Deciding first-order properties for sparse graphs

Zdeněk Dvořák* Daniel Král'* Robin Thomas†

Abstract

We present a linear-time algorithm for deciding first-order logic (FOL) properties in classes of graphs with bounded expansion. Many natural classes of graphs have bounded expansion; for instance, graphs of bounded tree-width, all proper minor-closed classes of graphs, graphs of bounded degree, graphs with no subgraph isomorphic to a subdivision of a fixed graph, and graphs that can be drawn in a fixed surface in such a way that each edge crosses at most a constant number of other edges. We also develop an almost linear-time algorithm for deciding FOL properties in classes of graphs with locally bounded expansion; those include classes of graphs with locally bounded tree-width or locally excluding a minor.

More generally, we design a dynamic data structure for graphs belonging to a class $\mathcal{G}$ of graphs of bounded expansion. After a linear-time initialization the data structure allows us to test an FOL property in constant time, and the data structure can be updated in constant time after addition/deletion of an edge, provided the list of possible edges to be added is known in advance and their addition results in a graph in $\mathcal{G}$. In addition, we design dynamic data structure for testing Σ_1-properties or the existence of short paths between prescribed vertices in such classes of graphs. All our results hold for relational structures.

*Institute for Theoretical Computer Science (ITI), Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic. E-mail: {rakdver,kral}@kam.mff.cuni.cz. Institute for Theoretical Computer Science (ITI) is supported by Ministry of Education of Czech Republic as projects 1M0545.

†School of Mathematics, Georgia Institute of Technology, Atlanta, GA. E-mail: thomas@math.gatech.edu.

1 Introduction

A celebrated theorem of Courcelle [2] states that for every integer $k \geq 1$ and every property Π definable in monadic second-order logic (MSOL) there is a linear-time algorithm to decide whether a graph of tree-width at most k satisfies Π. While the theorem itself is probably not useful in practice because of the large constants involved, it does provide an easily verifiable condition that a certain problem is (in theory) efficiently solvable. Courcelle's result led to developing a whole new area of algorithmic results, known as algorithmic meta-theorems, see the survey [19]. For specific problems there is very often a more efficient implementation, for instance following the axiomatic approach of [28].

While the class of graphs of tree-width at most k is fairly large, it does not include some important graph classes, such as planar graphs or graphs of bounded degree. Courcelle's theorem cannot be extended to these classes unless P=NP, because testing 3-colorability is NP-hard for planar graphs of maximum degree at most four [14] and yet 3-colorability is expressible in monadic second order logic.

Thus in the attempt at enlarging the class of input graphs, we have to restrict to the set of properties we want to test. One of the first result in this direction was a linear-time algorithm of Eppstein [9, 10] for testing the existence of a fixed subgraph in planar graphs. He then extended his algorithm to minor-closed classes of graphs with locally bounded tree-width [11]. Testing a fixed subgraph can be defined in a first order logic (FOL) by a Σ_1-sentence and several generalization of Eppstein's work in this direction appeared. The most general results include:

- a linear time algorithm of Frick and Grohe [12] for deciding FOL properties of planar graphs,
- an almost linear-time algorithm of Frick and Grohe [12] for deciding FOL properties for classes of graphs with locally bounded tree-width,
- a fixed parameter algorithm of Dawar, Grohe and Kreutzer [3] for deciding FOL properties for classes of graphs locally excluding a minor, and
- a linear-time algorithm of Nešetřil and Ossona de Mendez [22] for deciding Σ_1-properties for classes of graphs with bounded expansion.

We generalize all these results in two different ways: we consider classes of graphs with bounded expansion, which generalize ones appearing in the mentioned results, and we also consider dynamic setting where the input graph can change during computation.

1.1 Classes of sparse graphs

In order to state our results, we need to present the concept of classes of graphs of bounded expansion, introduced by Nešetřil and Ossona de Mendez in [20] and in the series of journal papers [21, 22, 23]. Examples of classes of graphs with bounded expansion include proper minor-closed classes of graphs, classes of graphs with bounded maximum degree, classes of graphs excluding a subdivision of a fixed graph, classes of graphs that can be embedded on a fixed surface with bounded number of crossings per each edge and many others, see [25]. Many structural and algorithmic properties generalize from proper minor-closed classes of graphs to classes of graphs with bounded expansion, see [6, 26].

Let us define the graph classes of interest to us. All graphs considered in this paper are simple and finite. A class of graphs is *hereditary* if it is closed under taking subgraphs. An *r-shallow minor* of a graph G is a graph that can be obtained from G by removing some of the vertices and edges of G and then contracting vertex-disjoint subgraphs of radius at most r to single vertices (removing arising parallel edges to keep the graph simple). The *grad (greatest reduced average density) of rank r* of a graph G is equal to the largest average density of an r-shallow minor of G. The grad of rank r of G is denoted by $\nabla_r(G)$. In particular, $2\nabla_0(G)$ is the maximum average degree of a subgraph of G. If $\mathcal{G}$ is a class of graphs, then the class of r-shallow minors of graphs contained in $\mathcal{G}$ is denoted by $\mathcal{G}\nabla r$.

A function $f : \mathbb{N} \to \mathbb{R}^+$ *bounds the expansion* of a graph G if $\nabla_r(G) \leq f(r)$ for every integer $r \geq 0$. A class $\mathcal{G}$ of graphs has *bounded expansion* if there exists a function $f : \mathbb{N} \to \mathbb{R}^+$ that bounds the expansion of all graphs in $\mathcal{G}$. We will also say that the function f *bounds the expansion* of graphs in $\mathcal{G}$.

If $\mathcal{G}$ is a class of graphs and g_0 a real-valued function on $\mathcal{G}$, then

$$\limsup_{G\in\mathcal{G}} g_0(G)$$

is the supremum of all reals α such that $\mathcal{G}$ contains infinitely many graphs

G with $g_0(G) \geq \alpha$. A class $\mathcal{G}$ of graphs is *nowhere-dense* if

$$\lim_{r\to\infty} \limsup_{G \in \mathcal{G}\nabla r} \frac{\log ||G||}{\log |G|} \leq 1.$$

It can be shown that every class of graphs with bounded expansion is nowhere-dense [24], but the converse is not true.

On the other hand, every class of graphs with locally bounded tree-width or locally excluding a minor is nowhere-dense. Recall that a class $\mathcal{G}$ of graphs has *locally bounded tree-width* if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that the subgraph induced by the r-neighborhood of every vertex v of a graph $G \in \mathcal{G}$ has tree-width at most $f(r)$ for every $r \geq 0$. A class $\mathcal{G}$ *locally excludes a minor* if there exist graphs $H_1, H_2, \ldots$ such that no subgraph of any graph $G \in \mathcal{G}$ induced by the r-neighborhood of any vertex v of G contains H_r as a minor. Finally, a class $\mathcal{G}$ of graphs has *locally bounded expansion* if there exists function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+$ such that the $\nabla_r(G) \leq f(d, r)$ for every graph G that is a subgraph of the d-neighborhhod of a vertex in a graph from $\mathcal{G}$. Similarly, one can define classes of *locally nowhere-dense* graphs, but it turns out that every such class is a class of nowhere-dense graphs, too.

If L is a language (see Subsection 1.3 for the necessary logic theory definitions), the *Gaifman graph* of an L-structure A is the undirected graph G_A with vertex set $V(G_A) = V(A)$ and an edge between two vertices $a, b \in V(A)$ $(a \neq b)$ iff there exists $R \in L^r$ and a tuple $(a_1, \ldots, a_r) \in R^A$ such that $a, b \in \{a_1, \ldots, a_r\}$ or there exists a function $f \in L^f$ such that $b = f(a)$. We say that the relational structure A is *guarded* by a graph G with vertex set $V(G) = V(A)$ if $G_A \subseteq G$. Observe that if G belongs to a class of graphs with bounded expansion, then its maximum average degree is bounded and thus G contains a linear number of complete subgraphs [29]. It follows that the size $|A|$ of any L-structure A guarded by a graph belonging to a fixed class of graphs with bounded expansion is $O(|V(A)|)$.

1.2 Our results

We study complexity of deciding properties that can be expressed in terms of a first-order logic formula for classes of sparse graphs and sparse relational structures. Our two algorithmic results that unify the results mentioned at the beginning are the following. Recall that an algorithm is almost linear if its running time is bounded by $O(n^{1+\varepsilon})$ for every $\varepsilon > 0$, where n is the size of the input instance.

Theorem 1. *Let $\mathcal{G}$ be a class of graphs with bounded expansion, L a language and φ an L-sentence. There exists a linear time algorithm that decides whether an L-structure guarded by a graph $G \in \mathcal{G}$ satisfies φ.*

Theorem 2. *Let $\mathcal{G}$ be a class of graphs with locally bounded expansion, L a language and φ an L-sentence. There exists an almost linear time algorithm that decides whether an L-structure guarded by a graph $G \in \mathcal{G}$ satisfies φ.*

Our approach differs from the methods used to prove the results from [12, 3] mentioned above and is based on structural results on the existence of low tree-depth colorings for graphs with bounded expansion. After we announced our results in the survey paper [6] in June 2009, we learnt that Dawar and Kreutzer [4] gave an independent proof of Theorems 1 and 2. In fact, they proved Theorem 2 for the larger class of nowhere-dense graphs, using a different (more logic-oriented) technique.

In addition to designing algorithms for deciding FOL properties, we also consider dynamic setting in which relations to structures can be added and removed. In Sections 5–7, we design the following data structures (the first can be viewed as a dynamic version of Theorem 1):

- for every class $\mathcal{G}$ of graphs with bounded expansion, a language L and an L-sentence φ, a data structure that is initialized with a graph $G \in \mathcal{G}$ and an L-structure A guarded by G in time $O(n)$ where n is the number of vertices of G and that supports the following operations:
  - adding a tuple to a relation of A in time $O(1)$ provided A stays guarded by G,
  - removing a tuple from a relation of A in time $O(1)$, and
  - answering whether $A \models \varphi$ in time $O(1)$,
- for every class $\mathcal{G}$ of graphs with bounded expansion, an integer d_0 and a language L with no function symbols, a data structure that is initialized with a graph $G \in \mathcal{G}$ in time $O(n)$ and the empty L-structure A with $V(A) = V(G)$ where n is the number of vertices of G and that supports the following operations:
  - adding a tuple to a relation of A in time $O(1)$ provided A stays guarded by G,
  - removing a tuple from relation of A in time $O(1)$, and

  - answering whether $A \models \varphi$ for any Σ_1-L-sentence φ with at most d_0 variables in time $O(|\varphi|)$ and if so, outputting one of the satisfying assignments, and

  if $\mathcal{G}$ is only a class of nowhere-dense graphs, then the data structure supports

  - adding a tuple to a relation of A in time $O(n^\varepsilon)$ provided A stays guarded by G,
  - removing a tuple from a relation of A in time $O(n^\varepsilon)$, and
  - answering whether $A \models \varphi$ for any Σ_1-L-sentence φ with at most d_0 variables in time $O(|\varphi| + n^\varepsilon)$ and if so, outputting one of the satisfying assignments,

  where ε is any positive real number, and

- for every class $\mathcal{G}$ of graphs with bounded expansion and an integer d, a data structure supporting queries whether two vertices are at distance at most d provided the input graph stays in $\mathcal{G}$ with the following running times:

  - initialization time $O(n)$,
  - amortized time $O(\log^d n)$ for adding an edge, and
  - time $O(1)$ for removing an edge,

  where n is the order of the stored graph, and, if $\mathcal{G}$ is only a class of nowhere-dense graphs, then the data structure supports

  - initialization time $O(n^{1+\varepsilon})$,
  - amortized time $O(n^\varepsilon)$ for adding an edge, and
  - time $O(n^\varepsilon)$ for removing an edge,

  where ε is any positive real number.

The second of these data structures is needed in our linear-time algorithm for 3-coloring triangle-free graphs on surfaces [8], also see [7], and we describe its use in more detail in Section 8. The last one is inspired by a data structure of Kowalik and Kurowski [16, 18] for deciding whether two vertices of a planar graph are connected by a path of length at most k, where k is a fixed constant.

1.3 Logic theory definitions

In this subsection, we provide additional definitions needed in our exposition, mostly from logic theory. A *language* L consists of a disjoint union of a finite set L^r of *relation symbols* and a finite set L^f of *function symbols*. Each relation symbol $R \in L^r$ is associated with an integer $a(R) \geq 0$ called the arity of R. For our purposes, we will assume that all function symbols have arity one.

If L is a language, then an *L-structure* A is a triple $(V, (R^A)_{R\in L^r}, (f^A)_{f\in L^f})$ consisting of a finite set V and for each m-ary relation symbol $R \in L^r$ a set $R^A \subseteq V^m$, the *interpretation* of R in A, and for each function symbol $f \in L^f$ a function $f^A : V \to V$ of one variable, the *interpretation* of f in A. We define $V(A) := V$. For example, graphs may be regarded as L-structures A, where L is the language consisting of a single binary relation representing the edges with $V(A)$ being their vertex sets. We define the size $|A|$ of A to be $|V(A)| + \sum_{R\in L^r} |R^A| + |L^f||V(A)|$.

Assume that we have an infinite set of variables. An *L-term* is defined as follows:

1. each variable is an L-term, and
2. if $f \in L^f$ and t is an L-term, then $f(t)$ is an L-term.

Each L-term is obtained by a finite number of applications of these two rules. We say that an L-term is *simple* if it is a variable or is of the from $f(x)$ where $f \in L^f$ and x is a variable. A term t *appears* in a term t' if either $t = t'$ or $t' = f(t'')$ for some $f \in L^f$ and t appears in t''.

An *atomic L-formula* φ is either the symbol $\top$ (which represents a tautology), its negation $\bot$, $R(t_1, \dots, t_m)$ where R is an m-ary relation symbol of L and $t_1, \dots, t_m$ are L-terms or $t_1 = t_2$ where t_1 and t_2 are L-terms. A term t appears in φ if it appears in one of the terms $t_1, \dots, t_m$. An *L-formula* is defined recursively as follows: every atomic L-formula is an L-formula, and if φ_1 and φ_2 are L-formulas and x is a variable, then $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $\exists x\ \varphi_1$ and $\forall x\ \varphi_1$ are L-formulas. Every L-formula is obtained by a finite application of these rules. We write $t_1 \neq t_2$ as a shortcut for $\neg(t_1 = t_2)$.

A term t *appears* in an L-formula $\varphi_1 \vee \varphi_2$ if it appears in φ_1 or φ_2, and we define appearance for the other cases analogously. An L-formula is *simple* if all terms appearing in it are simple. A variable x *appears freely* in an L-formula φ if either φ is atomic and x appears in φ, $\varphi = \varphi_1 \vee \varphi_2$ or $\varphi = \varphi_1 \wedge \varphi_2$ and x appears freely in at least one of the formulas φ_1 and

φ_2, or $\varphi = \exists y\ \varphi'$ or $\varphi = \forall y\ \varphi'$, x is distinct from y and x appears freely in φ'. Occurences of x in the formula φ not inside the scope of a quantifier bounding x, i.e., those that witness that x appears freely in φ, are called *free* and the variables that appear freely in φ are also referred to as *free variables*. If φ is a formula, then the notation $\varphi(x_1, \ldots, x_n)$ indicates that all variables that appear freely in φ are among $x_1, \ldots, x_n$. An *L-sentence* is an L-formula such that no variable appears freely in it. If $\varphi(x_1, \ldots, x_n)$ is an L-formula and A is an L-structure, then for $v_1, \ldots, v_n \in V(A)$, we define $A \models \varphi(v_1, \ldots, v_n)$ in the usual way.

2 Classes of graphs with bounded expansion

In this section, we survey results on classes of graphs with bounded expansion and classes of nowhere-dense graphs that we need in the paper. A *rooted forest* F is a forest there every tree is an out-branching. The *depth* of F is the number of vertices of the longest oriented path in F. A *closure* of F is the undirected graph obtained from F by adding all edges between vertices joined by a directed path and forgetting the orientation of all the edges. The *tree-depth* of an (undirected) graph G is the smallest integer d such that G is a subgraph of the closure of a rooted forest of depth d. A vertex coloring of a graph G is a *low tree-depth coloring of order* K if the union of any s color classes, $s \leq K$, induces a subgraph of G with tree-depth at most s.

The existence of low tree-depth colorings with bounded numbers of colors is one of major structural results for graphs with bounded expansion. Before we state this result formally, we have to introduce more definitions.

Consider an orientation of G. Let G' be the graph obtained from G by adding all edges xy such that:

- there exists a vertex z such that G contains an edge oriented from x to z and an edge oriented from z to y (*transitivity*), or
- there exists a vertex z such that G contains an edge oriented from x to z and an edge oriented from y to z (*fraternality*).

We call G' the *augmentation* of the orientation of G. The following was shown by Nešetřil and Ossona de Mendez [21]:

Theorem 3. *There exist functions f_1 and f_2 with the following property. Let G be a graph with expansion bounded by g. Consider an orientation of*

G such that each vertex has in-degree at most D, and let G' be the augmentation of this orientation of G. Then the expansion of G' is bounded by a function $g'(r) = f_1(g(f_2(r)), D)$.

Let G be a graph from a class $\mathcal{G}$ with bounded expansion. Consider the following series of graphs: G_0 is obtained from G by orienting edges in such a way that the maximum in-degree of G_0 is at most $2\nabla_0(G)$. Let G_1 be the augmentation of this orientation of G_0, and orient G_1 so that the maximum in-degree of G_1 is at most $2\nabla_0(G_1)$; the edges present in G_0 do not necessarily have to preserve their orientation. In general, G_k is the augmentation of an orientation of G_{k-1} with maximum in-degree $2\nabla_0(G_{k-1})$. A greedy algorithm can be used to find such an orientation of G_{k-1}. The graph G_k is referred to as a *k-th augmentation* of G. By Theorem 3, the class $\mathcal{G}_k$ consisting of the graphs G_k obtained in the described way from the graphs in $\mathcal{G}$ has bounded expansion. It follows that $2\nabla_0(G_k) \leq d$ for some constant d depending only on the class $\mathcal{G}$ and k, and thus G_k is d-degenerate. In particular, the chromatic number of a k-th augmentation of a graph $G \in \mathcal{G}$ is bounded by a constant that depends on $\mathcal{G}$ only.

If G is an orientation of a graph, then any proper coloring of its $(3d^2+1)$-th augmentation G' of G is a low tree-depth coloring of order d [21], also see [6] for further details. Moreover, the subgraph H of G' induced by the vertices of any $s \leq d$ color classes contains a rooted forest F with depth at most s such that H is a subgraph of the closure of F. We refer to this property of the augmentation as *depth-certifying* and call F a *depth-certifying forest.* The next theorem from [21] follows.

Theorem 4. *Let $\mathcal{G}$ be a class of graphs with bounded expansion and d an integer. There exists k such that any proper coloring of a k-th augmentation of a graph $G \in \mathcal{G}$ is a low tree-depth coloring of order d and the k-th augmentation is depth-certifying. In particular, G has a low tree-depth coloring of order d with at most K colors where K depends on $\mathcal{G}$ and d only. Moreover, such a coloring of $G \in \mathcal{G}$ and corresponding depth-certifying forests can be found in linear time.*

Similarly, for classes of nowhere dense graphs, we obtain [24]:

Theorem 5. *Let $\mathcal{G}$ be a class of nowhere-dense graphs and d an integer. There exists k such that any proper coloring of a k-th augmentation of a graph $G \in \mathcal{G}$ is a low tree-depth coloring of order d and the k-th augmentation is depth-certifying. In particular, G has a low tree-depth coloring of* *order d with at most $O(n^{\varepsilon})$ colors for every $\varepsilon > 0$. Moreover, such a coloring of $G \in \mathcal{G}$ and corresponding depth-certifying forests can be found in almost linear time.*

3 Deciding FOL properties in linear time

In this section, we prove Theorem 1. We start with a lemma which allows us to remove quantifiers from a FOL formula (Lemma 9). However, we need more definitions. Let X be a set of L-terms. An *X-template* T is a rooted forest with vertex set $V(T)$ equipped with a mapping $\alpha_T : X \to V(T)$ such that $\alpha_T^{-1}(w) \neq \emptyset$ for every vertex w of T with no descendants. If φ is a quantifier-free L-formula, then a *φ-template* is an X-template where X is the set of all terms appearing in φ. The *depth* of T is the maximum depth of a tree of T. Two X-templates T and T' are isomorphic if there exists a bijection $f : V(T) \to V(T')$ such that

- f is an isomorphism of T and T' as rooted forests, in particular, w is a root of T if and only if $f(w)$ is a root of T', and
- $f(\alpha_T(t)) = \alpha_{T'}(t)$ for every L-term $t \in X$.

The number of X-templates with a given depth is finite as stated in the next proposition. The proof is straightforward and is left to the reader.

Proposition 6. *For every finite set of terms X and every integer d, there exist only finitely many non-isomorphic X-templates of depth at most d.*

Let X be a set of terms with variables $\{x_1, \ldots, x_n\}$. An *embedding* of an X-template T in a rooted forest F is a mapping $\nu : V(T) \to V(F)$ such that $\nu(r)$ is a root of F for every root r of a tree of T and ν is an isomorphism of T and the subforest of F induced by $\nu(V(T))$. Let S be an L-structure guarded by the closure of F, and $v_1, \ldots, v_n \in V(S)$. We say that the embedding ν is $(v_1, \ldots, v_n)$*-admissible* for S if for every term $t \in X$, we have $\nu(\alpha_T(t)) = t(v_1, \ldots, v_n)$ where $t(v_1, \ldots, v_n)$ denotes the element of $V(S)$ obtained by substituing v_i for x_i in the term t and evaluating the functions forming the term t (in particular, if $x_i \in X$, then $\nu(x_i) = v_i$). Note that ν is uniquely determined by S and the values $v_1, \ldots, v_n$.

If F is a rooted forest, then the function $p : V(F) \to V(F)$ is the *F-predecessor function* if $p(v)$ is the parent of v unless v is a root of F; if v is a root of F, $p(v)$ is set to be equal to v.

We now show that it can be tested by a quantifier-formula whether an embedding is admissible.

Lemma 7 (Testing admissibility)**.** *Let $d \geq 0$ be an integer, L a language including a function symbol p and X a finite set of terms with variables x_1, …, x_n. If T is an X-template of depth at most d, then there exists a quantifier-free formula $\xi_T(x_1, \dots, x_n)$ such that for every rooted forest F and every L-structure S guarded by the closure of F such that p^S is the F-predecessor function in S, and for every n-tuple $v_1, \dots, v_n \in V(S)$, the L-structure S satisfies $\xi_T(v_1, \dots, v_n)$ if and only if there exists a $(v_1, \dots, v_n)$-admissible embedding of T in the forest F for S.*

Proof. Let $q : V(T) \to V(T)$ be the T-predecessor function. Set $\xi_T(x_1, \dots, x_n)$ to the conjuction of all formulas

- $p^k(t) = p^{k'}(t')$ if $q^k(\alpha_T(t)) = q^{k'}(\alpha_T(t'))$, and
- $p^k(t) \neq p^{k'}(t')$ if $q^k(\alpha_T(t)) \neq q^{k'}(\alpha_T(t'))$,

for all pairs of not necessarily distinct terms $t, t' \in X$ and all pairs of integers k and k', $0 \leq k, k' \leq d + 1$ (note that including the formulas with $t = t'$ allows for testing the depth of t in F).

It is straightforward to show that for $v_1, \dots, v_n \in V(S)$, an $(v_1, \dots, v_n)$-admissible embedding of T in F for S exists if and only if $S \models \xi_T(v_1, \dots, v_n)$. □

The following lemma will be extremely useful in the proof of Lemma 9.

Lemma 8. *Let $d \geq 0$ be an integer, L a language, $\varphi(x_0, \dots, x_n)$ a simple quantifier-free L-formula that is a conjuction of atomic formulas and their negations, and T a φ-template. There exists an integer K and an $\overline{L}$-formula $\overline{\varphi}_T$ such that the following holds:*

- *$\overline{L}$ is the language with $\overline{L}^r = L^r \cup \{U_1, \dots, U_k\}$ and $\overline{L}^f = L^f \cup \{p\}$ where $U_1, \dots, U_k$ are new nullary or unary relations, $k \leq K$,*
- *$\overline{\varphi}_T$ is quantifier-free and the variables $x_1, \dots, x_n$ are the only variables that appear in $\overline{\varphi}_T$, but $\overline{\varphi}_T$ need not be simple, and*
- *for every rooted forest F with depth at most d and every L-structure S guarded by the closure of F, there exists an $\overline{L}$-structure $\overline{S}$ with $V(S) = V(\overline{S})$ such that for every $v_1, \dots, v_n \in V(S)$,*

$$S \models \varphi(v_0, v_1, \ldots, v_n) \textit{ for some } v_0 \in V(S) \textit{ such that there exists a } (v_0, \ldots, v_n)\textit{-admissible embedding of } T \textit{ in } F \textit{ for } S \textit{ if and only if } \overline{S} \models \overline{\varphi}_T(v_1, \ldots, v_n)$$

where $p^{\overline{S}}$ is the F-predecessor function and the relations $U_1^{\overline{S}}, \ldots, U_k^{\overline{S}}$ can be computed (by listing the singletons they contain) in linear time given F and S. The interpretation of other symbols of L is preserved in $\overline{S}$.

Proof. Fix a φ-template T and let q be the T-predecessor function. Let X be the set of all terms appearing in φ. Let ξ_T be the formula from Lemma 7. Finally, let $K = \max(\Delta, c) + 1$ where Δ is the maximum degree of T and c is the number of components of T.

Let $t = f(x_i)$ be a term appearing in φ, for some function symbol $f \in L^f$ and a variable x_i with $0 \leq i \leq n$. If $\alpha_T(t)$ is neither an ancestor nor a descendant of $\alpha_T(x_i)$, then for any choice of $v_1, \ldots, v_n \in V(S)$, there is no $(v_0, \ldots, v_n)$-admissible embedding of T for S because v_i and $f^S(v_i)$ are adjacent in the Gaifmann graph of S; in particular, one is a descendent of the other in F. Hence, we can set $\overline{\varphi}_T$ to $\perp$. So, we can assume the following:

$$\text{The images under } \alpha_T \text{ of all function images of each variable } x_i \text{ are ancestors or descendants of } \alpha_T(x_i). \tag{1}$$

If $\alpha_T(x_0)$ is an ancestor of a vertex $\alpha_T(t)$, say $q^k(\alpha_T(t)) = \alpha_T(x_0)$ for $k \geq 0$, where t is a term such that x_0 does not appear in t, then $\overline{\varphi}_T$ will be the formula obtained from $\varphi \wedge \xi_T$ by replacing each x_0 with the term $p^k(t)$. Clearly, $\overline{S} \models \overline{\varphi}_T(v_1, \ldots, v_n)$ if and only if there is a choice of v_0 in $V(F)$ such that $S \models \varphi(v_0, \ldots, v_n)$ and there is a $(v_0, \ldots, v_n)$-admissible embedding of T in F for S. In the rest, we can assume the following: So, we can assume the following:

$$\text{Every term } t \text{ such that } \alpha_T(t) \text{ is contained in the subtree of } T \text{ rooted at } \alpha_T(x_0) \text{ is a function image of } x_0. \tag{2}$$

We now define an auxiliary formula φ' to be the formula obtained from φ by replacing all subformulas of the form:

- $t = t'$ where t and t' are terms such that $\alpha_T(t) \neq \alpha_T(t')$,
- $t \neq t'$ where t and t' are terms such that $\alpha_T(t) = \alpha_T(t')$, or

- $R(t_1, \ldots, t_m)$ such that $\alpha_T(t_1), \ldots, \alpha_T(t_m)$ are not vertices of a clique in the closure of T,

with $\perp$ since such a subformula is not satisfied for any choice of v_0 for which there exists a $(v_0, \ldots, v_n)$-admissible embedding of T in F for S. It follows that for every v_0 such that there is a $(v_0, \ldots, v_n)$-admissible embedding of T in F for S, $S \models \varphi(v_0, \ldots, v_n)$ if and only if $S \models \varphi'(v_0, \ldots, v_n)$.

Suppose first that the tree of T containing the vertex $\alpha_T(x_0)$ also contains an α_T-image of another variable. Let v be the nearest ancestor of $\alpha_T(x_0)$ in T such that there exists a term $t_v \in X$ such that x_0 does not appear in t_v and v is an ancestor of $\alpha_T(t_v)$. Note that $v \neq \alpha_T(x_0)$ by (2). Let d_v be the depth of v in T, d_{x_0} the depth of $\alpha_T(x_0)$ and m the number of children of v in T. Let $t_1, \ldots, t_{m-1}$ be terms such that $\alpha_T(t_i)$, $1 \leq i \leq m-1$, are vertices of different subtrees rooted at a child of v and not containing $\alpha_T(x_0)$. Observe that the variable x_0 does not appear in t_1, $\ldots$, t_{m-1} by (1).

Let X_0 be the subset of X consisting of terms in which x_0 appears, and let T_0 be the X_0-template obtained from T by taking the minimal rooted subtree containing $\alpha_T(X_0)$ and the root of the tree containing $\alpha_T(x_0)$, and restricting the function α_T to X_0. Further, let X_0' be the subset of X consisting of X_0 and the terms t such that $\alpha_T(t)$ lies on the path between the root and $\alpha_T(x_0)$, and let X_0'' be the subset of X_0 consisting of the terms mapped to a descendant of v.

We define a unary relation $U_1(w)$ to be the set of elements w of F at depth $d_v + 1$ such that the subtree of w in F contains an element v_0 at depth d_{x_0} (in F) with the following properties:

- there is a (v_0)-admissible embedding of the template T_0 in F for S, and
- all clauses appearing in the conjuction φ' with terms from X_0' only and with at least one term from X_0'' are true with $x_0 = v_0$ and the terms $t \in X_0' \setminus X_0$, say $\alpha_T(t) = q^k(\alpha_T(x_0))$, replaced with $p^{\overline{S},k}(v_0)$.

The relation $U_1(w)$ can be computed as follows: for every element $v_0 \in V(S)$ at depth d_{x_0} of F, evaluate all terms in X_0 test whether the tree T_0 and the rooted subtree of F containing the values of the terms are isomorphic as rooted trees (this can be done in time linear in the size of T_0 which is constant). If they are isomorphic, evaluate the clauses in the conjuction φ' with terms from X_0' only and with at least one term from X_0'' with the terms in $X_0' \setminus X_0$ replaced with $p^{\overline{S},k}(v_0)$. If all of them are true, add the predecessor w of v_0 at depth $d_v + 1$ in F to U_1. Since the time spent by checking every vertex v_0 at depth d_{x_0} of F is constant, the time needed to compute U_1 is linear.

We further define a unary relation $U_i(w)$, $2 \le i \le m+1$, to be the set of elements w of F at depth d_v such that $U_1(w')$ for at least $i-1$ children w' of w. Clearly, the relations $U_i(w)$, $2 \le i \le m+1$, can be computed in linear time when the relation U_1 has been determined.

Let φ'' be the formula obtained from φ' by removing all clauses with terms from X_0' only that contain at least one term from X_0''. Observe that if t is a term in φ'' such that x_0 appears in t, i.e., $t \in X_0 \setminus X_0''$, then $\alpha_T(t)$ lies on the path between v and the root. Let φ''' be the formula obtained from φ'' by replacing every term t, in which x_0 appears, with $p^k(t_v)$, where k is the integer such that $\alpha_T(t) = q^k(\alpha_T(t_v))$. Let T' be the $(X \setminus X_0)$-template obtained from T by taking the minimal rooted subtree containing $\alpha_T(X \setminus X_0)$ and restricting the function α_T to $X \setminus X_0$. The formula $\overline{\varphi}_T$ will then be the conjuction of the following formulas:

- the formula $\varphi'''(x_1, \ldots, x_n)$,
- the formula $\xi_{T'}$ from Lemma 7 for the $(X \setminus X_0)$-template T', and
- the formulas

$$\left(\bigwedge_{i \in Y} U_1(p^{k_i - 1}(t_i))\right) \Rightarrow U_{|Y|+2}(p^k(t_v))$$

  for all subsets Y of the set $\{1, \ldots, m-1\}$ where k is the integer such that $q^k(\alpha_T(t_v)) = v$ and k_i, $i = 1, \ldots, m-1$, are the integers such that $q^{k_i}(\alpha_T(t_i)) = v$.

If $\overline{S} \models \overline{\varphi}_T(v_1, \ldots, v_n)$, then there is a $(v_1, \ldots, v_n)$-admissible embedding of T' in F and the vertex $v = p^k(t_v)$ has a son w such that $U_1(w)$ and the subtree of F rooted in w does not contain the value of any term appearing in $X \setminus X_0$ (this is guaranteed by the last type of formulas in the definition of $\overline{\varphi}_T$). In particular, the subtree rooted in w contains a vertex v_0 such that the $(v_1, \ldots, v_n)$-admissible embedding of T' can be extended to a $(v_0, \ldots, v_n)$-admissible embedding of T in F for S and all clauses in the conjuction of φ' containing terms from X_0' are satisfied with $x_0 = v_0$.

Since $\overline{S} \models \varphi'(v_1, \ldots, v_n)$, it follows that $S \models \varphi(v_0, \ldots, v_n)$. The argument that the existence of v_0 such that $S \models \varphi(v_0, \ldots, v_n)$ and the existence of a $(v_0, \ldots, v_n)$-admissible embedding of T in F for S implies that $\overline{S} \models \overline{\varphi}_T(v_1, \ldots, v_n)$ follows the same lines.

The case that the tree of T that contains the vertex $\alpha_T(x_0)$ does not contain α_T-image of another variable is handled similarly. In this case, the predicate U_1 is defined for the roots of the trees of F, and the predicates $U_2, \ldots, U_{m+1}$ (where m is the number of components of T) are nullary predicates such that U_i is true if U_1 is satisfied for at least $i-1$ roots of the trees in F. $\square$

We now prove the lemma that forms the core of our first algorithm.

Lemma 9 (Quantifier elimination lemma)**.** *Let $d \geq 0$ be an integer, L a language and φ a simple L-formula of the form $\exists x_0\ \varphi'$ such that φ' is a quantifier-free L-formula with free variables $x_0, \ldots, x_n$. There exists an integer K and an $\overline{L}$-formula $\overline{\varphi}$ such that the following holds:*

- *$\overline{L}$ is the language with $\overline{L}^r = L^r \cup \{U_1, \ldots, U_k\}$ and $\overline{L}^f = L^f \cup \{p\}$ where $U_1, \ldots, U_k$ are new nullary or unary relations and $k \leq K$,*
- *$\overline{\varphi}$ is quantifier-free and the variables $x_1, \ldots, x_n$ are the only variables that appear in $\overline{\varphi}$, but $\overline{\varphi}$ need not be simple, and*
- *for every rooted forest F with depth at most d and every L-structure S guarded by the closure of F, there exists an $\overline{L}$-structure $\overline{S}$ with $V(S) = V(\overline{S})$ such that for every $v_1, \ldots, v_n \in V(S)$,*

  $$S \models \varphi(v_1, \ldots, v_n) \text{ if and only if } \overline{S} \models \overline{\varphi}(v_1, \ldots, v_n)$$

  where $p^{\overline{S}}$ is the F-predecessor function and the relations $U_1^{\overline{S}}, \ldots, U_K^{\overline{S}}$ can be computed (by listing the singletons they contain) in linear time given F and S. The interpretation of other symbols of L is preserved in $\overline{S}$.

Proof. Let d, L and φ' be fixed. We assume without loss of generality that the formula φ' is in the disjunctive normal form and all the variables $x_0, \ldots, x_n$ appear in φ'. If $n = 0$, set $K = 1$, enhance L with a nullary relation U_1 and set $\overline{\varphi} = U_1$. The truth value of φ can be determined in linear time by testing all possible choices for x_0 among the elements of $V(S)$ and this determines whether the relation U_1 is true or not.

Hence, we assume $n \geq 1$ in the rest. The proof proceeds by induction on the length of φ'. If φ' is a disjunction of two or more conjuctions, i.e., $\varphi' = \varphi_1 \vee \varphi_2$, we apply induction to the formulas $\exists x_0 \varphi_1$ and $\exists x_0 \varphi_2$. We obtain integers K_1 and K_2, languages L_1 and L_2, an L_1-formula $\overline{\varphi}_1$ and an L_2-formula $\overline{\varphi}_2$. We assume that the new unary relation symbols of L_1 and L_2 are distinct and set $K = K_1 + K_2$, $\overline{L}^r = L_1^r \cup L_2^r$, $\overline{L}^f = L_1^f = L_2^f = L^f \cup \{p\}$ and $\overline{\varphi} = \overline{\varphi}_1 \vee \overline{\varphi}_2$.

In the rest, we assume that φ' is a conjunction. By Lemma 8, for every φ'-template T of depth at most d, there exists a quantifier-free $\overline{L}$-formula $\overline{\varphi}_T$ such that for every $v_1, \ldots, v_n \in V(S)$, $\overline{S} \models \overline{\varphi}_T(v_1, \ldots, v_n)$ if and only if there exists v_0 such that there is a $(v_0, v_1, \ldots, v_n)$-admissible embedding of T in F for S and $S \models \varphi'(v_0, v_1, \ldots, v_n)$. The desired formula $\overline{\varphi}$ is then obtained as the disjunction of the $\overline{L}$-formulas $\overline{\varphi}_T$ where the disjunction runs over all choices of φ'-templates T. Note that the number of choices of T is finite by Proposition 6. $\square$

In order to apply Lemma 9, the formula needs to be simple but the lemma produces a formula that need not be simple. The following lemma copes with this issue:

Lemma 10. *Let L be a language and $\varphi(x_1, \ldots, x_n)$ an L-formula with q quantifiers. There exists a language L' that extends L, a simple L'-formula $\varphi'(x_1, \ldots, x_n)$ with q quantifiers and an integer k with the following properties. For every L-structure A guarded by a graph G, there exists an L'-structure A' guarded by a k-th augmentation $G^{(k)}$ of the orientation of G containing an arc from u to v for every $u, v \in V(A)$ and $f \in L^f$ with $u = f^A(v)$ such that $A \models \varphi(v_1, \ldots, v_n)$ if and only if $A' \models \varphi'(v_1, \ldots, v_n)$ for any $v_1, \ldots, v_n \in V(A) = V(A')$. Moreover, the L'-structure can be computed in linear time in the size of $G^{(k)}$.*

Proof. Let k be the maximum number of compositions of unary functions appearing in φ. For every composition of unary functions appearing in φ, define a new unary function, extend the vocabulary by this function and replace the composition in φ by this new unary function. Let L' be the resulting language and φ' the resulting L'-formula. In this way, we obtain a simple L'-formula φ' with q quantifiers where only variables $x_1, \ldots, x_n$ appear freely. The L'-structure A' is the extension of A obtained by defining the values of new function symbols to correspond to the compositions of original functions. Finally, the L'-structure A' is guarded by a k-th augmentation $G^{(k)}$. $\square$

We are now ready to prove Theorem 1; we prove it in a stronger form needed in Section 4.

Theorem 11. *Let $\mathcal{G}$ be a class of graphs with bounded expansion, L a language and $\varphi(x_1,\dots,x_n)$ an L-formula. There exists a language L', an integer m and a linear-time algorithm that given an L-structure A guarded by $G \in \mathcal{G}$ computes an L'-structure A' guarded by an m-th augmentation $G^{(m)}$ of G, and a quantifier-free L'-formula $\varphi'(x_1,\dots,x_n)$ such that*

$$A \models \varphi(v_1,\dots,v_n) \text{ if and only if } A' \models \varphi'(v_1,\dots,v_n) \text{ for every } n\text{-tuple } v_1,\dots,v_n \in V(A) = V(A').$$

In particular, if $n = 0$, the algorithm decides whether $A \models \varphi$.

Proof of Theorems 1 and 11. The proof proceeds by induction on the number of quantifiers contained in φ. If φ is quantifier-free, then there is nothing to prove. Hence, assume that φ contains at least one quantifier. By Lemma 10, we can also assume that φ is simple. Indeed, an edge oriented from u to v to the graph G for every $u, v \in V(A)$ such that $u = f^A(v)$; since the number of function symbols f is bounded, the maximum in-degree is increased by a constant only. Lemma 10 then yields an integer k and a simple formula, which is used in what follows, guarded by a graph $H = G^{(k)}$.

Since $\forall x\ \psi$ is equivalent to $\neg\exists x\ \neg\psi$, we can assume that φ contains a subformula of the form $\exists x_0\psi$, where ψ is a formula with variables $x_0, x_1, \dots, x_N$, $N \geq n$, and with no quantifiers. Let N_0 be the number $N + 1$ increased by the number of distinct function images of $x_0, x_1, \dots, x_N$ appearing in ψ. Let $K = 3N_0^2 + 1$ and consider a coloring of a K-th augmentation $H^{(K)}$ of H; this coloring is a depth-certifying low-tree-depth coloring of H of order N_0 (see the discussion before Theorem 4). Let K_0 be the number of colors used by this coloring and C_i, $i = 1, \dots, K_0$, a unary relation containing vertices with the i-th color. Note that K_0 is bounded by a constant depending on $\mathcal{G}$ and K only. Clearly, computing the augmentation and determining the coloring can be performed in linear time.

For each function symbol $f \in L^f$ and $i = 1, \dots, K_0$, let C_i^f be the predicate defined so that $C_i^f(v)$ is true for $v \in V(A)$ if and only if the color of $f(v)$ is i. The colors of the variables and their function images appearing in ψ can be described by an N_0-tuple α of numbers between 1 and K_0. For such an N_0-tuple α, let φ_α be the conjuction of the terms of form $C_{\alpha_i}(x_i)$ and $C^f_{\alpha_j}(x_i)$ that verifies that the assignment of the colors of the values of $x_0, \dots, x_N$ and their function images are consistent with α.

Clearly, $\exists x_0 \psi$ is equivalent to the disjunction of the formulas $\exists x_0(\psi \wedge \varphi_\alpha)$ where the disjunction ranges through all choices of α.

For each function symbol $f \in L^f$, we introduce a new function symbol f_α defined by $f_\alpha(v) = f(v)$ if both v and $f(v)$ have a color in α and by $f_\alpha(v) = v$ otherwise. For each relation symbol $R \in L^r$ of arity greater than one, let R_α be defined by restricting R to the vertices with color α. Let A_α be the corresponding relational structure. Let φ''_α be the formula obtained from $\exists x_0(\psi \wedge \varphi_\alpha)$ by replacing each function symbol f by f_α and each relation symbol R of arity greater than one by R_α. For a fixed N_0-tuple α, the formula $\exists x_0(\psi \wedge \varphi_\alpha)$ is true for A if and only if φ''_α is true for A_α. Note that A_α is guarded by the graph H_α obtained from H by removing the edges incident with the colors not in α. Since $N_0 \leq K$, $H^{(K)}$ contains an out-branching F_α of depth at most N_0 whose closure contains H_α.

Apply Lemma 9 to the formula φ''_α and F_α, obtaining a formula ψ_α and a structure A'_α. We claim that for any choice of $x_1, \ldots, x_N$, the formula $A'_\alpha \models \psi_\alpha$ is satisifed in A'_α if and only if the formula $\exists x_0(\psi \wedge \varphi_\alpha)$ is satisfied in A. If the colors $x_1, \ldots, x_N$ or the colors of their function images do not agree with α, then both formulas are not satisfied. Otherwise, all the values of $x_1, \ldots, x_N$ correspond to the vertices of F_α and the formula ψ_α is in A'_α satisfied if and only if $\exists x_0(\psi \wedge \varphi_\alpha)$ is satisfied in A as stated in Lemma 9.

Note that the application of Lemma 9 for each N_0-tuple α extends the language L by a single unary function, which is determined by F_α, and several nullary and unary relations, in addition to the new symbols defined in A_α. Since the number of choices of α is bounded by a function of N_0 and K_0, the number of new functional and relational symbols depends only on the formula ψ and the class $\mathcal{G}$.

Replace now the subformula $\exists x_0 \psi$ in φ by the disjunction of the formulas ψ_α for all choices of α. The resulting formula is guarded by a graph $H^{(K)} = G^{(k+K)}$, and because it contains one less quantifier than φ, we can apply induction to it.

Since each application of Lemmas 9 and 10 in the induction can be performed in linear time and the number of their applications is bounded by a function depending on φ and $\mathcal{G}$ only, the reduction of φ to φ' and computing A' can be done in linear time. $\square$

4 Deciding FOL properties in graphs with locally bounded expansion

In this section, we prove Theorem 11. To do so, we will have to exploit locality of FOL formulas which is stated in the next theorem by Gaifman [13]. We need one more definition. If φ is an FOL formula, then $\varphi\langle x\rangle_r$ is the formula obtained from φ by relativizing all quantifiers to the r-neighborhood of x in the Gaifman graph, i.e., by restricting every universal and existential quantifier to the elements of the r-neighborhood of x.

Theorem 12. *Every first-order L-sentence is equivalent to a Boolean combination of sentence of the form*

$$\exists x_1, \dots, x_k \left(\bigwedge_{1 \le i < j \le k} d(x_i, x_j) > 2r \wedge \bigwedge_{1 \le i \le k} \psi\langle x_i\rangle_r \right)$$

for suitable integers k and r where $d(x_i, x_j)$ is the distance between x_i and x_j in the Gaifman graph and ψ is a FOL L-formula.

The *r-neighborhood* $N_r(v)$ of a vertex v in a graph G is the set of all of vertices of G at distance at most r from v. An *(r,s)-neighborhood cover* of a graph G is a collection $\mathcal{N}$ of vertex subsets of $V(G)$ such that every $A \in \mathcal{N}$ is contained in an s-neighborhood of a vertex of G and the r-neighborhood of every vertex of G is contained in some $A \in \mathcal{N}$. The size $||\mathcal{N}||$ of the neighborhood cover $\mathcal{N}$ is the sum of the cardinalities of its sets. The next lemma due to Peleg [27] says that every graph has a neighborhood cover of small size.

Lemma 13. *Let $k \ge 1$. There is an algorithm that for a graph G and $r \ge 1$ computes an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of G of size at most $O(|V(G)|^{1+1/k}$ and its running time is bounded by $O(\sum_{A\in\mathcal{N}} ||G[A]||)$. Moreover, every vertex v is associated with $A \in \mathcal{N}$ such that $N_r(v) \subseteq A$.*

An immediate corollary of Lemma 13 is the following:

Lemma 14. *Let $\mathcal{G}$ be a class of nowhere-dense graphs. For every $k \ge 1$ and every $r \ge 1$, there exists an algorithm that computes an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of a graph $G \in \mathcal{G}$ in time $O(|V(G)|^{1+2/k})$ such that the size of $\mathcal{N}$ is at most $O(|V(G)|^{1+1/k})$. Moreover, the algorithm associates every vertex of G with $A \in \mathcal{N}$ such that $N_r(v) \subseteq A$.*

We now follow the lines of arguments from [12]. Let us start with establishing the following.

Lemma 15. *Let L be a language, ψ an L-sentence, r an integer and $\mathcal{G}$ a class of graphs with locally bounded expansion. There exists an almost linear time algorithm that, given an L-structure S guarded by a graph $G \in \mathcal{G}$, computes the set W of elements $w \in V(S)$ such that $\psi\langle w\rangle_r$.*

Proof. Let k be an integer whose value is determined later. Apply Lemma 14 to get an $(r, 2kr)$-neighborhood cover $\mathcal{N}$ of G. Let S_A be the substructure of L induced by $A \in \mathcal{N}$ and G_A the subgraph of G induced by A. The subgraph G_A has expansion bounded by a function depending only on k and r.

Extend the language L to L_ℓ by introducing a new binary relation and extend S_A to S_ℓ by setting the new binary relation to represent the edges of G_A. Let $\psi_\ell(x_0)$ be the formula ψ where for each variable x_i appearing in ψ a subformula expressing that the distance of x_i from x_0 is at most r is added in the scope where x_i is quantified (note that this can be expressed by a Σ_1-formula, in particular, $\psi_\ell(x_0)$ is a FOL formula). Clearly, $S_A \models \psi\langle w\rangle_r$ if and only if $S_\ell \models \psi_\ell(w)$.

By Theorem 11, there exist an integer k_ℓ and an extension L'_ℓ of the language L_ℓ with the following property: in linear time, the L_ℓ-structure S_ℓ can be extended to an L'_ℓ-structure S'_ℓ guarded by an augmentation $G_A^{(k_\ell)}$ of G_A and a quantifier-free L'-formula $\psi'(x_0)$ can be computed such that

$$S_\ell \models \psi_\ell(w) \text{ if and only if } S'_\ell \models \psi'_\ell(w)$$

for every element $w \in V(S_A) = V(S_\ell) = V(S'_\ell)$. Consequently, it is possible to list all elements $w \in A$ such that $S_A \models \psi\langle w\rangle_r$ in linear time (if k is fixed).

The set W then contains the elements w such that w is associated with $A \in \mathcal{N}$ (in the sense of Lemma 14) and $S_A \models \psi\langle w\rangle_r$. If k is chosen to be the value of a function growing to infinity with $|V(S)|$ sufficiently slowly that the algorithm for listing $w \in W$ given the $(r, 2kr)$-neighborhood cover is almost linear, we obtain an almost linear time algorithm. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. By Theorem 12, it is enough to show that the truth of every L-sentence of the form

$$\exists x_1, \dots, x_k \left(\bigwedge_{1 \le i < j \le k} d(x_i, x_j) > 2r \wedge \bigwedge_{1 \le i \le k} \psi\langle x_i \rangle_r \right) \tag{3}$$

can be decided in almost linear time for L-structures guarded by a graph from $\mathcal{G}$.

Fix k, r and ψ and assume that an L-structure S guarded by $G \in \mathcal{G}$ is given. By Lemma 15, it is possible in almost linear time to list all elements $w \in V(S)$ such that $S \models \psi\langle w \rangle_r$. Let W be the set of all such elements. Choose $w_1 \in W$ arbitrarily. Then, choose w_2 arbitrary in $W \setminus N_{2r}(w_1)$, w_3 in $W \setminus (N_{2r}(w_1) \cup N_{2r}(w_2))$, until $W \setminus (N_{2r}(w_1) \cup \dots \cup N_{2r}(w_\ell))$ becomes empty. If $\ell \ge k$, the formula is true (set $w_i = x_i$, $i = 1, \dots, k$). If $\ell < k$, then all elements of W are at distance at most $2r$ from one of the elements $w_1, \dots, w_\ell$.

Let W' be the elements $w' \in V(S)$ such that $d(w', w) \le r$ for some $w \in W$, and let S' be the L-substructure of S induced by W'. Since every vertex of W is at distance at most $2r$ from one of the elements $w_1, \dots, w_\ell$, it holds that $W' \subseteq N_{3r}(w_1) \cup \dots \cup N_{3r}(w_\ell)$. Consequently, every component of the subgraph G' of G induced by W' is contained in the $3r\ell$-neigborhood of one of the vertices $w_1, \dots, w_\ell$ (the bound can be tight since the $3r$-neighborhoods of $w_1, \dots, w_\ell$ need not be disjoint). Observe that G' guards S'.

Since $\mathcal{G}$ is a class of graphs with locally bounded expansion, G' has bounded expansion (with bounds on grads depending only on k and r). By Theorem 1, the first order sentence

$$\exists x_1, \dots, x_k \left(\bigwedge_{1 \le i < j \le k} d(x_i, x_j) > 2r \wedge \bigwedge_{1 \le i \le k} x_i \in W \right) \tag{4}$$

can be decided in linear time for S'. Since two vertices x and x' are at distance at most $2r$ in G if and only if their distance in G' is at most $2r$, the sentence (3) is true in S if and only if the sentence (4) is true in S'. This concludes the proof. □

5 Dynamic data structures for Σ_1-queries

In this section, we provide a data structure for answering Σ_1-queries. The update time is constant but the price we have to pay is that the graph that guards the relational structure must be fixed before the computation starts. Before we start our exposition, we need to introduce more definitions.

Let L be a language with no function symbols. For an integer k, a *k-labelled* L-structure is an L-structure S with a partial injective mapping $\alpha : [1, k] \to V(S)$, i.e., α need not be defined for all integers between 1 and k. In our further consideration, we will also allow k to be equal to zero.

The *trunk* of a k-labelled L-structure S is the L-structure obtained from S by removing all relations with elements only from $\alpha([1, k])$. A k-labelled L-structure S is *hollow* if S is equal to its trunk. Two k-labelled L-structures S_1 and S_2 are *k-isomorphic* if their trunks are isomorphic through an isomorphism commuting with mappings α_1 and α_2. In particular, every k-labelled L-structure is k-isomorphic to its trunk.

Suppose now that an L-structure S is guarded by the closure of a rooted tree T. For a vertex v of T at depth d, let $P_T(v)$ denote the path from the root of T to v and $T\langle v\rangle$ the elements the subtree of v (including v itself). Then, $S\langle v\rangle$ denotes the set of all d-labelled L-structures S' such that S' is an induced substructure of S with elements only in $P_T(v) \cup T\langle v\rangle$ and $\alpha(i) = w$ for every vertex $w \in P_T(v) \cap V(S')$ at depth $i - 1$. If a vertex of $P_T(v)$ at depth $i - 1$ is not contained in S', then $\alpha(i)$ is not defined.

We are now ready to prove a lemma that contains a core of our data structure.

Lemma 16. *Let L be a language with no function symbols, d_0 a fixed integer and F a rooted forest of depth at most d_0. There exists a data structure representing an L-structure S guarded by the closure of F such that*

- *the data structure is initialized in linear time,*
- *the data structure representing an L-structure S can be changed to the one representing an L-structure S' by adding or removing a tuple from one of the relations in constant time provided that both S and S' are guarded by the closure of F, and*
- *the data structure decides in time bounded by $O(|\varphi|)$ whether a given Σ_1-L-sentence φ with at most d_0 variables is satisfied by S, and if so, it outputs one of the satisfying assignments.*

Proof. For every vertex v of F at depth d, we will store the following two lists:

- the list of all relations from S that contain v and all their elements are in $P_F(v)$, and

- the list of all (non-d-isomorphic) d-labelled hollow L-structures with at most d_0 elements that are d-isomorphic to a d-labelled L-structure contained in $S\langle v\rangle$.

Since there are only finitely many non-d-isomorphic d-labelled L-structures with at most d_0 elements for every $d \leq d_0$, the length of each list of the second type is bounded by a constant depending only on d_0 and L. If v is a non-leaf vertex of F, there will be a third list associated with v:

- the list of all (non-isomorphic) $(d+1)$-labelled hollow L-structures S' with at most d_0 elements that appear in the second list of at least one child of v; for each such S', there will be stored the list of all children of v whose second list contains S'.

In addition, there will be a global list of all (non-isomorphic) L-structures with at most d_0 elements that appear as induced L-substructures in S.

Let us describe how all these lists are initialized. The initialization of the first type of lists is trivial: just put each relation to the list of its element that is farthest from the root. This can clearly be done in constant time per relation.

Initialization of other types of lists is more difficult. Fix a tree T of F. We proceed from the leaves towards the root of T. Let v be a vertex of T at depth d. If v is a leaf of T at depth d, then the second list of v contains only those hollow d-labelled L-structures S' that are formed by vertices on $P(v)$ such that if $v \in V(S')$, then S' contains precisely all unary relations of S containing v, and if $v \notin V(S')$, then S' contains no relations at all.

Suppose now that v is not a leaf of T. The third list associated with v can be initialized by merging the second type of lists of children of v. We describe how it can be decided whether a d-labelled hollow L-structure S' should be contained in the list of v of the second type. Assume that $S\langle v\rangle$ contains a d-labelled hollow L-structure S'' that is d-isomorphic to S'.

Then, $V(S'')$ can be decomposed into disjoint subsets $V_0, V_1, \ldots, V_m$ such that $V_0 = V(S'') \cap P(v)$, each of the sets V_i, $i = 1, \ldots, m$, is fully contained in a subtree of a child v_i of v, and different subsets $V_1, \ldots, V_m$ are contained in different substrees. Observe that every relation of S'' must be contained in $V_0 \cup V_i$ for some $i = 1, \ldots, m$. Moreover, the only relations of S'' contained in V_0 are those that contain v.

Hence, the existence of S'' can be tested by considering all partitions of $V(S')$ into disjoint subsets $V_0, V_1, \ldots, V_m$ such that $\alpha([1, d]) \subseteq V_0$, $|V_0 \setminus \alpha([1, d])| \leq 1$ and every relation of S' is contained in $V_0 \cup V_i$ for some

$i = 1, \ldots, m$, and then testing the existence of children $v_1, \ldots, v_m$ such that the second list of v_i contains a $(d+1)$-labelled hollow L-structure $(d+1)$-isomorphic to the $(d+1)$-labelled hollow L-structure of S' induced by $V_0 \cup V_i$; if $|V_0 \setminus \alpha([1,d])| = 1$, then $\alpha(d+1)$ is defined to be equal to the unique element of $V_0 \setminus \alpha([1,d])$ and we also test whether the relations of S' containing $\alpha(d+1)$ are precisely those relations of S restricted to $P(v)$ that contain v, i.e., those in the first list of v.

We now describe how to test the existence of children $v_1, \ldots, v_m$. Let W be the set of children of v such that: if v has at most m children with their second list containing a $(d+1)$-labelled hollow L-structure $(d+1)$-isomoprhic to the substructure of S' induced by $V_0 \cup V_i$, then W contains all such children of v. If v has more than m such children, then W contains arbitrary m of these children. Clearly, $|W| \leq m^2 \leq d_0^2$. In order to test the existence of such children $v_1, \ldots, v_m$ of v, we form an auxiliary bipartite subgraph B: one part of B is formed by numbers $1, \ldots, m$ and the other part by children of v contained in W. A child $w \in W$ is joined to a number i if the second list of w contains a $(d+1)$-labelled hollow L-structure $(d+1)$-isomoprhic to the substructure of S' induced by $V_0 \cup V_i$.

If B has a matching of size m, then this matching determines the choice of children $v_1, \ldots, v_m$. On the other hand, if such children exist, B contains a matching of size m: indeed, if $v_i \notin W$, then v has at least m children whose second lists contains $(d+1)$-labelled hollow L-structure $(d+1)$-isomoprhic to the substructure of S' induced by $V_0 \cup V_i$, and we can change v_i to one of these m children that is different from $v_{i'}$ for $i' \neq i$. Hence, we can assume that $v_i \in W$ for every $i = 1, \ldots, m$ which implies that B has a matching of size m.

Since the order of B is at most $m^2 + m$ and the number of disjoint non-empty partitions of $V(S')$ to $V_0, \ldots, V_m$ is bounded, testing the existence of a d-labelled hollow L-structure S'' can be performed in constant time for v.

It remains to construct the global list containing L-structures S_0 with at most d_0 elements that appear in S as induced substructures. We proceed similarly as when determining the lists of inner elements of the forest F. For every L-structures S' with at most d_0 elements, we compute the list of trees of F that contain S', i.e., S' is contained in the second list of the root of F. Now, S_0 is an induced substructure of S' if and only if there exist element-disjoint L-structures $S_1, \ldots, S_m$ such that $S_0 = S_1 \cup \cdots \cup S_m$ and $S_1, \ldots, S_m$ appear in m mutually distinct trees of F. For each such partition of S_0 into $S_1, \ldots, S_m$, we can test whether $S_1, \ldots, S_m$ appear in the list of roots of m distinct trees of F using the auxiliary bipartite graph described earlier. Since all structures involved contain at most d_0 elements, this phase requires time linear in the number of trees of F.

We have shown that the data structure can be initialized in linear time. Let us now focus on updating the structure and answering queries. Consider a tuple $(v_1, \ldots, v_k)$ that is added to a relation R or removed from a relation R. Let r be the root of a tree in F that contains all the elements $v_1, \ldots, v_k$ and assume that $v_1, \ldots, v_k$ appear in this order on a path from r. By the definition, the only lists affected by the change are those associated with vertices on the path $P(v_k)$. Recomputing each of these lists requires constant time (we proceed in the same way as in the initialization phase except we do not have to swap through the children of the vertices on the path to determine which of them contain particular k-labelled hollow L-substructure S' in their lists). Since the number of vertices on the path $P(v_k)$ is at most d_0, updating the data structure requires constant time only.

It remains to describe how queries are answered. Let φ be a Σ_1-sentence with $d \leq d_0$ variables. We generate all possible L-structures S_0 with d variables and check whether they satisfy the formula φ. Let $\mathcal{S}_0$ be the set of those satisfying φ. The set $\mathcal{S}_0$ can be generated in time $O(|\varphi|)$ since L and d_0 are fixed.

Observe that S satisfies φ if and only if it has an induced substructure isomorphic to a structure in $\mathcal{S}_0$. This can be tested in constant time by inspecting the global list. Providing the satisfying assignment can be done in constant time if during the computation for each substructure a certificate why it was included in the list is stored (which requires constant time overhead only). □

The following two theorems immediately follow from Lemma 16. We only prove the first one since the proof of the second one is analogous.

Theorem 17. *Let L be a language with no function symbols, d_0 a fixed integer and $\mathcal{G}$ a class of graphs with bounded expansion. There exists a data structure representing a L-structure S such that*

- *given a graph $G \in \mathcal{G}$, the data structure is initialized in linear time with S being initially empty,*
- *the data structure representing an L-structure S can be changed to the one representing an L-structure S' by adding or removing a tuple from one of the relations in constant time provided that both S and S' are guarded by G, and*

- *the data structure decides in time bounded by $O(|\varphi|)$ whether a given Σ_1-L-sentence φ with at most d_0 variables is satisfied by S, and if so, it outputs one of the satisfying assignments.*

Proof. By Theorem 4, there exists a constant K (depending on $\mathcal{G}$ only) such that it is possible to find in linear time a low-tree-depth coloring of order d_0 of any graph $G \in \mathcal{G}$ together with forests witnessing the low tree depth. For every d_0 color classes of G, we apply Lemma 16. Since K is a constant, the number of data structures we maintain is bounded by a constant which depends on $\mathcal{G}$ only. If the given sentence φ is satisfied, it is also satisfied in one of the unions of color classes. This is tested using the data structure from Lemma 16. If φ is not satisfied in any of the auxiliary data structures, it is not satisfied in S either and we report that. $\square$

Theorem 18. *Let L be a language with no function symbols, k_0 a fixed integer, ε a positive real and $\mathcal{G}$ a class of nowhere-dense graphs. There exists a data structure representing an L-structure S such that*

- *given a graph $G \in \mathcal{G}$, the data structure is initialized in time $O(n^{1+\varepsilon})$ with S being initially empty,*
- *the data structure representing an L-structure S can be changed to the one representing an L-structure S' by adding or removing a tuple from one of the relations in time $O(n^{\varepsilon})$ provided that both S and S' are guarded by G, and*
- *the data structure decides in time bounded by $O(|\varphi| + n^{\varepsilon})$ whether a given Σ_1-L-sentence φ with at most k_0 variables is satisfied by S, and if so, it outputs one of the satisfying assignments.*

6 Dynamic data structure for FOL-properties

In this section, we present our dynamic data structure for testing FOL properties. The main result of this section reads as follows:

Theorem 19. *Let $\mathcal{G}$ be a class of graphs with bounded expansion, L a language and φ an L-sentence. There exists a data structure that is initialized with an n-vertex graph $G \in \mathcal{G}$ and an L-structure A guarded by G in time $O(n)$ and supports the following operations:*

- *adding a tuple to a relation of A in constant time provided A stays guarded by G,*
- *removing a tuple from a relation of A in constant time, and*
- *it answers in constant time whether $A \models \varphi$.*

Note that in Theorem 19, we do not allow to change function values of functions from L to simplify our exposition; this does not present a loss of generality as one can model functions as binary relations.

Theorem 19 follows from a dynamized version of Theorem 11 (we state the theorem in the variant with no free variables for simplicity).

Theorem 20. *Let $\mathcal{G}$ be a class of graphs with bounded expansion, L a language and φ an L-sentence. There exists a language L' and a quantifier-free L'-sentence φ' and a data structure representing an L'-structure A' that can be initialized with an n-vertex graph $G \in \mathcal{G}$ and an L-structure A guarded by G in time $O(n)$, $V(A) = V(A')$, and that satisfies:*

- *$A \models \varphi$ if and only if $A' \models \varphi'$, in particular, testing whether $A \models \varphi$ can be performed in constant time,*
- *adding a tuple to a relation of A can be done in constant time provided A stays guarded by G, and*
- *removing a tuple from a relation of A can be done in constant time.*

In order to prove Theorem 20, we first establish a dynamized version of Lemma 8.

Lemma 21. *Let $d \geq 0$ be an integer, L a language, $\varphi(x_0, \ldots, x_n)$ a simple quantifier-free L-formula that is a conjuction of atomic formulas and their negations, and T a φ-template. There exists an integer K and an $\overline{L}$-formula $\overline{\varphi}_T$ such that the following holds:*

- *$\overline{L}$ is the language with $\overline{L}^r = L^r \cup \{U_1, \ldots, U_k\}$ and $\overline{L}^f = L^f \cup \{p\}$ where $U_1, \ldots, U_k$ are new nullary or unary relations, $k \leq K$,*
- *$\overline{\varphi}_T$ is quantifier-free and the variables $x_1, \ldots, x_n$ are the only variables that appear freely in $\overline{\varphi}_T$, but $\overline{\varphi}_T$ need not be simple, and*
- *for every rooted forest F with depth at most d and every L-structure S guarded by the closure of F, there exists an $\overline{L}$-structure $\overline{S}$ with $V(S) = V(\overline{S})$ such that for every $v_1, \ldots, v_n \in V(S)$*

$S \models \varphi(v_0, v_1, \ldots, v_n)$ for some $v_0 \in V(S)$ such that there exists a $(v_0, \ldots, v_n)$-admissible embedding of T in F for S if and only if

$$\overline{S} \models \overline{\varphi}_T(v_1, \ldots, v_n)$$

where $p^{\overline{S}}$ is the F-predecessor function and the relations $U_1^{\overline{S}}, \ldots, U_k^{\overline{S}}$ can be computed (by listing the singletons they contain) in linear time given F and S. The interpretation of other symbols of L is preserved in $\overline{S}$, and

- *adding or removing a tuple to a relation of S results in adding and removing a constant number of singletons from unary relations among $U_1^{\overline{S}}, \ldots, U_k^{\overline{S}}$, and the changes to all relations $U_1^{\overline{S}}, \ldots, U_k^{\overline{S}}$ can be computed in constant time, provided S stays guarded by the closure of F.*

Proof. We need to describe how the relations $U_1^{\overline{S}}, \ldots, U_k^{\overline{S}}$ can be updated in constant time after adding or removing a tuple to a relation of S. Let us consider in more detail the case analyzed in the proof of Lemma 8 and leave to the reader the case mentioned at the end of the proof of Lemma 8. Recall (see the proof of Lemma 8 for notation) that $U_1(w)$ is a unary relation containing elements w of F at depth $d_v + 1$ such that the subtree of w in F contains an element v_0 at depth d_{x_0} (in F) with the following properties:

- there is a (v_0)-admissible embedding of the template T_0 in F for S, and
- all clauses appearing in the conjuction φ' with terms from X_0' only and with at least one term from X_0'' are true with $x_0 = v_0$ and the terms $t \in X_0' \setminus X_0$, say $\alpha_T(t) = q^k(\alpha_T(x_0))$, replaced with $p^{\overline{S},k}(v_0)$.

Since none of the functions of S changes, the first condition cannot change when adding or removing a tuple to a relation of S. The second one can change only when a tuple containing a term from X_0'' with $x_0 = v_0$ is added or removed from a relation. Since all the values of the terms in X_0'' with $x_0 = v_0$ appear only in a subtree of w, only a single element can be added to or removed from U_1. Based on the tuple we add or remove, we can identify which w can be added or remove to U_1 and using the data structure introduced in the proof of Lemma 16, we can test in constant time the existence of v_0 satisfying the second condition (note that the values of all terms from X_0 with $x_0 = v_0$ are in the subtree of w and the values of the terms in $X_0' \setminus X_0$ are on the path from w to the root).

Once the relation U_1 is updated, the relations $U_2, \dots, U_k$ can be updated in constant time as well: keep a counter at every vertex at depth d_v determining the number of children in U_1. $\square$

We are now ready to prove Theorem 20.

Proof of Theorem 20. Note that when the L-sentence φ is fixed in Theorem 11, the language L' and the L'-sentence φ' are also fixed. Hence, the only object that changes when the relations of A changes are relations in A'; the functions in A' stay the same and thus the m-th augmentation of the graph G from Theorem 11 that guards A' is independent of A as long as A stays guarded by G.

We now have to inspect the proofs of Lemma 9 and Theorem 11 in more detail. Since the graph G and all its augmentations appearing in the inductive proof of Theorem 11 stay the same, the coloring used in Lemma 9 also does not change when A gets altered. In particular, at each step of the inductive proof of Theorem 11, every rooted forest F to which Lemma 9 is applied stays the same. In the proof of Lemma 9, we replace use of Lemma 21 with use of Lemma 21 and observe that every change in S results in a constant number of changes in $\overline{S}$ and these changes can be identified in constant time. Hence, in the inductive proof of Theorem 11, a single change in A results in constantly many changes to the structure obtained in the first inductive step, which result in constantly many changes to the structure obtained the second inductive step (each change in the structure obtained in the first inductive step yields only constantly many changes), etc. Since the time to update the final L'-structure A' is constant for each of constantly many choices that propagates through the induction from a single change of A, the overall update time is constant. $\square$

7 Dynamic short path data structure

We now present a generalization of the data structure designed by Kowalik and Kurowski [16, 18]. Similarly to their data structure, we use the following result on maintaining orientations with low in-degree of Brodal and Fragerberg [1].

Theorem 22. *For any integers $d > 0$ and $D > 4d$, there exists an algorithm that maintains an orientation $\vec{G}$ of a d-degenerate n-vertex graph G that supports adding and deleting edges such that the maximum in-degree of*

$\vec{G}$ is at most D (provided, it stays d-degenerate during the whole computation). The orientation $\vec{G}$ is maintained explicitly, i.e., each vertex stores a list of in- and out-neighbors. The amortized time of adding an edge is $O(\log_{D/2d} n)$ which includes necessary recomputing of the orientation. Edges can be deleted in time $O(D)$.

Note that in Theorem 22, if d and D are constants, then the amortized time of adding an edge is $O(\log n)$.

Given an L-structure S, $u, v \in V(S)$ and $R \in L^r$ a binary relational symbol, $S + R(u, v)$ denotes the L-structure obtained from S by adding the pair (u, v) to R^S, and $S - R(u, v)$ denotes the L-structure obtained by removing the pair (u, v) from R^S.

Our first theorem in this section reads as follows:

Theorem 23. *Let L be a language containing only binary relation symbols, ℓ an integer, and $\mathcal{G}$ a hereditary class of graphs with bounded expansion. There exists a data structure representing an L-structure S with $G_S \in \mathcal{G}$ such that*

- *the data structure can be initialized in time $O(|S|)$,*
- *it can be transformed to represent $S + R(u, v)$, $R \in L^r$, in amortized time $O(\log^\ell |S|)$, assuming that $G_{S+R(u,v)} \in \mathcal{G}$,*
- *it can be transformed to represent $S - R(u, v)$, $R \in L^r$, in constant time, and*
- *a query whether*

$$(\exists x_1)(\exists x_2)\dots(\exists x_{\ell-1})R_1(u, x_1) \wedge R_2(x_1, x_2) \wedge \dots \wedge R_\ell(x_{\ell-1}, v)$$

  for a given pair of elements u and v and a given sequence $R_1, \dots, R_k$ can be answered in constant time. Moreover, in the positive case, a choice of $x_1, \dots, x_{\ell-1}$ that satisfy $R_1(u, x_1) \wedge R_2(x_1, x_2) \wedge \dots \wedge R_\ell(x_{\ell-1}, v)$ can also be found in constant time.

Proof. The proof proceeds by induction on ℓ. If $\ell = 1$, we set D to be $5d$ where d is the degeneracy of graphs in $\mathcal{G}$. The data structure maintains an orientation $\vec{G}_S$ of G_S with maximum in-degree at most D using the algorithm from Theorem 22 with a loop present at every vertex (so, the maximum in-degree is at most $D + 1$). Moreover, each edge of G_S is associated with a list of binary relations it corresponds to and each loop, say that is incident with a vertex v, is associated with a list of binary relations R such that $R(v, v)$ and

The initialization of the data structure can clearly be done in linear time. Adding a pair (u, v) to a relation R is also easy: if $u = v$, we just add R to the list of the loop at u. If $u \neq v$ and the edge uv is present in G_S, we add $R(u, v)$ to the list of the edge uv. If the edge is not present, we add the edge uv with the list containing $R(u, v)$ to the graph and apply algorithm from Theorem 22 to reorient the edges to keep the maximum in-degree at most $D + 1$.

Removing a pair (u, v) from a relation R is analogous: if $u = v$, we remove R from the list of the loop at u, and if $u \neq v$, we remove $R(u, v)$ from the list of the edge uv. In the latter case, if the list of the edge uv becomes empty, we call the algorithm from Theorem 22 to remove the edge uv from the graph. This requires constant time.

Finally, testing whether $R_1(u, v)$ for $u, v \in V(S)$ can also be done in constant time. If $u = v$, we search the list of the loop at u (which has a constant size since the number of different types of relations is determined by L). If $u \neq v$, we first search the list of in-neighbors of u whether it contains v and the list of in-neighbors of v whether it contains u. In the negative case, $R_1(u, v)$ is false. In the positive case, we check whether the list of the edge uv contains $R_1(u, v)$. Since the number of in-neighbors of each of u and v is bounded by D, the whole test requires constant time.

Assume now that $\ell > 1$ and the data structure for $\ell - 1$ with properties given in the statement of the theorem has been designed. We will use the data structure for $\ell - 1$ and the languge L' we now define: L' is the vocabulary L extended by binary relational symbols $Q_{RR'}$ for all ordered pairs R and R' of (not necessary distinct) binary relational symbols from L.

As in the case $\ell = 1$, we maintain an orientation $\vec{G}_S$ of the graph G_S with maximum in-degree at most $D+1$ with loops at all vertices such that each edge is associated with a list of binary relations corresponding to it. The L'-structure S' will be the extension of the L-structure S defined as:

$$Q_{RR'}(u, v) \text{ if and only if } \exists x R(u, x) \wedge R'(x, v) \wedge ux, vx \in \vec{G}_S.$$

For every $Q_{RR'}(u, v)$, the data structure will contain the list of all x such that $R(u, x) \wedge R'(x, v)$. Note that S' is guarded by the augmentation of $\vec{G}_S$ and augmentations of graphs from $\mathcal{G}$ have bounded expansion by Theorem 3.

Let us now turn our attention to the implementation of the data structure. The initialization is easy: we construct an orientation $\vec{G}_S$ of G_S with maximum in-degree at most D, add at a loop at each vertex and associate each edge of G_S with lists of binary relations corresponding to it. For every $x \in V(S)$, every in-neighbors u and v of x and every R in the list of ux and R' in the list of vx, we include (u, v) to $Q_{RR'}$ and x to the list of $Q_{RR'}(u, v)$. Note that $Q_{RR'}(u, v)$ if $R(u, v)$, $uv \in \vec{G}_S$ and $R'(v, v)$ since v is an in-neighbor of itself because of the loop at v. Since the maximum in-degree of $\vec{G}_S$ is at most D, extending S to S' requires linear time. Finally, we initialize the data structure for the L'-structure S'.

We now desribe removing a pair u, x from a relation R. Assume that the edge ux is oriented from u to x in $\vec{G}_S$. Remove R from the list of ux and if the list of ux becomes empty and $u \neq x$, remove ux from G_S. Then, remove x from the list of $Q_{RR}(u, u)$ and if the list becomes empty, remove u, u from Q_{RR} using the appropriate routine of the data structure for S'. Then, for every in-neighbor v of x distinct from u and every R' contained in the list of the edge vx, remove x from the list of $Q_{RR'}(u, v)$. If the list of $Q_{RR'}(u, v)$ becomes empty, remove the pair u, v from $Q_{RR'}$ in S'. Clearly, removing a pair from a relation can be done in constant time.

Adding a pair u, x to R is slightly more difficult. If the edge ux is present in G_S, we add R to the list of ux and for every in-neighbor v of x and every R' in the list of the edge vx, we add x to the list of $Q_{RR'}(u, v)$. If u, v is is not contained in $Q_{RR'}$, we add u, v to $Q_{RR'}$ using the appropriate routine of the data structure for S'. In particular, x is added to the list of $Q_{RR}(u, u)$.

If the edge ux is not present in G_S, we first add this edge to G_S and apply the algorithm from Theorem 22 to keep the maximum in-degree of $\vec{G}_S$ bounded. Each time an edge is reoriented, we proceed as follows: if the edge ab was originally oriented from a to b and becomes oriented from b to a, then for every in-neighbor a' of b, every R in the list ab and every R' in the list of $a'b$, remove b from the list of $Q_{RR'}(a, a')$ and if this list become empty, update S'. Then, for every in-neighbor b' of a, every R in the list ba and every R' in the list of $b'a$, add a to the list of $Q_{RR'}(b, b')$ and update S' if $Q_{RR'}(b, b')$ was not present before. Since the amortized number of reorientations is $O(\log n)$ and updating S' by adding a new pair to a relation requires amortized time $O(\log^{\ell-1} n)$, the amortized time needed for an addition of a pair to a relation in S is $O(\log^{\ell} n)$ as claimed.

Finally, we have to describe how to answer a query of the form

$$(\exists x_1)(\exists x_2)\dots(\exists x_{\ell-1}) R_1(u, x_1) \wedge R_2(x_1, x_2) \wedge \dots \wedge R_\ell(x_{\ell-1}, v)$$

for a given pair u and v of elements of $V(S)$. First, for every in-neighbor

x_1 of u such that $R_1(u, x_1)$, apply S' to check whether

$$(\exists x_2) \ldots (\exists x_{\ell-1}) R_2(x_1, x_2) \wedge \ldots \wedge R_\ell(x_{\ell-1}, v) \; .$$

If so, return the found $x_2, \ldots, x_{\ell-1}$ together with the in-neighbor x_1 of u. Since the number of in-neighbors of u is at most $D + 1$, this step requires constant time only.

Next, for every in-neighbor $x_{\ell-1}$ of v such that $R_{\ell-1}(x_{\ell-1}, v)$, apply S' to check whether

$$(\exists x_1) \ldots (\exists x_{\ell-2}) R_1(u, x_1) \wedge \ldots \wedge R_{\ell-1}(x_{\ell-2}, x_{\ell-1}) \; .$$

If so, return the found $x_1, \ldots, x_{\ell-2}$ together with the in-neighbor $x_{\ell-1}$ of v. Again, this step requires constant time only.

If both just described tests fail and there exist $x_1, \ldots, x_{\ell-1}$ satisfying the query, then the edge from ux_1 is oriented towards x_1 and the edge $x_{\ell-1}v$ is oriented towards $x_{\ell-1}$. Consequently, there exists i, $2 \leq i \leq \ell - 2$ such that both the edges $x_{i-1}x_i$ and x_ix_{i+1} are oriented towards x_i. Hence, for each $i = 2, \ldots, \ell - 2$, we make the following query in the data structure for S':

$$(\exists x_1) \ldots (\exists x_{i-1})(\exists x_{i+1}) \ldots (\exists x_{\ell-1}) R_1(u, x_1) \wedge \ldots \wedge R_{i-1}(x_{i-2}, x_{i-1}) \wedge$$

$$Q_{R_iR_{i+1}}(x_{i-1}, x_{i+1}) \wedge R_{i+2}(x_{i+1}, x_{i+2}) \wedge \ldots \wedge R_\ell(x_{\ell-1}, v) \; .$$

If any of these queries is positive, then we return the values $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{\ell-1}$ given by the query together with any x_i contained in the list of the relation $Q_{R_iR_{i+1}}(x_{i-1}, x_{i+1})$. If all the queries are negative, then there is no choice of $x_1, \ldots, x_{\ell-1}$ and we report so. Since ℓ is fixed, the total query time is constant. □

Considering classes of nowhere-dense graphs, one can obtain the following theorem:

Theorem 24. *Let L be a language containing only binary relation symbols, ℓ an integer, ε a positive real, and $\mathcal{G}$ a hereditary class of nowhere-dense graphs. There exists a data structure representing an L-structure S, where $n = |V(S)|$, with $G_S \in \mathcal{G}$ such that*

- *the initial data structure representing S can be built in time $O(n^{1+\varepsilon})$,*
- *it can be transformed to represent $S + R(u, v)$, $R \in L^r$, in amortized time $O(n^\varepsilon)$, assuming that $G_{S+R(u,v)} \in \mathcal{G}$,*

- *it can be transformed to represent* $S - R(u, v)$, $R \in L^r$, *in time* $O(n^\varepsilon)$, *and*
- *it answers queries whether*

$$(\exists x_1)(\exists x_2)\ldots(\exists x_{\ell-1})R_1(u, x_1) \wedge R_2(x_1, x_2) \wedge \ldots \wedge R_\ell(x_{\ell-1}, v)$$

for a given pair of vertices u *and* v *and a given sequence* $R_1, \ldots, R_k$ *in time* $O(n^\varepsilon)$. *Moreover, in the positive case, a choice of* $x_1, \ldots, x_{\ell-1}$ *that satisfy* $R_1(u, x_1) \wedge R_2(x_1, x_2) \wedge \ldots \wedge R_\ell(x_{\ell-1}, v)$ *can also be found in time* $O(n^\varepsilon)$.

8 Application

In [8], we prove the following theorem.

Theorem 25. *For every surface* Σ *there is a linear-time algorithm to decide whether a triangle-free graph drawn in* Σ *is* 3*-colorable.*

Previously, it was not known whether the problem was polynomial-time solvable. The question was raised by Thomassen, who obtained a couple of related and deep theorems. A full exposition of the algorithm is lengthy and relies on other results not related to the present paper. What *is* related to this paper is the following question. Suppose that G is a triangle-free graph drawn in Σ that is actually 3-colorable. By Theorem 25 we can verify this in linear time, but can we actually find a 3-coloring in linear time?

We were able to answer this question in the affirmative, using the data structure of Theorem 17. Since the algorithm uses ingredients from covering space theory that would distract from the data structure contents, let us describe our algorithm in the special case when Σ is the sphere. This special case already illustrates the difficulties that needed to be overcome.

The following is classical theorem of Grötzsch [15].

Theorem 26. *Every triangle-free planar graph is* 3*-colorable.*

Thus the algorithm from Theorem 25 is trivial when Σ is the sphere: always answer "yes". However, finding a 3-coloring in linear time is challenging. Here is the difficulty. All known proofs of Theorem 26 proceed by collapsing facial 4-cycles. More precisely, let G be a triangle-free plane graph, let $v_1v_2v_3v_4$ be the boundary of a face of length four, let G_{13} be the graph obtained from G by identifying v_1 and v_3 inside the face $v_1v_2v_3v_4$, and let G_{24} be defined analogously. It is easy to see that either G_{13} or G_{24} or both are triangle-free. This can be used for an inductive proof and an algorithm. Once the number of faces of size four is sublinear in the order of G, another reduction can be identified and used [8].

Thus all we need to do is to decide in amortized constant time which of G_{13} and G_{24} is triangle-free. That can be done in constant time (after linear-time initialization) [16, 18] as long as the graph G does not change. But here the graph changes: in the next iteration we need to check whether a different graph is triangle-free, and so on. Kowalik [17] was able to modify the data structure of [16, 18] and to design an $O(n \log n)$ algorithm to 3-color planar graphs. This was improved by Dvořák, Kawarabayashi and Thomas [5] to linear time using a different method than we present here.

If G_{13} is triangle-free, then we say that G_{13} is an *elementary reduction* of G. Thus every triangle-free plane graph with at least one face of length four has an elementary reduction. A graph H is a *reduction* of a triangle-free plane graph G if it is obtained from G by repeatedly taking elementary reductions. If a 3-coloring of a reduction H of G is found, a 3-coloring of G can be obtained in the obvious way: if vertices u and v of an intermediate graph J were identified to create a vertex w of an elementary reduction J' of J, then each vertex of $J \backslash \{u, v\}$ will retain its color from the coloring of J', and u and v will both receive the color of w.

In view of what was said, we need the following theorem to be able to find a 3-coloring. Each time, Theorem 27 is applied a linear number of faces of size four disappear and thus the graph eventually reduces in linear time to a graph with a sublinear number of faces of size four which we can color using methods from [8].

Theorem 27. *There is a linear-time algorithm that given an n-vertex triangle-free plane graph G with $\Omega(n)$ faces of size four outputs a reduction of G with at most $n - \Omega(n)$ vertices.*

Proof. For a triangle-free plane graph G, let $H(G)$ be the graph obtained from G by adding all edges joining all pairs of diagonally opposite vertices on the boundary of a face of length four. Let $\mathcal{G}$ be the class of graphs of the form $H(G)$ for some triangle-free plane graph G. The class $\mathcal{G}$ has bounded expansion by [25] since every graph from $\mathcal{G}$ can be drawn in the plane with at most one crossing on each edge.

By Theorem 4 there exists an integer K such that every graph $H = H(G) \in \mathcal{G}$ has a proper coloring ψ using at most K colors such that the union of every two color classes has tree-depth at most two (this type of coloring is known as a star coloring since every two color classes induce a star forest). Let $F := E(H) - E(G)$ be the set of new edges of H, and let $\xi : F \to \{1, 2, \ldots, \binom{K}{2}\}$ be defined by $\xi(e) := \{\psi(u), \psi(v)\}$, where u, v are the ends of e.

We apply Theorem 17 to structures arising from G as follows. There will be two binary relations: R_1 will describe the edges of G, and R_2, initialized to be empty, will describe which edges of $H \backslash E(G)$ have been contracted. By a *chain* in such a structure we mean a sequence $v_1, v_2, \ldots, v_k$ of vertices of H such that for all $i = 2, 3, \ldots, k$ either $R_1(v_{i-1}, v_i)$ or $R_2(v_{i-1}, v_i)$, and the former $(R_1(v_{i-1}, v_i))$ holds either for exactly one or exactly three indices $i \in \{2, 3, \ldots, k\}$. We say that the chain *joins* v_1 and v_k. We will apply Theorem 17 to the formula $\phi(x, y)$ that expresses the existence of a chain of length at most 11 joining x and y.

Choose i_0 to be the largest among the color classes of ξ. For each edge $e \in F$ with $\xi(e) = i_0$ such that the other edge in the 4-face of G, we do the following. We use Theorem 17 to determine, in constant time, whether the ends of e are joined by a chain of length at most 11. If they are not, then we add the ends of e to R_2; otherwise we do not. We proceed to the next edge with color i_0.

The choice of ξ implies that at all times during the execution of the algorithm there do not exist vertices u_1, u_2, u_3, u_4 of H such that $R_2(u_{i-1}, u_i)$ for all $i = 2, 3, 4$. Consequently, the chain test from the previous paragraph correctly determines whether identifying the ends of e creates a loop or a triangle in G.

After all edges e with $\xi(e) = i_0$ have been exhausted we identify all pairs of vertices u, v such that $R_2(u, v)$ holds, and those pairs of vertices u', v' such that G has a 4-face with vertices u, u', v, v', $\xi(uv) = i_0$ and $R_2(u, v)$ does not hold. The identification can be performed in linear time: we first compute sets of vertices that should be identified to the same vertex and then using the drawing of G identify the pairs and rename them.

Since the number of faces of size four is linear in n, the choice of i_0 implies that we have identified $\Omega(n)$ pairs of vertices and the order of G has shrunken by $\Omega(n)$. $\square$

References

[1] G. S. Brodal, R. Fagerberg: *Dynamic representations of sparse graphs*, in: Proc. WADS'99, LNCS vol. 1663, Springer, 1999, 342–351.

[2] B. Courcelle: The monadic second-order logic of graph I. Recognizable sets of finite graphs, Inform. and Comput. 85 (1990), 12–75.

[3] A. Dawar, M. Grohe, S. Kreutzer: *Locally excluding a minor*, in: Proc. LICS'07, IEEE Computer Society Press, 270–279.

[4] A. Dawar, S. Kreutzer: *Parameterized Complexity of First-Order Logic*, Electronic Colloquium on Computational Complexity, TR09-131 (2009).

[5] Z. Dvořák, K. Kawarabayashi, R. Thomas: *Three-coloring triangle-free planar graphs in linear time*, in: Proc. SODA'09, ACM&SIAM, 2009, 1176–1182.

[6] Z. Dvořák, D. Král': *Algorithms for classes of graphs with bounded expansion*, in: Proc. WG'09, LNCS vol. 5911, Springer, 2009, 17–32.

[7] Z. Dvořák, D. Král', R. Thomas: *Coloring triangle-free graphs on surfaces*, in: Proc. SODA'09, ACM&SIAM, 2009, 120–129.

[8] Z. Dvořák, D. Král', R. Thomas: *Three-coloring triangle-free graphs on surfaces VI. A linear-time algorithm*, in preparation.

[9] D. Eppstein: *Subgraph isomorphism in planar graphs and related problems*, in: Proc. SODA'95, ACM&SIAM, 632–640.

[10] D. Eppstein: *Subgraph isomorphism in planar graphs and related problems*, J. Graph Algorithms Appl. 3 (1999), 1–27.

[11] D. Eppstein: *Diameter and treewidth in minor-closed graph families*, Algorithmica 27 (2000), 275–291.

[12] M. Frick, M. Grohe: *Deciding first-order properties of locally tree-decomposable structures*, J. ACM 48 (2001), 1184–1206.

[13] H. Gaifman: *On local and non-local properties*, in: Proc. Herbrands Symp. Logic Coloq., North-Holland, 1982.

[14] M. Garey, D. Johnson, L. Stockmeyer: *Some simplified NP-complete graph problems*, Theoret. Comput. Sci. 1 (1976) 237–267.

[15] H. Grötzsch: *Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel*, Wiss. Z. Martin-Luther-Universität, Halle, Wittenberg, Math.-Nat. Reihe 8 (1959), 109–120.

[16] L. Kowalik and M. Kurowski: *Short path queries in planar graphs in constant time*, in: Proc. STOC'03, 143–148.

[17] L. Kowalik: *Fast 3-coloring triangle-free planar graphs*, in: Proc. ESA'04, LNCS vol. 3221, Springer, 2004, 436–447.

[18] L. Kowalik, M. Kurowski: *Oracles for bounded length shortest paths in planar graphs*, ACM Trans. Algorithms 2 (2006), 335–363.

[19] S. Kreutzer: *Algorithmic meta-theorems*, to appear in a workshop volume for a workshop held in Durham 2006 as part of the Newton institute special programme on Logic and Algorithms. An extended abstract appeared in: Proc. IWPEC'08, LNCS vol. 5018, Springer, 2008, 10–12.

[20] J. Nešetřil, P. Ossona de Mendez: *Linear time low tree-width partitions and algorithmic consequences*, in: Proc. STOC'06, 391–400.

[21] J. Nešetřil, P. Ossona de Mendez: *Grad and classes with bounded expansion I. Decompositions.*, Eur. J. Comb. 29 (2008), 760–776.

[22] J. Nešetřil, P. Ossona de Mendez: *Grad and classes with bounded expansion II. Algorithmic aspects.*, Eur. J. Comb. 29 (2008), 777–791.

[23] J. Nešetřil, P. Ossona de Mendez: *Grad and classes with bounded expansion III. Restricted graph homomorphism dualities.*, Eur. J. Comb. 29 (2008), 1012–1024.

[24] J. Nešetřil, P. Ossona de Mendéz: On nowhere dense graphs, manuscript, 2008.

[25] J. Nešetřil, P. Ossona de Mendez and D. Wood, *Characterisations and Examples of Graph Classes with Bounded Expansion*, preprint (arXiv:0902.3265v2).

[26] J. Nešetřil, P. Ossona de Mendez: *Structural properties of sparse graphs*, in: M. Grötschel, G. O. H. Katona (eds.): Building Bridges Between Mathematics and Computer Science, Bolyai Society Mathematical Studies vol. 19, Springer, 2008.

[27] D. Peleg: *Distance-dependent distributed directories*, Info. Computa. 103 (1993), 270–298.

[28] N. Roberson, P. D. Seymour: *Graph minors. XIII: the disjoint paths problem*, J. Combin. Theory Ser. B 63 (1995), 65–110.

[29] D. Wood: *On the maximum number of cliques in a graph*, Graphs Combin. 23 (2007), 337–352.