N} \times Z_i \subseteq L_i.$$

Due to the choice of n, we have $s_i \notin S_\rho(\mathfrak{S}_1, \overline{s})$ for all ρ. Moreover, with Lemma 3.14 we get $S_{4^i}(\mathfrak{S}_1, s_i) \cong S_{4^i}(\mathfrak{G}, t_i)$. Finally, $\mathsf{size}(t_i) > \sigma(i)$, i.e., $t_i \notin U_i$, and $s_i \notin U$.

Case 2.2. $\mathsf{size}(t_i) \le \sigma(i)$, i.e., $t_i \in U_i$. We set $s_i = t_i \in U_i$. Note that $S_{4^i}(\mathfrak{G}, t_i) \subseteq U$, which implies $S_{4^i}(\mathfrak{S}_1, s_i) = S_{4^i}(\mathfrak{G}, t_i)$. Assume that $s_i \in S_{3 \cdot 4^i}(\mathfrak{S}_1, \overline{s})$. We will deduce a contradiction. Let $i+1 \le j \le \ell$ such that $d_{\mathfrak{S}_1}(s_i, s_j) \le 3 \cdot 4^i$. Since $s_i \in U$, we must have

$s_j \in U$ as well (there is no path in $\mathfrak{S}_1$ between the sets U and $\mathbb{N} \times Z$). Moreover, with Lemma 3.12 we get

$$\mathsf{size}(s_j) \leq \mathsf{size}(s_i) + r \cdot 3 \cdot 4^i \leq \sigma(i) + r \cdot 3 \cdot 4^i \overset{(4.2)}{\leq} \sigma(i+1),$$

i.e., $s_j \in U_{i+1}$. This implies that $f \restriction S_{4^{i+1}}(\mathfrak{S}_1, s_j)$ is the identity. Hence, $t_i \in S_{3.4^i}(\mathfrak{G}, t_j)$, a contradiction. We can finally choose for f' the identity isomorphism on $S_{4^i}(\mathfrak{S}_1, s_i) = S_{4^i}(\mathfrak{G}, t_i)$. This proves (a).

Let us now prove (b). Let $s_i \in L_i \cup S_{3.4^i}(\mathfrak{S}_1, \overline{s})$. Again, we distinguish two cases.

Case 1. $s_i \in S_{3.4^i}(\mathfrak{S}_1, \overline{s})$. This implies

$$S_{4^i}(\mathfrak{S}_1, s_i, \overline{s}) \subseteq S_{4^{i+1}}(\mathfrak{S}_1, \overline{s}).$$

We set $t_i = f(s_i) \in S_{3.4^i}(\mathfrak{G}, \overline{t})$. We can conclude as in Case 1 for the proof of point (a) above.

Case 2. $s_i \notin S_{3.4^i}(\mathfrak{S}_1, \overline{s})$. Hence, $s_i \in L_i$. We will find $t_i \in \mathsf{Trees}_A$ and an isomorphism $f' : S_{4^i}(\mathfrak{S}_1, s_i) \to S_{4^i}(\mathfrak{G}, t_i)$ such that $t_i \notin S_{3.4^i}(\mathfrak{G}, \overline{t})$. Then, Lemma 3.1 implies that the mapping $g = (f \restriction S_{4^i}(\mathfrak{S}_1, \overline{s})) \uplus f'$ is an isomorphism from $S_{4^i}(\mathfrak{S}_1, s_i, \overline{s})$ to $S_{4^i}(\mathfrak{G}, t_i, \overline{t})$, which is compatible with f. Moreover, we will show that if $t_i \in U_i$ or $s_i \in U_i$, then f' is the identity.

Case 2.1. $s_i \in U_i \subseteq \mathsf{Trees}_A$. We set $t_i = s_i \in U_i$, which implies $S_{4^i}(\mathfrak{G}, t_i) \subseteq U$. Thus, $S_{4^i}(\mathfrak{S}_1, s_i) = S_{4^i}(\mathfrak{G}, t_i)$. Assume that $t_i \in S_{3.4^i}(\mathfrak{G}, \overline{t})$. We will deduce a contradiction. Let $i + 1 \leq j \leq \ell$ such that $d_{\mathfrak{G}}(t_i, t_j) \leq 3 \cdot 4^i$. Lemma 3.12 implies

$$\mathsf{size}(t_j) \leq \mathsf{size}(t_i) + r \cdot 3 \cdot 4^i \leq \sigma(i) + r \cdot 3 \cdot 4^i \overset{(4.2)}{\leq} \sigma(i+1).$$

This implies that $f \restriction S_{4^{i+1}}(\mathfrak{S}_1, s_j)$ is the identity. Hence, $s_i \in S_{3.4^i}(\mathfrak{S}_1, s_j)$, a contradiction. We can finally choose for f' the identity isomorphism on $S_{4^i}(\mathfrak{S}_1, s_i) = S_{4^i}(\mathfrak{G}, t_i)$.

Case 2.2. $s_i \in \mathbb{N} \times Z_i$. Let $s_i = (n, u_1 \cdots u_m)$ with $u_1, \ldots, u_m \in V_i \cup W_i$, $m \in M$, and $\|u_1 \cdots u_m\| + \mathsf{int}(m) > \sigma(i)$. There is exactly one $1 \leq j \leq m$ with $u_j \in W_i$. Let $u_j = \alpha(v_1, \ldots, v_q)$ with $q \geq 1$, $\alpha \in A_q$, and $v_1, \ldots, v_q \in V_i$. Define the string

$$w = u_1 \cdots u_{j-1} v_1 \cdots v_q u_{j+1} \cdots u_m \tag{4.10}$$

of length $m + q - 1$. Since $m \in M$, we also have $m + q - 1 \in M$.

Case 2.2.1. $A_1 \neq \emptyset$. Then, we can choose for t_i a tree with the following properties:

- $t_i \setminus \mathsf{up}(t_i, r \cdot 4^i) = w$. For this, we connect all trees $u_1, \ldots, u_m$ to one tree using a chain of symbols of rank at least 2, starting from $u_j \in W_i$. Since $m \in M$, this is possible by Lemma 4.4 (applied to the ranked alphabet $A \setminus A_1$).
- $t_i \notin S_{3.4^i}(\mathfrak{G}, \overline{t})$ and $\mathsf{size}(t_i) > \sigma(i)$. This can be enforced by adding a long enough chain of unary symbols to the root.

With Lemma 3.14, the first point implies $S_{4^i}(\mathfrak{S}_1, s_i) \cong S_{4^i}(\mathfrak{G}, t_i)$. Moreover, since $\mathsf{size}(t_i) > \sigma(i)$, we have $t_i \notin U_i$.

Case 2.2.2. $A_1 = \emptyset$ and thus $\mathsf{int}(m) < \infty$. Note that $\|w\| = \|u_1 \cdots u_m\| - 1$, i.e.,

$$\|w\| + \mathsf{int}(m) = \|u_1 \cdots u_m\| + \mathsf{int}(m) - 1 \geq \sigma(i).$$

Every tree in the string w has size at most $r \cdot 4^i$. Hence, we have $\|w\| \leq (m + q - 1) \cdot r \cdot 4^i$. We get

$$(m + q - 1) \cdot r \cdot 4^i + \mathsf{int}(m) \geq \sigma(i).$$

Moreover, since $\mathsf{int}(m) \geq \frac{m-1}{p-1}$ by Lemma 4.3, we have $m+q-1 \leq \mathsf{int}(m)\cdot(p-1)+q \leq \mathsf{int}(m)\cdot(p-1)+p$. We get

$$(\mathsf{int}(m)\cdot(p-1)+p)\cdot r\cdot 4^i + \mathsf{int}(m) \geq \sigma(i).$$

Solving this inequality for $\mathsf{int}(m)$ yields

$$\mathsf{int}(m) \geq \frac{\sigma(i) - p\cdot r\cdot 4^i}{(p-1)\cdot r\cdot 4^i + 1}.$$

Plugging in the definition of $\sigma(i)$ from (4.1) yields

$$\mathsf{int}(m) \geq \ell\cdot r\cdot 7\cdot 4^i. \tag{4.11}$$

We now define $\ell+1$ different trees $t'_1,\ldots,t'_{\ell+1}$ as follows.

We first fix a sequence $\alpha_1,\ldots,\alpha_{\mathsf{int}(m)}$ of symbols from $A\setminus A_0$ such that that every chain, where the j^{th} internal node is labelled with α_j has exactly m leaves. By Lemma 4.4 such a sequence exists. In the following, we consider chains with $\mathsf{int}(m)+1$ many internal nodes such that the following hold:

- The j^{th} internal node $(1\leq j\leq \mathsf{int}(m))$ is labelled with α_j and the maximal internal node is labelled with $\alpha\in A_q$ (thus, such a chain has $m+q-1$ leaves).
- Every internal node belongs to $\{1,2\}^*$ (thus, every internal node, which is not the root, is either the first or the second child of its parent node).
- All leaves in the chain are labelled with some fixed constant $\square\in A_0$.

This means that such a chain is uniquely determined by its maximal internal node $u = \max(t)\in\{1,2\}^{\mathsf{int}(m)}$. We write $t = \mathsf{chain}(u)$.

Let $u,v\in\{1,2\}^{\mathsf{int}(m)}$ such that $u = xay$ and $v = xbz$ with $x,y,z\in\{1,2\}^*$, $a,b\in\{1,2\}$, $a\neq b$. Define $\mathsf{diff}(u,v) = |y|+1$ $(=|z|+1)$. Recall also the definition of the diff-value for two trees from Section 3.4. Then, we have

$$\mathsf{diff}(\mathsf{chain}(u),\mathsf{chain}(v)) > \mathsf{diff}(u,v). \tag{4.12}$$

In fact, $\mathsf{diff}(\mathsf{chain}(u),\mathsf{chain}(v)) \geq 2\cdot\mathsf{diff}(u,v)$ holds.

Since $\mathsf{int}(m)\geq \ell\cdot r\cdot 7\cdot 4^i$ by (4.11), we can find $\ell+1$ strings $w_1,\ldots,w_{\ell+1}\in\{1,2\}^{\mathsf{int}(m)}$ such that for all $k\neq k'$ we have

$$\mathsf{diff}(w_k,w_{k'}) \geq r\cdot 7\cdot 4^i. \tag{4.13}$$

We may for instance set

$$w_k = 1^{\mathsf{int}(m)-\ell\cdot r\cdot 7\cdot 4^i}1^{(k-1)\cdot r\cdot 7\cdot 4^i}2^{(\ell-k+1)\cdot r\cdot 7\cdot 4^i}.$$

Let us define the chain $c_k = \mathsf{chain}(w_k)$ for all $1\leq k\leq \ell+1$. Hence, (4.12) and (4.13) imply

$$\mathsf{diff}(c_k,c_{k'}) > r\cdot 7\cdot 4^i \tag{4.14}$$

for all $k\neq k'$. Moreover, every chain c_k has exactly $m+q-1$ leaves. Finally, the tree t'_k is obtained from the chain c_k as follows: We replace the q children of the maximal internal node $\max(c_k)$ (which is labelled with $\alpha\in A_q$) by $v_1,\ldots,v_q$ (in this order). All other $m-1$ leaves are replaced by the trees $u_1,\ldots,u_{j-1},u_{j+1},\ldots,u_m$ (the order does not matter). It follows that the string $t'_k\setminus\mathsf{up}(t'_k,r\cdot 4^i)$ is a permutation of the string w from (4.10). With Lemma 3.14 and Remark 3.15 this ensures that

$$S_{4^i}(\mathfrak{S}_1,s_i)\cong S_{4^i}(\mathfrak{G},t'_k)$$

for all $1 \leq k \leq \ell+1$. Moreover, since each of the trees $u_1, \ldots, u_{j-1}, v_1, \ldots, v_q, u_{j+1}, \ldots, u_m \in V_i$ has size at most $r \cdot 4^i$, the number of nodes in the subtree of t'_k rooted at a leaf of c_k may grow by at most $r \cdot 4^i$, when we replace the leaf by one of the trees $u_1, \ldots, u_{j-1}, v_1, \ldots, v_q, u_{j+1}, \ldots, u_m$. This implies

$$\mathsf{diff}(t'_k, t'_{k'}) \geq \mathsf{diff}(c_k, c_{k'}) - r \cdot 4^i \stackrel{(4.14)}{>} r \cdot 6 \cdot 4^i,$$

provided $k \neq k'$. Hence, Lemma 3.13 implies

$$d_{\mathfrak{G}}(t'_k, t'_{k'}) > 6 \cdot 4^i \tag{4.15}$$

for all $k \neq k'$. We claim that there is at least one $1 \leq k \leq \ell + 1$ such that $t'_k \notin S_{3 \cdot 4^i}(\mathfrak{G}, \overline{t})$. In order to obtain a contradiction, assume that for each $1 \leq k \leq \ell + 1$ there exists some t_h $(i + 1 \leq h \leq \ell)$ such that $d_{\mathfrak{G}}(t'_k, t_h) \leq 3 \cdot 4^i$. Since there are only $\ell - i \leq \ell$ such trees t_h, the pigeon hole principle implies that there exist $k \neq k'$ and h with $d_{\mathfrak{G}}(t'_k, t_h) \leq 3 \cdot 4^i$ and $d_{\mathfrak{G}}(t'_{k'}, t_h) \leq 3 \cdot 4^i$. Hence, $d_{\mathfrak{G}}(t'_k, t'_{k'}) \leq 6 \cdot 4^i$, which contradicts (4.15). We finally set $t_i = t'_k$, where k is chosen such that $t'_k \notin S_{3 \cdot 4^i}(\mathfrak{G}, \overline{t})$. Finally, note that $\mathsf{size}(t_i) = \|u_1 \cdots u_m\| + \mathsf{int}(m) > \sigma(i)$ (i.e., $t_i \notin U_i$) and $s_i \notin U$. This concludes the proof of the lemma. □

Lemma 4.5 allows us to prove the following lemma:

Lemma 4.6. *Assume that*

- $-1 \leq i \leq \ell$,
- $\overline{s} = (s_{i+1}, \ldots, s_\ell) \in ((\mathbb{N} \times Z) \cup U)^{\ell - i}$ *with* $s_j \in L_j \cup S_{3 \cdot 4^j}(s_{j+1}, \ldots, s_\ell)$ *for all* $j \in [i+1, \ell]$,
- $\overline{t} = (t_{i+1}, \ldots, t_\ell) \in \mathsf{Trees}_A^{\ell-i}$, *and*
- $f : S_{4^{i+1}}(\mathfrak{S}_1, \overline{s}) \to S_{4^{i+1}}(\mathfrak{G}, \overline{t})$ *is an isomorphism such that* $f \restriction S_{4^{i+1}}(\mathfrak{S}_1, s_j)$ *is the identity for all* $j \in [i+1, \ell]$ *with* $t_j \in U_{i+1}$ *or* $s_j \in U_{i+1}$.

Then, for every quantifier-free first-order formula ψ *over the signature of* $\mathfrak{G}$ *and all quantifiers* $Q_0, \ldots, Q_i \in \{\forall, \exists\}$ *we have*

$$\mathfrak{S}_1 \models Q_i x_i \in L_i(\overline{s}) \cdots Q_0 x_0 \in L_0(x_1, \ldots, x_i, \overline{s}) : \psi(x_0, \ldots, x_i, \overline{s})$$

$$\iff$$

$$\mathfrak{G} \models Q_i x_i \cdots Q_0 x_0 : \psi(x_0, \ldots, x_i, \overline{t}).$$

Proof. The lemma can be shown by induction on i, starting with $i = -1$. For the induction base $(i = -1)$, note that the existence of the isomorphism f ensures that $\overline{s}$ and $\overline{t}$ satisfy the same quantifier-free formulas. The induction step uses Lemma 4.5 and the classical back-and-forth argument from the proof of the Ehrenfeucht-Fraïssé-Theorem. □

Setting $i = \ell$ in Lemma 4.6, it follows $\mathfrak{G} \models \varphi$ if and only if $\mathfrak{S}_1 \models \varphi_1$, where φ_1 is from (4.9).

For the remainder of the poof of Theorem 4.1, we proceed as follows: We simplify the sentence φ_1 (which is not an ordinary first-order sentence due to the additional constraints for the variables $x_0, \ldots, x_\ell$) and the structure $\mathfrak{S}_1$ further, so that we can finally apply Theorem 4.2. In a first step (Step 1 below), we eliminate in the formula φ_1 the relativation of the variables x_i to the spheres $S_{3 \cdot 4^i}(\mathfrak{S}_1, x_{i+1}, \ldots, x_\ell) \subseteq L_i(x_{i+1}, \ldots, x_\ell)$. Then, the structure $\mathfrak{S}_1$ will be freplaced by an isomorphic structure $\mathfrak{S}_3$ (using an intermediate isomorphic copy $\mathfrak{S}_2$). These is done in Step 2 and Step 3 below. The structure $\mathfrak{S}_3$ will be almost of the form $\mathfrak{T}^+$ for a finite labelled graph $\mathfrak{T}$ (these are the structures appearing in Theorem 4.2).

The only difference is that the universe of $\mathfrak{S}_3$ is a regular language of the form $\Delta^*\Theta\Delta^*$ (for finite alphabets Δ and Θ) instead of the set of all non-empty finite words (as it is the case for $\mathfrak{T}^+$). Also the constraint sets $L_i \subseteq L_i(x_{i+1},\ldots,x_\ell)$ from $\mathfrak{S}_1$ will be mapped to simple regular languages in $\mathfrak{S}_3$. We finally transform $\mathfrak{S}_3$ into a structure $\mathfrak{S}_4 = \mathfrak{T}^+$ by enlarging the finite alphabet over which words from $\mathfrak{S}_3$ are defined (Step 4).

Step 1. Recall that quantifiers in φ_1 are relativized to the sets

$$L_i(x_{i+1},\ldots,x_\ell) = L_i \cup S_{3\cdot 4^i}(\mathfrak{S}_1, x_{i+1},\ldots,x_\ell).$$

Note that $x_i \in S_{3\cdot 4^i}(\mathfrak{S}_1, x_{i+1},\ldots,x_\ell)$ means that $\bigvee_{j=i+1}^{\ell} d_{\mathfrak{S}_1}(x_i,x_j) \leq 3\cdot 4^i$ holds. By Lemma 3.3 we can find an equivalent first-order formula of size $O((\ell-i)\cdot i + (\ell-i)\cdot|\Sigma|) \leq O(\ell^2+\ell\cdot|\Sigma|)$ and quantifier rank $O(i) \leq O(\ell)$ (we take the formula $\theta(x,y) = (x = y \vee \bigvee_{\sigma\in\Sigma}\sigma(x,y)\vee\sigma(y,x))$ in Lemma 3.3; note that the binary representation of $3\cdot 4^i$ has only 2 1-bits). After replacing the constraints $x_i \in S_{3\cdot 4^i}(\mathfrak{S}_1, x_{i+1},\ldots,x_\ell)$ for $1\leq i\leq \ell$, the resulting equivalent sentence has size $|\varphi| + O(\ell^3+\ell^2\cdot|\Sigma|)$ and quantifier rank $O(\ell)$.

Step 2. It remains to eliminate constraints of the form $x_i \in L_i = (\mathbb{N}\times Z_i)\cup U_i$. In order to do this, we will change the labelled graph $\mathfrak{S}_1$ to a labelled graph of the form $\mathfrak{T}^+$ for a finite labelled graph $\mathfrak{T}$. The basic idea will be to change the alphbet U by taking words over of U of some bounded length as the new symbols; the resulting alphabet will be the set $U'\cup U''$ below.

In the following, we assume that $A_1 = \emptyset$ (and hence $\mathsf{int}(m) < \infty$ for all m); the case $A_1 \neq \emptyset$ is the simpler one.

In order to cope with the length constraint $|w| \in M$ in the definition of the set Z_i from (4.8), we define for $0\leq i\leq \ell$ the sets

$$\begin{aligned} U' &= \{w \in U^+ \mid |w|+1 \in \mathsf{ranks}\},\\ V_i' &= \{w \in V_i^+ \mid |w|+1 \in \mathsf{ranks}\} \subseteq U'. \end{aligned}$$

We have

$$|U'| \leq (|U|+1)^{p-1} \overset{(4.5)}{\leq} (|A|+1)^{(p-1)\cdot(\sigma(\ell)+p\cdot r\cdot 4^\ell)} \overset{(4.1)}{\leq} |A|^{2^{O(\ell)}p^2\cdot r^2}. \tag{4.16}$$

Moreover, for $0\leq i\leq\ell$ let us define W_i' as the set of all minimal words (with respect to the factor relation on words) $w \in V_i^*W_iV_i^*$ with $|w|\in M$ (and hence $w\in Z$ by (4.6)) and $\|w\|+\mathsf{int}(|w|) > \sigma(i)$ (and hence $w \in Z_i$ by (4.8)). It follows that for such a word w we have

$$\|w\| + \mathsf{int}(|w|) - (p-1)\cdot r\cdot 4^i - 1 \leq \sigma(i).$$

Since $|w| \leq \|w\|$ and $\mathsf{int}(|w|) \geq \frac{|w|-1}{p-1}$ by Lemma 4.3, we have

$$|w| + \frac{|w|-1}{p-1} - (p-1)\cdot r\cdot 4^i - 1 \leq \sigma(i)$$

or equivalently

$$|w| \leq \frac{p-1}{p}\cdot\sigma(i) + \frac{(p-1)^2}{p}\cdot r\cdot 4^i + 1.$$

Hence, for all $w\in W_i'$ we have

$$|w| \leq \sigma(i) + p\cdot r\cdot 4^i + 1. \tag{4.17}$$

Let us set

$$\gamma = \sigma(\ell) + p\cdot r\cdot 4^\ell + 1 \overset{(4.1)}{\leq} p\cdot r^2\cdot 2^{O(\ell)} + p\cdot r\cdot 4^\ell + 1 = p\cdot r^2\cdot 2^{O(\ell)}, \tag{4.18}$$

which is an upper bound for the right-hand side of (4.17). Note that γ is exponential in our input size. Let

$$U'' = \{w \in Z \mid |w| \leq \gamma\},$$

which contains all alphabets W'_i ($0 \leq i \leq \ell$) as well as U. We have

$$|U''| \leq (|U|+1)^{\gamma} \overset{(4.5)}{\leq} (|A|+1)^{\gamma\cdot(\sigma(\ell)+p\cdot r\cdot 4^{\ell})} \overset{(4.1)}{\leq} |A|^{2^{O(\ell)}p^2\cdot r^4} \tag{4.19}$$

which is doubly exponential in our input size.

For the further discussion, it is important that elements of $U' \cup U''$ are viewed as single symbols. For a word $w \in (U' \cup U'')^*$ we can define an expanded word $\mathsf{exp}(w) \in U^*$ in the natural way (e.g. $\mathsf{exp}((a)(abba)(b)(ba)) = aabbabba$). Note that for every word $w \in U'^*U''U'^*$ we have $|\mathsf{exp}(w)| \in M$ (i.e., $\mathsf{exp}(w) \in Z$). Vice versa, for every word $w \in Z$ there exists at least one word $w' \in U'^*U''U'^*$ with $\mathsf{exp}(w') = w$. Moreover, for every word $w \in V_i'^*W_i'V_i'^*$ we have $|\mathsf{exp}(w)| \in M$ and $\|\mathsf{exp}(w)\| + \mathsf{int}(|\mathsf{exp}(w)|) > \sigma(i)$. Vice versa, if $w \in V_i^*W_iV_i^* \cap Z$ with $\|w\| + \mathsf{int}(|w|) > \sigma(i)$ (i.e., $w \in Z_i$), then there exists at least one word $w' \in V_i'^*W_i'V_i'^*$ with $\mathsf{exp}(w') = w$. This allows us to replace the constraint set $Z_i = \{w \in V_i^*W_iV_i^* \cap Z \mid \|w\| + \mathsf{int}(|w|) > \sigma(i)\}$ by the set $V_i'^*W_i'V_i'^*$. Note that for a word $w \in Z_i$ there may exist several words $w' \in V_i'^*W_i'V_i'^*$ with $\mathsf{exp}(w') = w$. This is not a problem: by taking the set $\mathbb{N} \times Z_i$ in the structure $\mathfrak{S}_1$, we basically take $\aleph_0$ many copies of w.

By our lifting construction from Section 3.2, every binary relation $\xrightarrow{\sigma}$ ($\sigma \in \Sigma$) on Trees_A is defined on $U' \cup U'' \subseteq \mathsf{Trees}_A^+$ and hence on $(\mathbb{N} \times U'^*U''U'^*) \cup U$. Using this, it follows that our labelled graph $\mathfrak{S}_1 = ((\mathbb{N} \times Z) \cup U,\ \Sigma,\ \{\xrightarrow{\sigma} \mid \sigma \in \Sigma\})$ is isomorphic to the labelled graph

$$\mathfrak{S}_2 = ((\mathbb{N} \times U'^*U''U'^*) \cup U,\ \Sigma,\ \{\xrightarrow{\sigma} \mid \sigma \in \Sigma\}).$$

The isomorphism maps the constraint set $L_i = (\mathbb{N} \times Z_i) \cup U_i$ to $(\mathbb{N} \times V_i'^*W_i'V_i'^*) \cup U_i$.

Step 3. In order to get rid of the direct product with $\mathbb{N}$ in $\mathbb{N} \times V_i'^*W_i'V_i'^*$ we add a new symbol \$ to the alphabet $U' \cup U''$. We lift the relations $\xrightarrow{\sigma}$ ($\sigma \in \Sigma$) from U''^* to $(U'' \cup \{\$\})^*$ in the standard way (\$ does not occur in the left-hand and right-hand sides of the relations $\xrightarrow{\sigma}$). Then, the labelled graph $\mathfrak{S}_2$ (and hence $\mathfrak{S}_1$) is isomorphic to the graph

$$\mathfrak{S}_3 = ((U' \cup \{\$\})^*U''(U' \cup \{\$\})^* \cup U,\ \Sigma,\ \{\xrightarrow{\sigma} \mid \sigma \in \Sigma\}).$$

The isomorphism maps U identically to U and the set $\mathbb{N} \times \{w\}$ (for $w \in U'^*U''U'^*$) is mapped bijectively onto the set of those words from $(U' \cup \{\$\})^*U''(U' \cup \{\$\})^* \setminus U$, whose projection onto the subalphabet U'' is w. Hence, the constraint set $\mathbb{N} \times V_i'^*W_i'V_i'^*$ is mapped to the set

$$(V_i' \cup \{\$\})^*W_i'(V_i' \cup \{\$\})^* \setminus U. \tag{4.20}$$

Step 4. In order to express in first-order logic that a word belongs to the above constaint set (4.20), we introduce another symbol #. Hence, our final alphabet is

$$\Gamma = U' \cup U'' \cup \{\$, \#\}.$$

With (4.16) and (4.19), the size of Γ can be estimated as

$$|\Gamma| = 2 + |U'| + |U''| \leq |A|^{2^{O(\ell)}p^2\cdot r^4}. \tag{4.21}$$

Next, we define a finite labelled graph $\mathfrak{T} = (\Gamma, \Sigma', \{\xrightarrow{a} \mid a \in \Sigma'\})$ with node set Γ as follows. The set of actions is

$$\Sigma' = \Sigma \cup \Gamma.$$

The set of transitions is defined as follows. By our lifting construction from Section 3.2, every binary relation $\xrightarrow{\sigma}$ ($\sigma \in \Sigma$) on Trees_A is defined on Γ (\$ and # do not occur in the left-hand and right-hand sides of the relations $\xrightarrow{\sigma}$). Moreover, for $a \in \Gamma$ we define the relation

$$\xrightarrow{a} \ = \ \{(a, \#)\}.$$

Finally, using the construction from Section 3.2, we define the labelled graph

$$\mathfrak{S}_4 = \mathfrak{T}^+.$$

We will construct a sentence φ_4 over the signature of $\mathfrak{S}_4$ such that $\mathfrak{S}_1 \models \varphi_1$ if and only if $\mathfrak{S}_4 \models \varphi_4$. Using the edge relations $\xrightarrow{a}$ ($a \in \Gamma$), we can express $x \in \Omega^+$ (for $\Omega \subseteq \Gamma$) as

$$\bigwedge_{a \in \Gamma \setminus \Omega} \neg \exists y : a(x, y).$$

Moreover, a constraint $|x|_\Omega \geq k$ (saying that there are at least k occurrences of symbols from Ω in the word x) can be expressed as

$$\exists y_1, \ldots, y_k \Big(\bigwedge_{j \neq j'} y_j \neq y_{j'} \wedge \bigwedge_{j \in [1,k]} \bigvee_{a \in \Omega} a(x, y_i) \Big).$$

This allows us to express e.g. $|x|_\Omega = k$ or $x \in \Omega$. Hence, a constraint $x \in L_i = (\mathbb{N} \times Z_i) \cup U_i$ in φ_1 can be replaced by the formula

$$\big(x \in (V_i' \cup W_i' \cup \{\$\})^+ \ \wedge \ |x|_{W_i'} = 1 \ \wedge \ x \notin U\big) \ \vee \ x \in U_i$$

of size $O(|\Gamma|)$. (for the correctness of this formula it is important that $V_i' \cap W_i' = \emptyset$ which follows from $V_i \cap W_i = \emptyset$). The size of the resulting sentence φ_4 can be bounded by $|\varphi| + O(\ell^3 + \ell^2 \cdot |\Sigma| + \ell \cdot |\Gamma|)$ and its quantifier rank is still $O(\ell)$.

We can now conclude the proof of Theorem 4.1. Recall that our overall goal is to check, whether $\mathfrak{G} \models \varphi$ holds. By the above constructions, this is equivalent to $\mathfrak{S}_4 \models \varphi_4$. By Theorem 4.2, this can be decided on an alternating Turing machine in time

$$O(|\Gamma|^{O(\ell)} \cdot |\varphi_4|) \leq \mathsf{poly}(|\Gamma|^{O(\ell)} + |\varphi| + |\Sigma|)$$

using $O(\ell) \leq O(|\varphi|)$ many alternations. Recall from (4.21) that $|\Gamma| \leq |A|^{2^{O(\ell)} p^2 \cdot r^4}$. Hence, we can bound the running time by $\mathsf{poly}(|A|^{2^{O(\ell)} p^2 \cdot r^4} + |\varphi| + |\Sigma|)$, which is doubly exponential in the input size. This concludes the proof of Theorem 4.1.

4.2. **Proof of Theorem 4.2.** Let us fix a finite labelled graph $\mathfrak{G} = (V, \Sigma, \{\xrightarrow{\sigma} \mid \sigma \in \Sigma\})$ and let $n = |V|$. We want to decide the first-order theory of $\mathfrak{G}^+$. For this we can w.l.o.g. assume that $n \geq 2$. Moreover, we can assume that $\Sigma = V \times V$ and that the edge from $a \in V$ to $b \in V$ is labelled with (a, b) (the original edge relations are definable by disjunctions in this new graph). Our decision procedure for the first-order theory of $\mathfrak{G}^+$ uses the method of Ferrante and Rackoff from Section 3.3 for the function $H(k, \ell) = n^{k+\ell+2} + k$. For this, we define a suitable equivalence relation $\equiv_{k,\ell}$ on k-tuples over V^*. The definition of this equivalence relation uses a simpler equivalence relation $\equiv_d$ defined on words, which corresponds to counting and comparing symbols up to the threshold d. The main combinatorial lemma for the equivalence $\equiv_{k,\ell}$ is Lemma 4.8. It rougly says that if $\overline{u} \equiv_{k,\ell} \overline{v}$ and $u \in V^*$, then one can always find a "short" word v such that $(\overline{u}, u) \equiv_{k,\ell} (\overline{v}, v)$. This corresponds to point (b) in Theorem 3.4. To apply the method of Ferrante and Rackoff, we also have to show that

$\overline{u} \equiv_{k,0} \overline{v}$ implies that $\overline{u}$ and $\overline{v}$ satisfy the same quantifer-free formulas in $\mathfrak{G}^+$ (point (a) in Theorem 3.4). This is stated in Lemma 4.9.

Recall that for a word $u \in A^*$ over a finite alphabet A and $a \in A$, $|u|_a$ denotes the number of occurrences of a in u. For $d \geq 1$ and $u, v \in A^*$, we write $u \equiv_d v$ if for all $a \in A$ the following holds:

- $|u|_a = |v|_a$ or
- $(|u|_a \geq d$ and $|v|_a \geq d)$

Note that $\equiv_d$ is an equivalence relation and that $u \equiv_{d+1} v$ implies $u \equiv_d v$.

Let A and B be finite alphabets. For two words $u = a_1 a_2 \cdots a_k \in A^*$ and $v = b_1 b_2 \cdots b_k \in B^*$ of the same length k we define the convolution

$$u \otimes v = (a_1, b_1)(a_2, b_2) \cdots (a_k, b_k) \in (A \times B)^k.$$

Lemma 4.7. *Let $\alpha \in \mathbb{N}$, $u, v \in \Gamma^*$ (where Γ is a finite alphabet), $u' \in V^*$ with $|u| = |u'|$, $u \equiv_{\alpha \cdot n} v$, and $|v| \geq \alpha \cdot n \cdot |\Gamma|$. Then there exists $v' \in V^*$ with $|v| = |v'|$ and $u \otimes u' \equiv_\alpha v \otimes v'$.*

Proof. Let $a \in \Gamma$ and $b \in V$. Consider the values $m_{a,b} = |u \otimes u'|_{(a,b)}$ and $n_a = |v|_a$. Finding a word $v' \in V^*$ such that $|v'| = |v|$ and $u \otimes u' \equiv_\alpha v \otimes v'$ is equivalent to finding numbers $n_{a,b}$ (which will be $|v \otimes v'|_{(a,b)}$) such that

- $\sum_{b \in V} n_{a,b} = n_a$ for all $a \in \Gamma$ and
- $m_{a,b} = n_{a,b}$ or $(m_{a,b} \geq \alpha$ and $n_{a,b} \geq \alpha)$ for all $a \in \Gamma$, $b \in V$.

Note that $u \equiv_{\alpha \cdot n} v$ implies

$$\sum_{b \in V} m_{a,b} = n_a \quad \text{or} \quad (\sum_{b \in V} m_{a,b} \geq \alpha \cdot n \text{ and } n_a \geq \alpha \cdot n)$$

for all $a \in \Gamma$. Also recall that $|V| = n$. We choose the numbers $n_{a,b}$ as follows, where $a \in \Gamma$:

- If $\sum_{b \in V} m_{a,b} = n_a$, then we set $n_{a,b} = m_{a,b}$ for all $b \in V$.
- If $\sum_{b \in V} m_{a,b} \geq \alpha \cdot n$ and $n_a \geq \alpha \cdot n$, then (since $|V| = n$) there must be at least one $b \in V$ with $m_{a,b} \geq \alpha$. We first set $n_{a,b} = m_{a,b}$ for all $b \in V$ with $m_{a,b} < \alpha$. For all remaining $b \in V$ (which satisfy $m_{a,b} \geq \alpha$) we set $n_{a,b}$ to some value $\geq \alpha$ such that the total sum $\sum_{b \in V} n_{a,b}$ becomes n_a. Since $n_a \geq \alpha \cdot n$ this is possible.

□

For all $k, \ell \in \mathbb{N}$ we define an equivalence relation $\equiv_{k,\ell}$ on the set $(V^*)^k$ of k-tuples of words over V as follows: Let $(u_1, \ldots, u_k), (v_1, \ldots, v_k) \in (V^*)^k$. Then $(u_1, \ldots, u_k) \equiv_{k,\ell} (v_1, \ldots, v_k)$ if and only if the following conditions hold:

(a) For all $1 \leq i, j \leq k$, $|u_i| = |u_j|$ if and only if $|v_i| = |v_j|$.
(b) For all $1 \leq i \leq k$, $u_i = v_i$ or $|u_i| \geq n^{k+\ell+1}$ and $|v_i| \geq n^{k+\ell+1}$.
(c) For all $1 \leq i \leq k$ the following holds: Let $1 \leq i_1 < i_2 \cdots < i_m \leq k$ be exactly those indices such that $|u_i| = |u_{i_1}| = \cdots = |u_{i_m}|$. Hence, $|v_i| = |v_{i_1}| = \cdots = |v_{i_m}|$ due to (a). Then $u_{i_1} \otimes u_{i_2} \otimes \cdots \otimes u_{i_m} \equiv_\alpha v_{i_1} \otimes v_{i_2} \otimes \cdots \otimes v_{i_m}$, where $\alpha = n^{\ell+1}$.

Lemma 4.8. *Let $k \geq 0$, $\ell > 0$, $(u_1, \ldots, u_k) \equiv_{k,\ell} (v_1, \ldots, v_k)$, and let $u_{k+1} \in V^*$. Then there exists $v_{k+1} \in V^*$ such that $|v_{k+1}| \leq n^{k+\ell+1} + k$ and $(u_1, \ldots, u_k, u_{k+1}) \equiv_{k+1,\ell-1} (v_1, \ldots, v_k, v_{k+1})$.*

Proof. Assume that $(u_1, \ldots, u_k) \equiv_{k,\ell} (v_1, \ldots, v_k)$ and let $u_{k+1} \in V^*$. We distinguish several cases:

Case 1. $|u_{k+1}| \neq |u_i|$ for all $1 \leq i \leq k$.

Case 1.1. $|u_{k+1}| < n^{k+\ell+1}$. Then, we must have $|u_{k+1}| \neq |v_i|$ for all $1 \leq i \leq k$ (if $|v_i| = |u_{k+1}| < n^{k+\ell+1}$, then we must have $u_i = v_i$ by (b) and hence $|u_{k+1}| = |u_i|$). We set $v_{k+1} = u_{k+1}$.

Case 1.2. $|u_{k+1}| \geq n^{k+\ell+1}$. Choose a number λ with $n^{k+\ell+1} \leq \lambda \leq n^{k+\ell+1} + k$ and $|v_i| \neq \lambda$ for all $1 \leq i \leq k$. We will find a word v_{k+1} such that $|v_{k+1}| = \lambda$ and $u_{k+1} \equiv_\alpha v_{k+1}$ for $\alpha = n^\ell$. Since $|u_{k+1}| \geq n^{k+\ell+1}$, there exists a symbol $a \in V$ such that $|u_{k+1}|_a \geq n^{k+\ell} \geq n^\ell = \alpha$. If $\lambda \geq |u_{k+1}|$, then we simply increase the number of occurrences of a in u_{k+1} until a word of length λ is reached. If $\lambda < |u_{k+1}|$, then $|u_{k+1}| > n^{\ell+1}$. Hence, there even exists $a \in V$ with $|u_{k+1}|_a > n^\ell$. We remove one of the occurrences of a in u_{k+1}. We can repeat this step until a word of length λ is reached.

Case 2. $|u_{k+1}| = |u_i|$ for some $1 \leq i \leq k$. Let $1 \leq i_1 < i_2 \cdots < i_m \leq k$ be exactly those indices such that $|u_{k+1}| = |u_{i_1}| = \cdots = |u_{i_m}|$. Let $u = u_{i_1} \otimes u_{i_2} \otimes \cdots \otimes u_{i_m}$ and $v = v_{i_1} \otimes v_{i_2} \otimes \cdots \otimes v_{i_m}$. Point (c) implies $u \equiv_{n^{\ell+1}} v$.

Case 2.1. $|u_{k+1}| < n^{k+\ell+1}$. Hence, we have $|u_i| < n^{k+\ell+1}$. This implies $|u_i| = |v_i| < n^{k+\ell+1}$ by (b). We set $u_{k+1} = v_{k+1}$. Note that $v_{i_1} = u_{i_1}, \ldots, v_{i_m} = u_{i_m}$ by (b). This implies

$$u_{i_1} \otimes u_{i_2} \otimes \cdots \otimes u_{i_m} \otimes u_{k+1} \equiv_\alpha v_{i_1} \otimes v_{i_2} \otimes \cdots \otimes v_{i_m} \otimes v_{k+1}$$

for all α.

Case 2.2. $|u_{k+1}| \geq n^{k+\ell+1}$. Hence, we have $|u| = |u_i| \geq n^{k+\ell+1}$. This implies $|v| = |v_i| \geq n^{k+\ell+1}$ by (b). We have to choose a word v_{k+1} with $|v_{k+1}| = |v_i|$ and $u \otimes u_{k+1} \equiv_\alpha v \otimes v_{k+1}$ for $\alpha = n^\ell$. This is possible by Lemma 4.7: Note that $\alpha \cdot n = n^{\ell+1}$ and thus $u \equiv_{\alpha \cdot n} v$. In order to apply Lemma 4.7 we set in addition $u' = u_{k+1}$, $v' = v_{k+1}$, and $\Gamma = V^m$. This implies

$$|v| \geq n^{k+\ell+1} = n^\ell \cdot n \cdot n^k \geq n^\ell \cdot n \cdot n^m = \alpha \cdot n \cdot |\Gamma|.$$

Hence, Lemma 4.7 can be applied indeed. □

Recall the definition of the infinite graph $\mathfrak{G}^+$ from (3.1).

Lemma 4.9. *If $(u_1, \ldots, u_k) \equiv_{k,0} (v_1, \ldots, v_k)$, then the tuples $(u_1, \ldots, u_k)$ and $(v_1, \ldots, v_k)$ satisfy the same quantifier-free formulas in the graph $\mathfrak{G}^+$.*

Proof. By symmetry, it suffices to prove the following two points:

(a) If $u_i = u_j$ then also $v_i = v_j$.

(b) If $u_i \xrightarrow{(a,b)} u_j$ for some $(a, b) \in V \times V$ then also $v_i \xrightarrow{(a,b)} v_j$.

Let us first prove (a). W.l.o.g. assume that $i = 1$ and $j = 2$. Let $2 < i_1 < i_2 < \cdots < i_m$ be those indices such that $|u_1| = |u_2| = |u_{i_1}| = \cdots = |u_{i_m}|$. Since $(u_1, \ldots, u_k) \equiv_{k,0} (v_1, \ldots, v_k)$, we get $|v_1| = |v_2| = |v_{i_1}| = \cdots = |v_{i_m}|$ and $u_1 \otimes u_2 \otimes u_{i_1} \otimes \cdots \otimes u_{i_m} \equiv_\alpha v_1 \otimes v_2 \otimes v_{i_1} \otimes \cdots \otimes v_{i_m}$ for $\alpha = n \geq 2$. Since $u_1 = u_2$, all symbols that occur in $u_1 \otimes u_2 \otimes u_{i_1} \otimes \cdots \otimes u_{i_m}$ are of the form $(a, a, \cdots)$ for some $a \in V$. Hence, the same has to hold for $v_1 \otimes v_2 \otimes v_{i_1} \otimes \cdots \otimes v_{i_m}$. But this means that $v_1 = v_2$.

For point (b), assume first that $a = b$. Thus, $u_i = u_j$ and $|u_i|_a > 0$. By point (a), we already know that $v_i = v_j$. If $i = j$, then we can w.l.o.g. assume that $i = j = 1$. Let $1 < i_1 < i_2 < \cdots < i_m$ be those indices such that $|u_1| = |u_{i_1}| = \cdots = |u_{i_m}|$. Since $(u_1, \ldots, u_k) \equiv_{k,0} (v_1, \ldots, v_k)$, we get $|v_1| = |v_{i_1}| = \cdots = |v_{i_m}|$ and $u_1 \otimes u_{i_1} \otimes \cdots \otimes u_{i_m} \equiv_\alpha v_1 \otimes v_{i_1} \otimes \cdots \otimes v_{i_m}$ for $\alpha = n \geq 2$. Since $|u_1|_a > 0$, the word $u_1 \otimes u_{i_1} \otimes \cdots \otimes u_{i_m}$ contains at least one occurrence of a symbol of the form $(a, \ldots)$. Hence, the same holds for $v_1 \otimes v_{i_1} \otimes \cdots \otimes v_{i_m}$. But this means that $v_1 \xrightarrow{(a,a)} v_1$. If $i \neq j$, then we can argue similarly.

Finally, let us assume that $a \neq b$. We must have $i \neq j$. W.l.o.g. assume that $i = 1$ and $j = 2$. Let us choose the indices $2 < i_1 < i_2 < \cdots < i_m$ as for the proof of point (a) above. Since $u_1 \xrightarrow{(a,b)} u_2$, the following holds for the word $u = u_1 \otimes u_2 \otimes u_{i_1} \otimes \cdots \otimes u_{i_m}$: u contains exactly one occurrence of a symbol of the form $(a, b, \ldots)$ and all other symbols in u are of the form $(c, c, \ldots)$ for $c \in V$. Again, the same has to be true for $v_1 \otimes v_2 \otimes v_{i_1} \otimes \cdots \otimes v_{i_m}$ (only here it is important that $n \geq 2$ and not just $n \geq 1$). Hence, $v_1 \xrightarrow{(a,b)} v_2$. $\square$

We can now prove Theorem 4.2. Let $\varphi = Q_0 x_0 \cdots Q_\ell x_\ell : \psi(x_0, \ldots, x_\ell)$ be a first-order formula of quantifier rank $\ell + 1$ over the signature of $\mathfrak{G}^+$, where $Q_0, \ldots, Q_\ell \in \{\forall, \exists\}$ and ψ is quantifier-free. For $0 \leq i \leq \ell$ let $L_i = \{w \in V^+ \mid |w| \leq n^{\ell+2} + i\}$. Theorem 3.4 (with $H(k, \ell) = n^{k+\ell+2} + k$), Lemma 4.8, and 4.9 imply that $\mathfrak{G}^+ \models \varphi$ if and only if

$$\mathfrak{G}^+ \models Q_1 x_0 \in L_0 \cdots Q_\ell x_\ell \in L_\ell : \psi(x_0, \ldots, x_\ell).$$

This can be decided on an alternating Turing machine in time $O(n^{\ell+2} \cdot |\varphi|)$ with ℓ alternations by guessing words $v_i \in L_i$ either existentially (if $Q_i = \exists$) or universally (if $Q_i = \forall$) and then verifying the statement $\psi(x_0, \ldots, x_\ell)$.

5. An $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$ lower bound

In this section, we will prove that there exists a fixed GTRS such that the corresponding ground tree rewrite graph has an $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$-complete first-order theory. This will be achieved using a suitable tiling problem. Tiling problems turned out to be an important tool for proving hardness and undecidability results in logic, see e.g. [4]. In a first step we will prove hardness for 2NEXP (doubly exponential non-deterministic time) in Section 5.2. In Section 5.3, we will finally push the lower bound to $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$.

5.1. **Tiling systems.** A *tiling system* is a tuple $S = (\Theta, \mathbb{H}, \mathbb{V})$, where Θ is a finite set of *tile types*, $\mathbb{H} \subseteq \Theta \times \Theta$ is a *horizontal matching relation*, and $\mathbb{V} \subseteq \Theta \times \Theta$ is a *vertical matching relation*. A mapping $\sigma : [0, k-1] \times [0, k-1] \to \Theta$ (where $k \geq 0$) is a *k-solution for S* if for all $(x, y) \in [0, k-1] \times [0, k-1]$ the following holds:

- if $x < k-1$, $\sigma(x, y) = \theta$, and $\sigma(x+1, y) = \theta'$, then $(\theta, \theta') \in \mathbb{H}$, and
- if $y < k-1$, $\sigma(x, y) = \theta$, and $\sigma(x, y+1) = \theta'$, then $(\theta, \theta') \in \mathbb{V}$.

Let $\mathsf{Sol}_k(S)$ denote the set of all k-solutions for S. Let $w = w_0 \cdots w_{n-1} \in \Theta^n$ be a word and let $k \geq n$. With $\mathsf{Sol}_k(S, w)$ we denote the set of all $\sigma \in \mathsf{Sol}_k(S)$ such that $\sigma(x, 0) = w_x$ for all $x \in [0, n-1]$. For a tiling system S we define its $(2^{2^n} \times 2^{2^n})$ *tiling problem* as follows:

$(2^{2^n} \times 2^{2^n})$ tiling problem for tiling system $S = (\Theta, \mathbb{H}, \mathbb{V})$

INPUT: A word $w \in \Theta^n$.
QUESTION: Does $\mathsf{Sol}_{2^{2^n}}(S, w) \neq \emptyset$ hold?

The following proposition is folklore, see also [4, 9].

Proposition 5.1. [4, 9] *There is some fixed tiling system S_0 whose $(2^{2^n} \times 2^{2^n})$ tiling problem is* 2NEXP*-hard under logspace reductions.*

5.2. **Hardness for** 2NEXP. Let us fix the tiling system $S_0 = (\Theta_0, \mathbb{H}_0, \mathbb{V}_0)$ of Proposition 5.1 whose tiling problem is hard for 2NEXP. We now define a fixed GTRS $\mathcal{R}_0 = (A, \Sigma, R)$ and prove that the first-order theory of $\mathfrak{G}(\mathcal{R}_0)$ is 2NEXP-hard under logspace reductions. We define

$$\begin{array}{rcl} A_0 & = & \{\heartsuit, \mathbb{1}, \mathbb{1}_\dagger, \mathbb{1}_\ddagger, \mathbb{0}, \mathbb{0}_\dagger, \mathbb{0}_\ddagger\}, \\ A_1 & = & \Theta_0, \\ A_2 & = & \{\bullet\}, \text{ and} \\ \Sigma & = & \{\ell, r, h, u, m_\dagger, m_\ddagger, \} \cup \Theta_0 \cup A_0. \end{array}$$

The set of rewrite rules R is given as follows:

(1) $X \xmapsto{X} X$ for each $X \in A_0$,
(2) $X \xmapsto{m_\dagger} X_\dagger$ for each $X \in \{\mathbb{1}, \mathbb{0}\}$ (this will correspond to *marking* a leaf),
(3) $X_\dagger \xmapsto{m_\ddagger} X_\ddagger$ for each $X \in \{\mathbb{1}, \mathbb{0}\}$ (this will correspond to *selecting* a leaf),
(4) $X_\dagger \xmapsto{h} \heartsuit$ for each $X \in \{\mathbb{1}, \mathbb{0}\}$,
(5) $\bullet(\heartsuit, \heartsuit) \xmapsto{u} \heartsuit$,
(6) $\theta(X_\ddagger) \xmapsto{\theta} \theta(X_\ddagger)$ for all $\theta \in \Theta_0$, $X \in \{\mathbb{1}, \mathbb{0}\}$,
(7) $\bullet(\heartsuit, X_\ddagger) \xmapsto{r} X_\ddagger$ for each $X \in \{\mathbb{1}, \mathbb{0}\}$, and
(8) $\bullet(X_\ddagger, \heartsuit) \xmapsto{\ell} X_\ddagger$ for each $X \in \{\mathbb{1}, \mathbb{0}\}$.

For the rest of this section we fix $\mathfrak{G}_0 = \mathfrak{G}(\mathcal{R}_0)$. Let us fix an input $w = \theta_0 \cdots \theta_{n-1} \in \Theta^n$ of the $(2^{2^n} \times 2^{2^n})$ tiling problem for S_0. Our goal is to compute in logspace from w a first-order sentence φ over Σ such that

$$\mathsf{Sol}_{2^{2^n}}(S_0, w) \neq \emptyset \quad \iff \quad \mathfrak{G}_0 \models \varphi.$$

For each subset $\Gamma \subseteq \Sigma$, we define $\xrightarrow{\Gamma} = \bigcup_{\gamma \in \Gamma} \xrightarrow{\gamma}$. The following lemma follows immediately from Lemma 3.3 (take the formula $\theta(x, y) = \bigvee_{\gamma \in \Gamma} \gamma(x, y)$).

Lemma 5.2. *Given a subset of actions $\Gamma \subseteq \Sigma$ and $j \in [0, 2^{n+1}]$ (in binary) one can compute in logspace a first-order formula $\Gamma^j(x, y)$ such that for all $t, t' \in \mathsf{Trees}_A$ we have $\mathfrak{G}_0 \models \Gamma^j(t, t')$ if and only if $t\ (\xrightarrow{\Gamma})^j\ t'$ in $\mathfrak{G}_0$.*

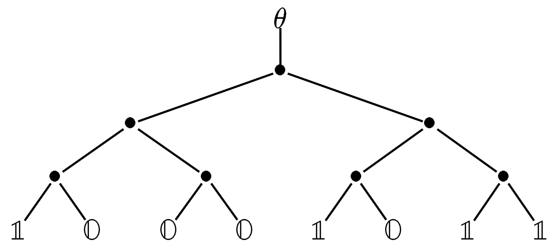
In case $\Gamma = \{\gamma\}$ is a singleton, we also write $\gamma^j(x, y)$ for the formula $\Gamma^j(x, y)$ of Lemma 5.2. Moreover, for subsets $\Gamma_1, \ldots, \Gamma_k \subseteq \Sigma$ and $j_1, \ldots, j_k \in \mathbb{N}$, we write $[\Gamma_1^{j_1} \cdots \Gamma_k^{j_k}](x, y)$ for the formula

$$\exists x_0, \ldots, x_k : \big(x_0 = x \wedge x_k = y \wedge \bigwedge_{i=1}^{k} \Gamma_i^{j_i}(x_{i-1}, x_i)\big).$$

A tree $t \in \mathsf{Trees}_A$ is a *tile tree* if $t = \theta(t')$ for some $t' \in \mathsf{Trees}_A$ such that the following holds:

- $\theta \in \Theta_0$,
- The label of every leaf of t' is from $\{\mathbb{0}, \mathbb{1}\}$.
- The distance of every leaf of t' to the root of t' is $n + 1$.
- Every internal node of t' is labeled with $\bullet$.

Example 5.3. This is a tile tree in case $n+1=3$:



Let us fix a tile tree t. Note that t has precisely $2^{n+1} = 2 \cdot 2^n$ leaves. Hence, there is a one-to-one correspondence between $[0, 2^{n+1} - 1]$ and leaves of t by means of their lexicographic order in t. For each leaf λ let $\mathsf{lex}(\lambda) \in [0, 2^{n+1} - 1]$ be the position of λ among all leaves w.r.t. the lexicographic order (starting with 0). The intention is that t represents the θ-labeled grid element $(M, N) \in [0, 2^{2^n} - 1]^2$, where each leaf λ that is a left (resp. right) child represents the $\lfloor \frac{\mathsf{lex}(\lambda)}{2} \rfloor^{\text{th}}$ least significant bit of the 2^n-bit binary presentation of M (resp. of N): In case λ is a left child, then $t(\lambda) = \mathbb{0}$ (resp. $t(\lambda) = \mathbb{1}$) if and only if the $\lfloor \frac{\mathsf{lex}(\lambda)}{2} \rfloor^{\text{th}}$ least significant bit of M equals 0 (resp. 1) and analogously if λ is a right child this corresponds to N. For the tile tree t from Example 5.3 we have $M = 1 + 4 + 8 = 13$ and $N = 8$.

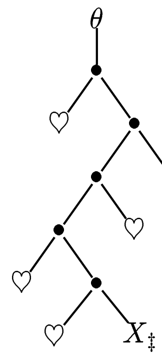
We say a leaf λ of a tree t is *marked* if $t(\lambda) = X_\dagger$ for some $X \in \{\mathbb{0}, \mathbb{1}\}$. We say a leaf λ of a tree t is *selected* if $t(\lambda) = X_\ddagger$ for some $X \in \{\mathbb{0}, \mathbb{1}\}$. A *marked tile tree* is a tree that can be obtained from a tile tree t by marking *every* leaf of t. For the rest of this section, let $D = 2^{n+1} - (n+2)$.

Lemma 5.4. *One can compute in logspace a first-order formula* $\mathsf{marked}(x)$ *such that for every tree* $t \in \mathsf{Trees}_{A \setminus \{\mathbb{0}_\ddagger, \mathbb{1}_\ddagger, \heartsuit\}}$ *with precisely* 2^{n+1} *marked leaves we have:* $\mathfrak{G}_0 \models \mathsf{marked}(t)$ *if and only if the marked leaves of* t *are the leaves of some (unique) marked tile subtree of* t.

Proof. The idea is to express the following: Whenever we select any of the 2^{n+1} marked leaves, we can execute from the resulting tree some sequence from the language

$$h^{2^{n+1}-1} u^D \{\ell, r\}^{n+1} \Theta_0.$$

Let us explain the intuition behind this. Assume we have selected exactly one of the 2^{n+1} marked leaves of t, and let t' be the resulting tree. First, note that after executing the sequence $h^{2^{n+1}-1}$ from t', we have replaced each of the marked leaves of t' with the symbol $\heartsuit$, reaching some tree t''. Second, when executing u^D from t'' we have reached, in case t contained a marked tile subtree, some tree t''' that has a chain of the following form as a subtree, where $X \in \{\mathbb{0}, \mathbb{1}\}$ and $\theta \in \Theta_0$:



Finally, from t''' we can now "shrink" this subtree to the tree $\theta(X_\ddagger)$ by executing some sequence from $\{\ell, r\}^{n+1}$ followed by executing θ. Formally, we define $\mathsf{marked}(x)$ as follows:

$$\forall y \left(m_\ddagger(x, y) \to \exists z : [h^{2^{n+1}-1} u^D \{\ell, r\}^{n+1} \Theta_0](y, z) \right)$$

Note that in this formula, y runs over all trees that can be obtained by selecting a marked leaf of x. Basically, in this way we quantify over all marked leaves of x. Note that the formula $\mathsf{marked}(x)$ ensures that the marked leaves of x are all at the same depth in x. $\square$

A *grid tree* is a tree t for which every leaf is inside a subtree of t that is a tile tree.

Lemma 5.5. *One can compute in logspace a first-order formula* $\mathsf{grid}(x)$ *such that for all* $t \in \mathsf{Trees}_A$ *we have* $\mathfrak{G}_0 \models \mathsf{grid}(t)$ *if and only if* t *is a grid tree.*

Proof. The formula grid will be a conjunction of the following two statements: (i) every leaf is either labeled with $\mathbb{0}$ or $\mathbb{1}$, (ii) for each leaf of t that we can mark via the action $m_\dagger$, we can mark $2^{n+1} - 1$ further leaves reaching some tree t' with $\mathfrak{G}_0 \models \mathsf{marked}(t')$. Formally, $\mathsf{grid}(x)$ is the conjunction of

$$\bigwedge_{a \in A_0 \setminus \{\mathbb{0}, \mathbb{1}\}} \neg a(x, x),$$

which realizes (i), and the formula

$$\forall y \left(m_\dagger(x, y) \to \exists z \big(m_\dagger^{2^{n+1}-1}(y, z) \wedge \mathsf{marked}(z) \big) \right),$$

which realizes (ii). $\square$

A *marked grid tree* is a tree that can be obtained from a grid tree t by replacing exactly one tile subtree of t by some marked tile tree. A *selected grid tree* is a tree that can be obtained from some marked grid tree t by selecting *precisely one* marked leaf λ of t. In that case, $\mathsf{lex}(\lambda) \in [0, 2^{n+1} - 1]$ is the lexicographical position of λ within the marked tile tree.

Lemma 5.6. *One can compute in logspace for each* $i \in [1, n+1]$ *a first-order formula* $bit_i(x)$ *such that for every selected grid tree* t *with selected leaf* λ *we have that the* i^{th} *least significant bit of* $\mathsf{lex}(\lambda)$ *is* 1 *if and only if* $\mathfrak{G}_0 \models bit_i(t)$.

Proof. We define $\mathrm{bit}_i(x) \;=\; \exists y : [h^{2^{n+1}-1} u^D \{\ell, r\}^{i-1} r](x, y)$. $\square$

Lemma 5.7. *One can compute for each $\circ \in \{<,=\}$ in logspace a first-order formula $\varphi_\circ(x,y)$ such that for every two selected grid trees t_1 and t_2 with selected leaves λ_1 and λ_2 we have $\mathfrak{G}_0 \models \varphi_\circ(t_1,t_2)$ if and only if $\mathsf{lex}(\lambda_1) \circ \mathsf{lex}(\lambda_2)$.*

Proof. We only treat the case when $\circ$ equals $<$; its definition should be self-explanatory:

$$\bigvee_{j\in[1,n+1]} \Big((\neg \mathrm{bit}_j(x) \wedge \mathrm{bit}_j(y)) \wedge \bigwedge_{1\le i<j} (\mathrm{bit}_i(x) \leftrightarrow \mathrm{bit}_i(y)) \Big) \qquad \square$$

Recall that the unique marked tile subtree of a marked grid tree t represents a θ-labeled grid element $(M,N) \in [0, 2^{2^n}-1]^2$ for some $\theta \in \Theta_0$. Therefore, let us define $M(t) = M$, $N(t) = N$, and $\Theta_0(t) = \theta$.

Lemma 5.8. *One can compute in logspace FO-formulas $\varphi_\theta(x)$, $\varphi_{i,M}(x,x')$, $\varphi_{i,N}(x,x')$, where $\theta \in \Theta_0$ and $i \in \{0,1\}$ such that for all marked grid trees t and t' the following holds:*
(1) $\mathfrak{G}_0 \models \varphi_\theta(t)$ if and only if $\Theta_0(t) = \theta$,
(2) $\mathfrak{G}_0 \models \varphi_{i,M}(t,t')$ if and only if $M(t)+i = M(t')$, and
(3) $\mathfrak{G}_0 \models \varphi_{i,N}(t,t')$ if and only if $N(t)+i = N(t')$.

Proof. For point (1) we define $\varphi_\theta(x)$ as follows:

$$\exists y : [m_\ddagger h^{2^{n+1}-1} u^D \{\ell, r\}^{n+1} \theta](x,y)$$

For the remaining points (2) and (3), we only give the formula $\varphi_{1,M}(x,x')$, i.e., we wish to express that for any two marked grid trees t and t' we have $\mathfrak{G}_0 \models \varphi_{1,M}(t,t')$ if and only if $M(t)+1 = M(t')$. Let us fix two marked grid trees t and t'. Assume we have selected among the 2^{n+1} marked leaves of t some leaf λ. Recall that λ represents one of the 2^n bit positions of $M(t)$ if and only if λ is a *left* child, otherwise it would represent a bit position of $N(t)$. Hence we will only be interested in leaves of t and t' which are left children. For this sake, let us express that the selected leaf of a selected grid tree z is a left child via the formula $\mathsf{left}(z)$:

$$\mathsf{left}(z) \quad = \quad \exists z', z'' \left(h(z,z') \wedge \ell(z',z'') \right)$$

Our formula $\varphi_{1,M}(x,y)$ is defined as follows:

$$\exists x', y' \left(m_\ddagger(x,x') \wedge m_\ddagger(y,y') \wedge \varphi_=(x',y') \wedge \mathbb{0}_\ddagger(x',x') \wedge \mathbb{1}_\ddagger(y',y') \wedge \mathsf{left}(x') \wedge \psi_1 \wedge \psi_2 \right).$$

Thus, we select a position $p \in [0, 2^n-1]$ that is set to 0 (resp. 1) in the binary representation of $M(t)$ (resp. $M(t')$). The formula $\psi_1(x,y,x',y')$ is defined as

$$\forall z \Big(\big(m_\ddagger(x,z) \wedge \varphi_<(z,x') \wedge \mathsf{left}(z) \big) \to \mathbb{1}_\ddagger(z,z) \Big) \wedge$$
$$\forall z \Big(\big(m_\ddagger(y,z) \wedge \varphi_<(z,y') \wedge \mathsf{left}(z) \big) \to \mathbb{0}_\ddagger(z,z) \Big).$$

It expresses that each bit at some position that is smaller than p is set to 1 (resp. 0) in the binary representation of $M(t)$ (resp. $M(t')$). The formula ψ_2 expresses that the binary representations of $M(t)$ and $M(t')$ agree on each position that is bigger than p. Formally, $\psi_2(x,y,x',y')$ is defined as

$$\forall u, v \Big(\big(m_\ddagger(x,u) \wedge m_\ddagger(y,v) \wedge \varphi_=(u,v) \wedge \varphi_<(x',u) \wedge \mathsf{left}(u) \big) \to \big(\mathbb{1}_\ddagger(u,u) \leftrightarrow \mathbb{1}_\ddagger(v,v) \big) \Big). \quad \square$$

We define the FO-formula $\mathsf{sol}(x)$ as the conjunction of the following formulas, where $\mathsf{mark}(z_1, z_2)$ is an abbreviation for $m_\dagger^{2^{n+1}}(z_1, z_2) \wedge \mathsf{marked}(z_2)$:

- x is a grid tree:
$$\mathsf{grid}(x)$$
- Whenever we mark two tile subtrees of x that represent the same grid element, their Θ-labels agree:
$$\forall y, z\Big((\mathsf{mark}(x,y) \wedge \mathsf{mark}(x,z) \wedge \varphi_{0,M}(y,z) \wedge \varphi_{0,N}(y,z)) \to \bigwedge_{\theta \in \Theta_0} (\varphi_\theta(y) \leftrightarrow \varphi_\theta(z))\Big)$$
- Whenever we mark a tile subtree of x that corresponds to the grid element (M,N) and $M < 2^{2^n} - 1$ there exists some tile subtree of x that corresponds to the grid element $(M+1, N)$ and the horizontal matching relation is satisfied:
$$\forall y\Big(\big(\mathsf{mark}(x,y) \wedge \exists z(m_\ddagger(y,z) \wedge \mathbb{O}_\ddagger(z,z) \wedge \mathsf{left}(z))\big) \to$$
$$\exists y'\big(\mathsf{mark}(x,y') \wedge \varphi_{1,M}(y,y') \wedge \varphi_{0,N}(y,y') \wedge \bigvee_{(\theta,\theta') \in \mathbb{H}_0} (\varphi_\theta(y) \wedge \varphi_{\theta'}(y'))\big)\Big)$$
- Analogously to the previous formula, we can express that whenever we mark a tile subtree of x that corresponds to the grid element (M,N) and $N < 2^{2^n} - 1$ there exists some tile subtree of x that corresponds to the grid element $(M, N+1)$ and the vertical matching relation is satisfied.

Finally we can construct a formula $\varphi_w(x)$ that guarantees that grid element $(j, 0)$ is labeled by θ_j (recall that $w = \theta_0 \cdots \theta_{n-1}$) for each $j \in [0, n-1]$:
$$\exists y_0, \ldots, y_{n-1}\Big(\bigwedge_{j \in [0,n-1]} (\mathsf{mark}(x, y_j) \wedge \varphi_{\theta_j}(y_j)) \ \wedge\ \forall z(m_\ddagger(y_0, z) \to \mathbb{O}_\ddagger(z,z)) \ \wedge$$
$$\bigwedge_{j \in [1,n-1]} (\varphi_{1,M}(y_{j-1}, y_j) \wedge \varphi_{0,N}(y_{j-1}, y_j))\Big)$$
Our final formula φ is defined as $\varphi = \exists x(\mathsf{sol}(x) \wedge \varphi_w(x))$. It follows by construction that
$$\mathsf{Sol}_{2^{2^n}}(S_0, w) \neq \emptyset \quad \Longleftrightarrow \quad \mathfrak{G}_0 \models \varphi.$$
With Proposition 5.1 we get:

Theorem 5.9. *The first-order theory of $\mathfrak{G}_0$ is* $\mathsf{2NEXP}$*-hard under logspace reductions.*

5.3. **Pushing hardness to $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$.** Let us fix a tiling system
$$S = (\Theta, \mathbb{H}, \mathbb{V}).$$
Given $\sigma, \sigma' \in \mathsf{Sol}_k(S)$ we say σ' *extends σ vertically* if $\sigma'(x, 0) = \sigma(x, k-1)$ for each $x \in [0, k-1]$. Let $\mathsf{Sol}_k(S, \sigma)$ be the set of all $\sigma' \in \mathsf{Sol}_k(S)$ such that σ' extends σ vertically. The standard encoding of Turing machine computations into tilings shows that there is a fixed tiling system $S_1 = (\Theta_1, \mathbb{H}_1, \mathbb{V}_1)$ such that the following problem is hard for $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$ under logspace reductions.

Linearly alternating $(2^{2^n} \times 2^{2^n})$ tiling problem (for S_1)

INPUT: A word $w = \theta_0\theta_1 \cdots \theta_{n-1} \in \Theta_1^n$, where n is odd.
QUESTION: Does $\exists \sigma_1 \in \mathsf{Sol}_{2^{2^n}}(S_1, w) \forall \sigma_2 \in \mathsf{Sol}_{2^{2^n}}(S_1, \sigma_1) \cdots \exists \sigma_n \in \mathsf{Sol}_{2^{2^n}}(S_1, \sigma_{n-1})$: true hold?

The idea is that the quantified solutions σ_i represent subcomputations of an alternating Turing-machine, where all states in the subcomputation are either existential (if i is odd) or universal (if i is even). Our definition of vertical extension of solutions ensures that these subcomputations can be combined into on single computation of the alternating Turing-machine. A similar encoding of alternating Turing machines by tiling systems can be found in [9].

Let $\mathfrak{G}_1$ be the fixed GTRS graph that is obtained from $\mathfrak{G}_0$ of Section 5.2 when we replace the tiling system S_0 by S_1.

Corollary 5.10. *The first-order theory of* $\mathfrak{G}_1$ *is hard for* $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$ *under logspace reductions.*

Proof. We recycle the proof presented in Section 5.2. We adapt the formulas constructed in Section 5.2 to the fixed tiling system S_1 (instead of S_0). Recall that we can compute in logspace a formula $\mathsf{sol}(x)$ such that for every tree t we have that $\mathfrak{G}_1 \models \mathsf{sol}(t)$ if and only if t corresponds to a 2^{2^n}-solution for S_1. It is an easy exercise to construct in logspace a formula ext such that for any two trees t and t' each satisfying sol we have $\mathfrak{G}_1 \models \mathsf{ext}(t, t')$ if and only if the solution corresponding to t' extends that of t vertically. We obtain that a word w (with $n = |w|$ odd) is a positive instance of the linearly alternating $(2^{2^n} \times 2^{2^n})$ tiling problem if and only if $\mathfrak{G}_1$ is a model of the sentence

$$\exists x_1 \left(\mathsf{sol}(x) \wedge \varphi_w(x) \wedge \forall x_2 \left((\mathsf{sol}(x_2) \wedge \mathsf{ext}(x_1, x_2)) \rightarrow \ \cdots \ \exists x_n \left(\mathsf{sol}(x_n) \wedge \mathsf{ext}(x_{n-1}, x_n)\right) \right) \right).$$

$\square$

We should remark that hardness for $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, \mathsf{poly}(n))$ can be also proved using the method of Compton and Henson [11] (monadic interpretation of addition on large numbers). The use of tilings has the advantage of giving an almost generic reduction. On the other hand, the method of [11] yields completeness under the slightly stronger log-lin reductions.

6. The first-order theory with regular unary predicates

For a GTRS $\mathcal{R} = (A, \Sigma, R)$ and a set of trees $L \subseteq \mathsf{Trees}_A$, we denote with $(\mathfrak{G}(\mathcal{R}), L)$ the structure that results from the labelled graph $\mathfrak{G}(\mathcal{R})$ by adding the set L as an additional unary predicate. Note that if L is a regular set of trees, then $(\mathfrak{G}(\mathcal{R}), L)$ is a tree automatic structure, and hence has a decidable first-order theory.

By the following result, our $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$ upper bound for the first-order theory of a ground tree rewrite graph does not carry over to ground tree rewrite graphs expanded by a regular unary predicate.

Theorem 6.1. *There exists a fixed GTRS* $\mathcal{R}_2 = (A, \Sigma, R)$ *and a fixed regular tree language* $L \subseteq \mathsf{Trees}_A$ *such that the first-order theory of* $(\mathfrak{G}(\mathcal{R}_2), L)$ *is non-elementary.*

Proof sketch. The proof idea is an adaption of the proof of Theorem 2 in [20] and is hence only shortly sketched. We reduce from the satisfiability problem for first-order logic over binary words. Binary words are considered as structures over the signature $(P_0, P_1, \leq)$, where P_0 and P_1 are unary relations (representing those positions, where the letter is 0 and 1, respectively), and where $\leq$ is the natural order relation on positions. The idea is that a tree $t \in \mathsf{Trees}_A$ (where $A_2 = \{\bullet\}$ and $A_0 = \{0,1\}$) corresponds to the unique word over $\{0,1\}$ that one obtains by simply reading the yield string (the sequence of node labels when traversing the leaves in lexicographic order) of t. Let $\mathsf{yield}(t)$ denote the yield string of t.

We translate a given first-order sentence φ over the signature $(P_0, P_1, \leq)$ into a first-order formula $\psi(x)$ over the signature of $(\mathfrak{G}(\mathcal{R}_2), L)$ such that for every tree $t \in \mathsf{Trees}_A$ we have: $\mathsf{yield}(t) \models \varphi$ if and only if $(\mathfrak{G}(\mathcal{R}_2), L) \models \psi(t)$. Assume that $x_1, \ldots, x_n$ are the variables that occur in φ. Bounding a variable x_i $(1 \leq i \leq n)$ of φ to a certain position in the word $\mathsf{yield}(t)$ is simulated by labelling the corresponding leaf of the tree t by a chain of unary symbols of length i. In order to keep the GTRS $\mathcal{R}_2$ fixed, this chain has to be built up in i rewrite steps that are controlled by the formula $\psi(x)$. In order to verify an atomic predicate $x_i < x_j$ in the tree t one has to check, whether the i-labelled node of t is lexicographically smaller than the j-labelled node. To do this using a fixed GTRS, one first replaces the chain of length i (resp., j) that identifies the position to which x_i (resp., x_j) is bound by a special constant a (resp. b). Again, this process has to be controlled by the formula $\psi(x)$. Finally, we can check $x_i < x_j$ using the regular set of trees that contain a unique a-labelled leaf and a unique b-labelled leaf, and the a-labelled leaf is lexicographically smaller than the b-labelled leaf. This regular set will be the set L in the theorem. □

7. Open problems

We proved that the uniform first-order theory of ground tree rewrite graphs belongs to the complexity class $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$ and that there exists a fixed ground tree rewritie graph with an $\mathsf{ATIME}(2^{2^{\mathsf{poly}(n)}}, O(n))$-complete first-order theory.

A complexity gap in this context exists for the first-order theory of the one-step rewrite graph of a semi-Thue system (word rewrite system): It is known to be 2EXPSPACE-hard and decidable but it is not known to be elementary [26]. One may try to tackle this problem using techniques similar to those used in this paper.

An important open problem concerning ground tree rewrite graph concerns bisimulation equivalence. It is not known whether the following problem is decidable: Given a ground tree rewrite system $\mathcal{R}$ and two trees s and t, are s and t are bisimilar in the graph $\mathfrak{G}(\mathcal{R})$? For pushdown graphs this problem is decidable [41] but not elementary, as was recently shown in [1]. A further question is the complexity of deciding bisimilarity between a ground tree rewrite system and a finite system, lying between PSPACE and coNEXP [20].

Acknowledgments

The work of the second author is supported by the DFG project GELO. We want to thank the referees of this paper for their valuable comments.

References

[1] M. Benedikt, S. Göller, S. Kiefer, and A. S. Murawski. Bisimilarity of pushdown automata is nonelementary. In *Proc. of LICS 2013*, pages 488–498. IEEE Computer Society, 2013.
[2] L. Berman. The complexity of logical theories. *Theor. Comput. Sci.*, 11:71–77, 1980.
[3] A. Blumensath. Automatic structures. Diploma thesis, RWTH Aachen, 1999.
[4] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Universitext. Springer-Verlag, Berlin, 2001.
[5] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. of CONCUR 1997*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.
[6] A. Bouajjani, M. Müller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In *Proc. of CONCUR 2005*, volume 3653 of *Lecture Notes in Computer Science*, pages 473–487. Springer, 2005.
[7] W. S. Brainerd. Tree generating regular systems. *Information and Control*, 14(2):217–231, 1969.
[8] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
[9] B. S. Chlebus. From domino tilings to a new model of computation. In *Proc. of Symposium on Computation Theory 1984*, volume 208 of *Lecture Notes in Computer Science*, pages 24–33. Springer, 1984.
[10] H. Comon, G. Godoy, and R. Nieuwenhuis. The confluence of ground term rewrite systems is decidable in polynomial time. In *Proc. of FOCS 2001*, pages 298–307. IEEE Computer Society, 2001.
[11] K. J. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Ann. Pure Appl. Logic*, 48(1):1–79, 1990.
[12] M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems. *Inf. Comput.*, 88(2):187–201, 1990.
[13] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proc. of LICS 1990*, pages 242–248. IEEE Computer Society, 1990.
[14] A. Dawar, M. Grohe, S. Kreutzer, and N. Schweikardt. Model Theory Makes Formulas Large. In *Proc. of ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*. Springer, 2007.
[15] A. Deruyver and R. Gilleron. The reachability problem for ground trs and some extensions. In *Proc. of TAPSOFT 1989, Vol.1*, volume 351 of *Lecture Notes in Computer Science*, pages 227–243. Springer, 1989.
[16] J. Ferrante and C. Rackoff. *The Computational Complexity of Logical Theories*. Number 718 in Lecture Notes in Mathematics. Springer, 1979.
[17] M. J. Fischer and M. O. Rabin. Super-exponential complexity of Presburgerarithmetic. In *Complexity of Computation*, volume 7 of *SIAM-AMS Proceedings*, pages 27–41. Amer. Math. Soc., 1974.
[18] H. Gaifman. On local and nonlocal properties. In *Proc. of Logic Colloquium '81*, pages 105–135. North Holland, 1982.
[19] G. Godoy, A. Tiwari, and R. M. Verma. Deciding confluence of certain term rewriting systems in polynomial time. *Ann. Pure Appl. Logic*, 130(1-3):33–59, 2004.
[20] S. Göller and A. W. Lin. The complexity of verifying ground tree rewrite systems. In *Proc. of LICS 2011*, pages 279–288. IEEE Computer Society, 2011.
[21] S. Göller and A. W. Lin. Refining the process rewrite systems hierarchy via ground tree rewrite systems. In *Proc. of CONCUR 2011*, volume 6901 of *Lecture Notes in Computer Science*, pages 543–558. Springer, 2011.
[22] S. Göller and M. Lohrey. The first-order theory of ground tree rewrite graphs. In *Proc. of FSTTCS 2011*, volume 13 of *LIPIcs*, pages 276–287. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011.
[23] W. P. Hanf. Model-theoretic methods in the study of elementary logic. In J. W. Addison, L. Henkin, and A. Tarski, editors, *The Theory of Models*, pages 132–145. North-Holland, Amsterdam, 1965.
[24] V. Kahlon and A. Gupta. On the analysis of interacting pushdown systems. In *Proc. of POPL 2007*, pages 303–314. ACM Press, 2007.
[25] A. Kartzow. FO model checking on nested pushdown trees. In *Proc. of MFCS 2009*, volume 5734 of *Lecture Notes in Computer Science*, pages 451–463. Springer, 2009.
[26] D. Kuske and M. Lohrey. Decidable first-order theories of one-step rewriting in trace monoids. *Theory Comput. Syst.*, 38(1):39–81, 2005.
[27] D. Kuske and M. Lohrey. Automatic structures of bounded degree revisited. *Journal of Symbolic Logic*, 76(4):1352–1380, 2011.

[28] A. W. Lin. Accelerating tree-automatic relations. In *Proc. of FSTTCS 2012*, volume 18 of *LIPIcs*, pages 313–324. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.
[29] C. Löding. Ground tree rewriting graphs of bounded tree width. In *Proc. of STACS 2002*, volume 2285 of *Lecture Notes in Computer Science*, pages 559–570. Springer, 2002.
[30] C. Löding. *Infinite Graphs Generated by Tree Rewriting.* PhD thesis, RWTH Aachen, 2003.
[31] C. Löding. Reachability problems on regular ground tree rewriting graphs. *Theory Comput. Syst.*, 39(2):347–383, 2006.
[32] D. Lugiez and P. Schnoebelen. Decidable first-order transition logics for pa-processes. *Inf. Comput.*, 203(1):75–113, 2005.
[33] J. Marcinkowski. Undecidability of the first order theory of one-step right ground rewriting. In *Proc. of RTA 1997*, volume 1232 of *Lecture Notes in Computer Science*, pages 241–253. Springer, 1997.
[34] R. Mayr. *Decidability and Complexity of Model Checking Problems for Infinite-State Systems.* PhD thesis, TU-Munich, 1998.
[35] R. Mayr. Process rewrite systems. *Inf. Comput.*, 156(1-2):264–286, 2000.
[36] D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.
[37] M. Oyamaguchi. The church-rosser property for ground term-rewriting systems is decidable. *Theor. Comput. Sci.*, 49:43–79, 1987.
[38] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
[39] S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *Proc. of TACAS 2005*, volume 3440 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2005.
[40] T. Rybina and A. Voronkov. Upper bounds for a theory of queues. In *Proc. of ICALP 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 714–724. Springer, 2003.
[41] G. Sénizergues. The bisimulation problem for equational graphs of finite out-degree. *SIAM J. Comput.*, 34(5):1025–1106, 2005.
[42] L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic.* PhD thesis, Department of Electrical Engineering, MIT, 1974.
[43] S. Tison. Fair termination is decidable for ground systems. In *Proc. of RTA 1989*, volume 355 of *Lecture Notes in Computer Science*, pages 462–476. Springer, 1989.
[44] A. W. To. *Model Checking Infinite-State Systems: Generic and Specific Approaches.* PhD thesis, LFCS, School of Informatics, University of Edinburgh, 2010.
[45] A. W. To and L. Libkin. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In *Proc. of FOSSACS 2010*, volume 6014 of *Lecture Notes in Computer Science*, pages 221–236. Springer, 2010.
[46] R. Treinen. The first-order theory of linear one-step rewriting is undecidable. *Theor. Comput. Sci.*, 208(1-2):179–190, 1998.
[47] H. Vogel. Turing machines with linear alternation, theories of bounded concatenation and the decision problem of first-order theories,. *Theor. Comput. Sci.*, 23:333–337, 1983.
[48] S. G. Vorobyov. The undecidability of the first-order theories of one step rewriting in linear canonical systems. *Inf. Comput.*, 175(2):182–213, 2002.
[49] I. Walukiewicz. Model checking CTL properties of pushdown systems. In *Proc. of FSTTCS 2000*, volume 1974 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2000.
[50] I. Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.