Parameterized circuit complexity of model checking first-order logic on sparse structures *

Michał Pilipczuk Sebastian Siebertz Szymon Toruńczyk

Institute of Informatics, University of Warsaw, Poland
{michal.pilipczuk,siebertz,szymtor}@mimuw.edu.pl

May 9, 2018

Abstract

We prove that for every class $\mathscr{C}$ of graphs with effectively bounded expansion, given a first-order sentence φ and an n-element structure $\mathbb{A}$ whose Gaifman graph belongs to $\mathscr{C}$, the question whether φ holds in $\mathbb{A}$ can be decided by a family of AC-circuits of size $f(\varphi) \cdot n^c$ and depth $f(\varphi) + c \log n$, where f is a computable function and c is a universal constant. This places the model-checking problem for classes of bounded expansion in the parameterized circuit complexity class para-AC^1. On the route to our result we prove that the basic decomposition toolbox for classes of bounded expansion, including orderings with bounded weak coloring numbers and low treedepth decompositions, can be computed in para-AC^1.

1 Introduction

Model-checking on sparse structures. We study the model-checking problem for first-order logic (FO): given a relational structure $\mathbb{A}$ and a first-order sentence φ over the vocabulary of $\mathbb{A}$, decide whether φ holds in $\mathbb{A}$. A naive algorithm for this problem recursively browses through all evaluations of each quantified variable and runs in time $n^{\mathcal{O}(|\varphi|)}$, where n is the size of the universe of $\mathbb{A}$. Thus, the running time is polynomial for every fixed φ, but the degree of the polynomial depends on φ. In the language of parameterized complexity, this means that the model-checking problem for first-order logic on arbitrary structures is in the complexity class XP when parameterized by the input formula φ. This is traditionally put in contrast to the class FPT (for *fixed-parameter tractable*) where we require the existence of an algorithm with running time $f(\varphi) \cdot n^c$ for a computable function f and universal constant c; thus, the degree of the polynomial factor has to be independent of the parameter. See [10, 13, 22] for an introduction to parameterized complexity.

*The work of M. Pilipczuk and S. Siebertz is supported by the National Science Centre of Poland via POLONEZ grant agreement UMO-2015/19/P/ST6/03998, which has received funding from the European Union's Horizon 2020 research and innovation programme (Marie Skłodowska-Curie grant agreement No. 665778). The work of Sz. Toruńczyk is supported by the National Science Centre of Poland grant 2016/21/D/ST6/01485.

In general structures we do not hope for an FPT algorithm for model-checking FO, because the problem is complete for the class AW[$\star$] (cf. [22]). Already the problem of deciding the existence of a clique of size k in a given graph of size n, which is easily expressible by a first-order formula with k existential quantifiers, is W[1] hard in general, so believed not to be FPT when parametrized by k, i.e., solvable by an algorithm with running time $f(k)\cdot n^c$ for some computable f and fixed c. However, it was realized that on *sparse* structures, i.e. those whose Gaifman graph is sparse, efficient parameterized algorithms for model-checking FO exist. Starting with the result of Seese [37], who gave an FPT algorithm on structures with universally bounded degree, a long line of research focused on showing fixed-parameterized tractability of the problem for more and more general classes of sparse structures: of bounded local treewidth [23] (this includes planar and bounded-genus structures), excluding a fixed minor [21], and locally excluding a minor [11].

This line of research naturally converged to studying abstract notions of sparsity: classes of *bounded expansion* and *nowhere dense* classes. These two concepts form foundations of a deep and rapidly developing theory of sparse graph classes, first introduced and pursued by Nešetřil and Ossona de Mendez [30, 31, 32, 33], which by now has found multiple applications in combinatorics, algorithm design, and logic. We refer the reader to the book of Nešetřil and Ossona de Mendez [34] for a comprehensive overview of the field as of 2012, and to the lecture notes of the first two authors for a compact and updated exposition of the basic toolbox [36].

Formally, a graph H is a *depth-r minor* of a graph G if H can be obtained from a subgraph of G by contracting mutually disjoint connected subgraphs of radius at most r. A class of graphs $\mathscr{C}$ has *bounded expansion* if there is a function $f\colon \mathbb{N}\to\mathbb{N}$ such that for every $r\in\mathbb{N}$, in every depth-r minor of a graph from $\mathscr{C}$ the ratio between the number of edges and the number of vertices is bounded by $f(r)$. More generally, $\mathscr{C}$ is *nowhere dense* if there is a function $t\colon \mathbb{N}\to\mathbb{N}$ such that no graph from $\mathscr{C}$ admits the clique $K_{t(r)}$ as a depth-r minor. Every class of bounded expansion is nowhere dense, but the converse does not necessarily hold [32]. Class $\mathscr{C}$ has *effectively bounded expansion*, respectively is *effectively nowhere dense*, if the respective function f or t as above is computable. These definitions are naturally generalized to classes of relational structures by considering the Gaifman graph of a structure.

Many classes of sparse graphs studied in the literature have (effectively) bounded expansion. These include: planar graphs, graphs of bounded maximum degree, graphs of bounded treewidth, and more generally, graphs excluding a fixed (topological) minor. A notable negative example is that classes with bounded *degeneracy*, equivalently with bounded *arboricity*, do not necessarily have bounded expansion, as there we have only a finite bound on the edge density in subgraphs (aka depth-0 minors).

By the result of Dvořák, Král', and Thomas [16], the FO model-checking problem on any class $\mathscr{C}$ of effectively bounded expansion admits a linear FPT algorithm, i.e. with running time $f(\varphi)\cdot n$ for computable f. Grohe, Kreutzer, and the second author [27] lifted this result to any effectively nowhere dense class $\mathscr{C}$; here, the dependence on the structure size n is *almost linear*, i.e. of the form $n^{1+\varepsilon}$ for any $\varepsilon>0$. As observed by Dvořák et al. [16], the result of Grohe et al. [27] is the final answer as long as subgraph-closed classes are concerned: on any subgraph-closed class $\mathscr{C}$ that is not nowhere dense, the FO model-checking problem is already AW[$\star$]-complete. Conceptually, this means that the notion of nowhere denseness exactly characterizes classes of inputs where sparsity-based arguments can lead to efficient parameterized algorithms for deciding first-order definable properties.

Parameterized circuit complexity. In this paper we take a different angle on the parameterized complexity of model-checking FO, namely that of *circuit complexity*. A fundamental fact from descriptive complexity is that FO is essentially equivalent to AC^0 (c.f. [28] for a precise statement). In particular, every fixed first-order expressible property can be checked by a family of AC-circuits of polynomial size and constant depth. More precisely, provided the property is expressed by a first-order sentence φ, the circuit for n-element inputs has size $n^{\mathcal{O}(|\varphi|)}$ and depth $\mathcal{O}(|\varphi|)$. Viewing this via the standard interpretation of circuits as an abstraction for parallel algorithms, this is a highly parallelizable algorithm performing total (sequential) XP work. Obviously, in general we cannot expect the problem to be solvable by circuits of FPT size (i.e., of size $f(\varphi) \cdot n^c$ for computable f and constant c), as evaluating such circuits would yield a sequential FPT algorithm, implying $\mathsf{FPT} = \mathsf{AW}[\star]$. However, the question for known classes for which FO model-checking is FPT persists: how, and in what sense, can we solve FO model-checking on these classes using circuits of FPT size and low depth? Viewing circuits again as a model for parallelization, this would correspond to a well-parallelizable FPT algorithm.

Curiously, even though the complexity-theoretical foundations of parameterized complexity are expressed using circuit complexity, the question of what are the appropriate analogues of standard circuit complexity classes in parameterized complexity was not systematically studied up to very recently, when Elberfeld et al. [20] and Bannach et al. [4] introduced an appropriate definitional layer and gave several foundational results. Slightly informally, a parameterized problem is in the class para-AC^i (where $i > 0$) if it can be solved by an (appropriately uniform) family of AC-circuits $(C_{n,k})_{n,k\in\mathbb{N}}$, where the circuit $C_{n,k}$ solves the problem on inputs of size n and parameter value k, such that each $C_{n,k}$ has size $f(k) \cdot n^c$ and depth $f(k) + c \cdot \log^i n$, for a computable function f and universal constant c. The classes para-NC^i are defined similarly using NC-circuits. The class para-AC^0 is defined slightly differently: we require the depth to be bounded by a universal constant, independent of the parameter. By allowing the depth to be bounded by a function of the parameter we obtain the larger class para-$\mathsf{AC}^{0\uparrow}$. We give the formal definitions and fix our notation in Section 2.

In [4] Bannach et al. showed how the technique of *color coding* can be implemented in para-AC^0, leading to a first batch of results for several parameterized problems. Later, Bannach and Tantau [5] showed that model-checking monadic second-order logic (MSO) can be done in para-$\mathsf{AC}^{0\uparrow}$ on structures of bounded treedepth and in para-$\mathsf{NC}^{2+\varepsilon}$ for any $\varepsilon > 0$ on structures of bounded treewidth; here, the parameter is both the formula and the treedepth, respectively the treewidth of the input structure. This is a parameterized circuit complexity analogue of the classic theorem of Courcelle stating that model-checking MSO is fixed-parameter tractable when parameterized by the formula and the treewidth of the input structure. Recent advances show descriptive relations between parameterized circuit complexity and fragments of FO with a bounded number of variables [9], and applications to kernelization [6].

In this light, it is natural to ask about the parameterized circuit complexity of model checking FO on sparse structures. Investigating this question is precisely the goal of this work. Our main result is encompassed by the following theorem.

Theorem 1. *Suppose $\mathscr{C}$ is a graph class with effectively bounded expansion and let Σ be a relational vocabulary of arity 2. Then the following problem parameterized by $\varphi \in \mathrm{FO}[\Sigma]$ is in* para-AC^1*: given a Σ-structure $\mathbb{A}$ whose Gaifman graph belongs to $\mathscr{C}$, determine whether $\mathbb{A} \models \varphi$.*

Unraveling the definitions, Theorem 1 states that model-checking FO on Σ-structures with Gaifman graphs belonging to $\mathscr{C}$ can be done using a family of AC circuits $(C_{n,\varphi})_{n\in\mathbb{N},\varphi\in\mathrm{FO}[\Sigma]}$, where $C_{n,\varphi}$ verifies the satisfaction of φ on structures with n elements, and each $C_{n,\varphi}$ has size $f(\varphi)\cdot n^c$ and depth $f(\varphi)+c\log n$, for a computable function f and universal constant c. Viewing circuits as an abstraction for parallel algorithms, this means that the problem can be solved in parallel time $f(\varphi)+c\log n$ and performing total work $f(\varphi)\cdot n^c$. Hence Theorem 1 can be regarded as a parallelized variant of the result of Dvořák et al. [16].

The assumption in Theorem 1 that Σ has arity 2 allows us to abstract away the question of how the input is represented, as we simply assume that each relation in $\mathbb{A}$ is encoded on input as a one- or two-dimensional boolean table. In the presence of higher-arity relations, the choice of an encoding could influence the statements about bounds on circuit sizes in a technical way. We prefer to avoid these issues and simply assume that there are no higher-arity relations.

Our techniques. We prove Theorem 1 by analyzing the existing approach to proving fixed-parameter tractability in the sequential case. Essentially, the idea is to first compute a suitable decomposition of the input structure, and then leverage this decomposition to give a quantifier elimination procedure for first-order logic. A suitable decomposition has the form of a *low treedepth coloring* that uses a bounded number of colors; it is known that such colorings exist for graphs from classes of bounded expansion [30]. Efficient algorithms for computing low treedepth colorings provided by Nešetřil and Ossona de Mendez [30] allowed Dvořák et al. [16] to give an efficient quantifier elimination procedure that reduces every first-order formula to a quantifier-free formula at the cost of extending the structure by adding new unary relations and unary functions, which however do not change the Gaifman graph. From this, an algorithm for model-checking follows. Later, Grohe and Kreutzer [25] gave a new presentation of the quantifier elimination procedure, which is conceptually quite different from the original argument of [16]. In particular it reduces every formula to an existential formula instead of quantifier-free one, but the extension of the structure does not use function symbols.

Our work toward the proof of Theorem 1 is divided into two parts. First, we prove that a low treedepth coloring of the graph can be computed in para-AC^1. Second, using this result we revisit the quantifier elimination procedure and show that it can be implemented in para-AC^1.

For computing a low treedepth coloring, on high level we follow the standard approach: first find a vertex ordering of the given graph with bounded *weak coloring number*, and then apply a coloring procedure on this vertex ordering to get a low treedepth coloring. Classic implementations of both these steps are sequential, however we show that both of them can be performed in para-AC^1. For the first step, the main idea is to construct the ordering by extracting the vertices not one by one, as a sequential algorithm would do, but in much larger chunks. Namely, we perform $\log n$ rounds where in each round at least half of the remaining vertices are extracted and ordered, which directly translates to a construction of a para-AC^1 circuit family. This comes at a price in the quality of the obtained ordering; in other words, we trade the approximation factor of the algorithm for its parallelization. For the second step, we follow the classic divide-and-conquer approach to parallel coloring graphs of bounded maximum degree. Then we extend this to graphs of bounded degeneracy by applying essentially the same technique of dividing the graph into $\log n$ parts, each inducing a graph of bounded maximum degree.

We remark that the approach explained above is heavily inspired by the existing body of work on *distributed algorithms* for sparse graphs. We relied on ideas from Barenboim and Elkin [8] who, among other results, gave an $\mathcal{O}(\log n)$-time distributed algorithm that, given a graph of degeneracy d, finds its proper coloring with $\mathcal{O}(d^2)$ colors[1]. In particular, Barenboim and Elkin showed how to compute an approximate degeneracy ordering of such a graph in $\mathcal{O}(\log n)$ communication rounds by extracting half of the remaining vertices in each round; our algorithm for this step is a circuit implementation of this procedure, lifted to the weak coloring number instead of degeneracy. Using the results of [8] and the approach via fraternal augmentations, Nešetřil and Ossona de Mendez [35] gave a logarithmic-time distributed algorithm that, given a graph from a fixed class of bounded expansion, computes its treedepth-p coloring using a constant number of colors, for any constant p.

For the quantifier elimination procedure, we essentially revisit the existing approach and show that it can be implemented in para-AC^1. This requires technical attention in several places, but conceptually there is no new ingredient. Our argument is roughly based on the exposition of Grohe and Kreutzer [25]. However, we obtain a stronger final form, similar to that of Dvořák et al. [16], replace the usage of FO types with an explicit combinatorial argument in the spirit of marking witnesses as in [16], and streamline the presentation.

Additional results. We believe that our approach to the proof of Theorem 1 has an additional benefit in that we implement most of the basic algorithmic toolbox for classes of bounded expansion in para-AC^1. This can be re-used for problems other than model-checking first-order logic.

For instance, consider the problem of computing the smallest *distance-r dominating set* in a given graph G, which is a subset of vertices D such that every vertex of G is at distance at most r from some vertex of D. This problem often serves as a benchmark for sparsity methods, and it was considered in the theory of sparse graphs from the points of view of parameterized algorithms [12], approximation [1, 18], and kernelization [14, 19, 29]. In particular, Amiri et al. [1] have recently used the results of Nešetřil and Ossona de Mendez [35] to give a distributed logarithmic-time constant-factor approximation algorithm for the distance-r dominating set problem on any class of bounded expansion. By combining their ideas with our constructions of orderings with bounded weak coloring numbers, we immediately obtain the following approximation result.

Theorem 2. *Suppose $\mathscr{C}$ is a graph class with effectively bounded expansion. Then there exists a computable function $\alpha\colon \mathbb{N} \to \mathbb{N}$ such that the following problem parameterized by $r \in \mathbb{N}$ is in* para-AC^1*: given a graph $G \in \mathscr{C}$, compute a distance-r dominating set in G of size at most $\alpha(r) \cdot \mathrm{dom}_r(G)$. Here, $\mathrm{dom}_r(G)$ denotes the size of a smallest distance-r dominating set in G.*

Organization. In Section 2 we establish notation and recall known results. In Section 3 we show how to compute vertex orderings with low weak coloring numbers, while in Section 4 we show how to construct low treedepth colorings. Section 5 contains the proof of the main result, Theorem 1. We conclude in Section 6 with some final remarks and open problems.

[1] Barenboim and Elkin use the parameter *arboricity* which differs from degeneracy by multiplicative factor at most 2.

2 Preliminaries

All graphs considered in this paper are simple, i.e. do not contain self-loops or multiple edges connecting the same pair of vertices. We use standard graph notation, see e.g. [10].

Classes of bounded expansion. As mentioned in the introduction, a graph H is a *depth-r minor* of a graph G if H can be obtained from a subgraph of G by contracting mutually disjoint connected subgraphs of radius at most r. Graph H is a *depth-r topological minor* of G if there is a subgraph of G that is an $\leqslant 2r$-*subdivision* of H, that is, can be obtained from H by replacing each edge by a path of length at most $2r+1$. It is easy to see that if H is a depth-r topological minor of G, then H is also a depth-r minor of G.

A graph H is a *depth-r minor* of G if one can find a *depth-r minor model* $(I_u)_{u\in V(H)}$ of H in G, where I_u for $u \in V(H)$ are pairwise vertex-disjoint connected subgraphs of G of radius at most r such that for each $uv \in E(H)$ there is an edge between a vertex of I_u and a vertex of I_v. In other words, in the standard definition of a minor model we restrict the *branch sets* I_u to have radius at most r. Similarly, H is a *depth-r topological minor* of G if there exists a *depth-r topological minor model* μ of H in G: such μ is a mapping that sends vertices of H to pairwise different vertices of G and edges of H to paths of length at most $2r+1$ in G such that $\mu(uv)$ has endpoints $\mu(u)$ and $\mu(v)$ for each $uv \in E(H)$, and paths in $\mu(E(H))$ pairwise may share only the endpoints.

The *edge density* of a graph G is the ratio between the number of edges and the number of vertices in G, i.e., $\frac{|E(G)|}{|V(G)|}$. For a graph G, by $\nabla_r(G)$ we denote the maximum over all depth-r minors H of G of the edge density of H. Similarly, $\widetilde{\nabla}_r(G)$ denotes the maximum edge density in a depth-r topological minor of G. As a depth-r topological minor is also a depth-r minor, we have $\widetilde{\nabla}_r(G) \leqslant \nabla_r(G)$. However, it is known that $\nabla_r(G)$ is also bounded from above by a computable function of r and $\widetilde{\nabla}_r(G)$ [17]. For a graph class $\mathscr{C}$, we denote $\nabla_r(\mathscr{C}) = \sup_{G\in\mathscr{C}} \nabla_r(G)$. and $\widetilde{\nabla}_r(\mathscr{C}) = \sup_{G\in\mathscr{C}} \widetilde{\nabla}_r(G)$. A graph class $\mathscr{C}$ has *bounded expansion* if $\nabla_r(\mathscr{C})$ is finite for all $r \in \mathbb{N}$, equivalently if $\widetilde{\nabla}_r(\mathscr{C})$ is finite for all $r \in \mathbb{N}$. It has *effectively bounded expansion* if there is a computable (from r) upper bound on $\nabla_r(\mathscr{C})$, equivalently on $\widetilde{\nabla}_r(\mathscr{C})$.

Parameterized circuit complexity. We explain the definitional layer of parameterized circuit complexity basing our notation on Bannach et al. [4], though prior foundational work on parameterized circuit complexity was done by Elberfeld et al. [20]. An AC-*circuit* C is a directed acyclic graph with node set consisting of input, conjunction (AND), disjunction (OR), and negation (NOT) gates. There are no restrictions on the fan-in or fan-out of the gates. One or more sources of C are designated as the output gates, and both input and output gates of C are ordered. If $(u_1, \ldots, u_n)$ and $(v_1, \ldots, v_m)$ are the input and output gates of C, respectively, then C evaluates a function from $\{0,1\}^n$ to $\{0,1\}^m$ defined as follows: given input $x = (x_1, \ldots, x_n)$, set the value of each input gate u_i to x_i, evaluate the gates of the circuit in a bottom-up manner naturally, and define the output y to be the sequence of values computed in gates $(v_1, \ldots, v_m)$. The *depth* of a circuit is the length of a longest path from an input gate to an output gate. The *size* of a circuit is the number of its gates.

A *parameterized transformation* is a function $F\colon \{0,1\}^\star \to \{0,1\}^\star$ together with a polynomial-time computable function $\kappa\colon \{0,1\}^\star \to \mathbb{I}$, called *parameterization*. Here $\mathbb{I}$ is some indexing set for parameters and we assume that its elements can be encoded as binary strings. Typically in parameterized complexity we have $\mathbb{I} = \mathbb{N}$ — the parameter is just an integer — but it will be convenient to assume larger generality, as some our circuit families will be indexed by first-order sentences on graphs or tuples of integers. A *parameterized problem* is just a parameterized transformation with the output always belonging to $\{0,1\}$, for false and true, respectively.

A parameterized transformation is in the class FPT if there is an algorithm that computes it in time $f(k) \cdot n^c$ on inputs of size n and parameter value k, where f is a computable function and c is a universal constant. It is in class linFPT if moreover $c = 1$ and $\kappa(\cdot)$ is linear-time computable.

For an indexing set $\mathbb{I}$, we may consider a family $(C_{n,k})_{n\in\mathbb{N},k\in\mathbb{I}}$ of AC-circuits, where each $C_{n,k}$ has exactly n inputs. We say that such a family is $\mathsf{dlogtime}$-*uniform* if there exists an algorithm that given $n \in \mathbb{N}$, $k \in \mathbb{I}$, and $i \in \mathbb{N}$, all encoded in binary, computes the i-th bit of the encoding of $C_{n,k}$ in time $f(k) + \mathcal{O}(\log i + \log n)$, for some computable function f. All circuit families in this paper are $\mathsf{dlogtime}$-uniform; this will always follow from the construction in a straightforward manner, so we refrain from providing technical details in order not to obfuscate the main ideas.

For $i > 0$, we say that a parameterized transformation (F, κ) is in para-AC^i if there exists a $\mathsf{dlogtime}$-uniform family of circuits $(C_{n,k})_{n\in\mathbb{N},k\in\mathbb{I}}$ such that

- $C_{n,k}$ has size $f(k)\cdot n^{\mathcal{O}(1)}$ and depth $f(k)+\mathcal{O}(\log^i n)$ for some computable function $f\colon \mathbb{I} \to \mathbb{N}$;
- for each $x \in \{0,1\}^\star$, the output of $C_{|x|,\kappa(x)}$ applied to x is $F(x)$.

Note that the above definition implicitly assumes that for all x, x' with $|x| = |x'|$ and $\kappa(x) = \kappa(x')$, the outputs $F(x)$ and $F(x')$ have the same length, as this length must be equal to the number of outputs of $C_{|x|,\kappa(x)}$. Bannach et al. [4] also define a larger class para-$\mathsf{AC}^{i\uparrow}$ by relaxing the restriction on the depth from $f(k) + \mathcal{O}(\log^i n)$ to $f(k) \cdot \log^i n$, for a computable function f. It is easy to see that para-$\mathsf{AC}^i \subseteq$ para-$\mathsf{AC}^{i\uparrow} \subseteq$ para-$\mathsf{AC}^{i+\varepsilon}$ for any $\varepsilon > 0$. In fact, in this paper we would be able to simplify some arguments if we only wanted to prove containment in para-$\mathsf{AC}^{1\uparrow}$.

For $i = 0$, the classes para-AC^0 and para-$\mathsf{AC}^{0\uparrow}$ are defined slightly differently: in para-AC^0 we require that the depth of the circuits is $\mathcal{O}(1)$, i.e. bounded by a universal constant independent of the parameter, while in para-$\mathsf{AC}^{0\uparrow}$ we allow the depth of $C_{n,k}$ to be bounded by $f(k)$, for a computable function f.

Let us briefly elaborate on the differences between the classes para-AC^i and para-$\mathsf{AC}^{i\uparrow}$. Both definitions are natural candidates for what a parameterized analogue of AC^i should be, as in both cases the class becomes AC^i whenever k is fixed to be a constant. However, bounding the depth by $f(k) + c \cdot \log n^i$ instead of $f(k) \cdot \log n^i$ gives better guarantees when transforming the circuit to a formula (a circuit with maximum fan-out 1). For instance, every para-NC^1 circuit (where we restrict fan-in to be at most 2) can be unravelled to an equivalent para-NC^1 formula, which is the analogue of a well-known property of NC^1, but this is no longer the case for para-$\mathsf{NC}^{1\uparrow}$, because the formula size would be $n^{f(k)}$ instead of $f(k) \cdot n^{\mathcal{O}(1)}$. Similarly, every para-$\mathsf{AC}^0$ circuit can be unravelled to an equivalent para-AC^0 formula, but this property is not shared by para-$\mathsf{AC}^{0\uparrow}$. Nevertheless, classes para-$\mathsf{AC}^{i\uparrow}$ are still worth studying due to encompassing many natural algorithms. This state reflects the situation in parameterized analogues of nondeterministic

logspace, where the difference between bounding the space by $f(k) + c \log n$ and by $f(k) \cdot \log n$ has dramatic implications for determinization results. We refer to the work of Elberfeld et al. [20] for a broader exposition of these connections.

Graph problems. Typically, throughout this paper the input to a parameterized transformation will be a graph G on n vertices. In this case we will always assume that the input is encoded as the $n \times n$ binary adjacency matrix of the graph; thus the circuit computing the transformation needs to have n^2 inputs. Abusing the above notation somewhat, for parameterized circuit classes, we will interpret the $|x|$ for an encoding x of G as the number n of vertices of G, instead of the actual length n^2 of x. Thus, the domain of a parameterized transformation defined on graphs consists of all words of length n^2 for some integer n, interpreted as adjacency matrices, and each circuit $C_{n,k}$ will actually have n^2 inputs. In case the input to the transformation consists of a graph together with some additional piece of information (e.g. additional numerical parameters, a coloring of the graph, or an ordering of its vertices), the appropriate encoding of this additional information (the form of which will be specified later) is provided via extra input gates. In all cases, the circuit $C_{n,k}$ will be responsible for the treatment of instances with n vertices and parameter value k.

In case of problems parameterized by additional numerical parameters (e.g. "given a graph G with maximum degree at most d, where d is the parameter"), we assume that the numerical parameters are appended to the input and the parameterization function κ just extracts this part of the input and presents it as the parameter. Also, we do not assume that a circuit needs to check whether the input satisfies the stated constraint. For instance, if we say that a circuit computes some output given a graph whose treedepth is at most h, then we only state that the output is computed correctly provided the input graph has treedepth at most h, while we do not assert anything about the output of the circuit on graphs of higher treedepth.

Basic circuit constructions. One of the fundamental results for parameterized circuit complexity is that counting up to a threshold parameter d can be done in para-AC^0. More precisely, consider the problem Threshold parameterized by d: given a word $x \in \{0,1\}^\star$, determine whether x has at most d ones. A naive construction of a constant-depth circuit would be to check every d-tuple of inputs, but this would result in a circuit of size $\Omega(n^d)$. However, Bannach et al. [4] showed that Threshold in fact is in para-AC^0 using a parallel implementation of color coding.

Theorem 3 (Lemma 3.3 of [4]). Threshold *is in* para-AC^0.

Note that by our definition of para-AC^0, the circuits in the family provided by Theorem 3 have depth bounded by a universal constant, independent of d. From Theorem 3 we can immediately derive the following corollary; henceforth, a subset of vertices of an n-vertex graph will be encoded in our circuits as its characteristic (binary) vector using n gates.

Corollary 1. *The following transformation parameterized by d is in* para-AC^0*: given a graph G, compute the set of vertices of G that have degree at most d.*

Proof. For every vertex u of G, apply the circuit given by Theorem 3 to the row of the adjacency matrix of G corresponding to u. □

Another primitive in our algorithms will be counting distances in graphs up to a fixed threshold $r \in \mathbb{N}$. This is encapsulated in the following lemma.

Lemma 2. *There exists a* dlogtime*-uniform family of* AC*-circuits* $(D_{n,r})_{n,r\in\mathbb{N}}$ *such that each* $D_{n,r}$*, given a* n*-vertex graph* G*, outputs the* $n \times n$ *boolean matrix encoding, for each pair of vertices* u, v *of* G*, whether the distance between* u *and* v *in* G *is at most* r*. Each* $D_{n,r}$ *has size* $n^{\mathcal{O}(1)}$ *and depth* $\mathcal{O}(\log r)$*.*

Proof. Let A be the adjacency matrix of G with ones added on the diagonal. Then it suffices to compute A^r, the rth boolean power of A, that is, its rth power in the $(\mathrm{OR}, \mathrm{AND})$-semiring over $\{0, 1\}$. Observe that the distance between u and v in G is at most r if and only if the entry of A^r in the intersection of the u-column and v-row is equal to 1. Observe that A^r can be computed from A by a circuit of size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log r)$ using the iterative squaring algorithm. □

3 Bounded degeneracy

In this section we study the case of graphs of degeneracy d. Those are graphs which can be linearly ordered in such a way that every vertex has at most d smaller neighbors. It is well known that a class $\mathscr{C}$ of graphs has degeneracy bounded by some constant d if and only if there is a number c such that in every subgraph H of a graph $G \in \mathscr{C}$, the ratio between the number of edges and number of vertices in H is bounded by c. Therefore, bounded degeneracy is similar to bounded expansion, but we only bound the density of depth-0 minors, i.e. subgraphs. Many proof techniques concerning bounded expansion classes stem from the techniques for classes of bounded degeneracy. This is no different in our paper. In this section, we prove some parallelized variants of known results for classes of bounded degeneracy. Specifically, it is known that graphs of degeneracy d admit a proper coloring using $d + 1$ colors, and in Lemma 7, we show that a coloring using $\mathcal{O}(d^2)$ colors can be computed in para-AC^1. In the following section, we extend this result to classes of bounded expansion.

Definition and basic properties. A *vertex ordering* of a graph G is any ordering $\sigma = (v_1, \ldots, v_n)$ of the vertices of G. A vertex ordering σ can be also understood via the linear order $\leqslant_\sigma$ on $V(G)$ imposed by it: $u \leqslant_\sigma v$ iff $u = v$ or u appears earlier than v in σ. Whenever a vertex ordering of an n-vertex graph is represented in a circuit construction, we assume that it is represented as the $n \times n$ boolean matrix encoding the order $\leqslant_\sigma$. The *degeneracy* of the ordering σ is the least d such that every vertex $v \in V(G)$ has at most d neighbors u satisfying $u <_\sigma v$. The *degeneracy* of a graph is the smallest possible degeneracy of its vertex orderings.

The following basic lemma is well-known. In fact, the property expressed in it is commonly used as the base definition of degeneracy.

Proposition 3. *The degeneracy of a graph* G *is equal to the smallest integer* d *such that every subgraph of* G *contains a vertex of degree at most* d*.*

By Proposition 3 it is clear that degeneracy is a monotone graph parameter, i.e., the degeneracy of a subgraph of a graph G is never larger than the degeneracy of G. Further, it also yields a greedy polynomial-time algorithm for computing the degeneracy of a given graph G: starting

with the whole graph G, repeatedly remove a vertex of the smallest degree from the current graph, until there are no more vertices left. The reversal of the order of removing the vertices is then a vertex ordering of G with optimum degeneracy. To see this, observe that suppose d is the largest degree of a removed vertex encountered during the procedure. On one hand, clearly the obtained vertex ordering has degeneracy d. On the other hand, at the moment when we removed a vertex of degree d, the currently considered subgraph of G had minimum degree d, which certifies that the degeneracy of G cannot be smaller than d by Proposition 3.

We use the following fact that a graph of degeneracy d has a linear number of edges, and moreover there are few vertices with degrees significantly larger than d.

Proposition 4. *An n-vertex graph G of degeneracy at most d has at most dn edges. Moreover, for every real $c \geqslant 1$, G has less than $\frac{n}{c}$ vertices of degree larger than $2cd$.*

Proof. For the first assertion, consider a vertex ordering σ of G of degeneracy at most d and count the edges by their higher (in σ) endpoints. For the second assertion, observe that otherwise by the hand-shaking lemma G would have more than $\frac{1}{2} \cdot \frac{n}{c} \cdot 2cd = dn$ edges, a contradiction with the first assertion. □

Finally, we recall the well-known fact that a graph of degeneracy d admits a proper coloring with $d+1$ colors. Recall here that a proper coloring of a graph is a coloring of its vertices such that no edge has both endpoints of the same color. Equivalently, every color class is an independent set.

Proposition 5. *A graph of degeneracy d admits a proper coloring with $d+1$ colors.*

Proof. Let G be the graph in question and let σ be a vertex ordering of G of degeneracy d. Consider the following greedy procedure that colors vertices of G with colors $\{1, \ldots, d+1\}$: iterate through vertices of G in the order of σ and for each vertex u assign to it any color that is not present among the neighbors of u smaller in σ. Since there are at most d such neighbors, such a color will always exist. □

Our goal in this section is to prove a parallelized variant of Proposition 5. This will be achieved in Lemma 7 below.

3.1 Block vertex orderings and computational aspects

We will use a relaxed variant of degeneracy orderings where vertices come in ordered blocks and every vertex has few neighbors in its own and smaller blocks.

Definition 1. A *block vertex ordering* of a graph G is an ordered partition $\tau = (B_1, B_2, \ldots, B_\ell)$ of the vertex set of G; its *length* is the number of blocks ℓ. The *degeneracy* of τ is the least integer d such that for each $i \in \{1, \ldots, \ell\}$, every vertex $v \in B_i$ has at most d neighbors in $\bigcup_{j=1}^{i} B_j$.

A block vertex ordering τ as above naturally imposes a total quasi-order $\leqslant_\tau$ on the vertex set of G: $u \leqslant_\tau v$ iff $u \in B_i$ and $v \in B_j$ with $i \leqslant j$. In our circuits we will assume that a block vertex ordering of an n-vertex graph is represented by the $n \times n$ boolean matrix encoding $\leqslant_\tau$.

Obviously, if $\sigma = (v_1, \ldots, v_n)$ is a vertex ordering of G, then the degeneracy of σ is equal to the degeneracy of the block vertex ordering $(\{v_1\}, \ldots, \{v_n\})$. On the other hand, if $\tau = (B_1, \ldots, B_\ell)$ is a block vertex ordering of G of degeneracy d, then by ordering each block arbitrarily and concatenating these orderings we obtain a vertex ordering of G of degeneracy at most d.

It will be important however that provided G has bounded degeneracy, we may find a block vertex ordering of small degeneracy that is of logarithmic length. This idea is also the cornerstone of the work of Barenboim and Elkin [8], who gave logarithmic-time distributed algorithms to approximately color graphs of bounded degeneracy. Our block vertex orderings correspond to *H-partitions* in their nomenclature, and similarly to us they show that an H-partition of small degeneracy and logarithmic size can be efficiently computed by repeatedly taking vertices of small degree. Actually, Barenboim and Elkin attribute the idea to an earlier work of Arikati et al. [2] that used the PRAM model of parallel algorithms.

Lemma 6. *The following transformation parameterized by d is in* para-AC^1*: Given an n-vertex graph G of degeneracy at most d, compute a block vertex ordering of G of degeneracy at most $4d$ and length at most $\log n$.*

Proof. We describe combinatorially how the block vertex ordering is constructed. Starting with $G_0 = G$, define the last block to consist of all vertices of G_0 that have degree at most $4d$ in G_0; by Proposition 4 with $c = 2$, this block constitutes more than half of the vertex set of G_0. Remove the block from G_0 yielding a graph G_1 and apply again the same procedure to G_1. That is, define the second-to-last block to consist of all vertices of G_1 that have degree at most $4d$ in G_1 and remove this block yielding G_2; since G_1 is a subgraph of G, again Proposition 4 ensures us that in this manner we remove more than half of the remaining vertex set. Thus the construction finishes with an empty graph after at most $\log n$ iterations and yielding at most $\log n$ blocks in total. It is straightforward to see that the degeneracy of the obtained block vertex ordering is at most $4d$.

We are left with implementing the above procedure using an AC-circuit family with prescribed size and depth constraints. We may perform exactly $\lfloor \log n \rfloor$ iterations, where after i iterations the last i blocks are defined; in case the whole vertex set has already been exhausted, the next iterations are idle. It is straightforward to implement each iteration by an AC-circuit of size $f(k) \cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(1)$ using Corollary 1, so by performing the iterations sequentially we obtain an AC-circuit of size $f(k) \cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log n)$. □

By ordering arbitrarily each block of the block vertex ordering given by Lemma 6 we obtain the following corollary: given a graph G of degeneracy at most d, computing a vertex ordering of G of degeneracy at most $4d$ can be done in para-AC^1. In other words, this is a 4-approximation algorithm for degeneracy in para-AC^1, where the target degeneracy is the parameter.

In fact, if in the proof of Lemma 6 we replaced $c = 2$ with $c = 1 + \varepsilon/2$ for any $\varepsilon > 0$, we would still have that the procedure performs $\mathcal{O}(\log n)$ iterations while producing a vertex ordering of degeneracy at most $2 + \varepsilon$, so this constitutes a $(2 + \varepsilon)$-approximation in para-AC^1.

It is natural to ask whether the following problem of determining degeneracy exactly is also in para-AC^1: for a parameter d, determine whether the degeneracy of a given graph is at most d. Recall here that this problem can be solved in polynomial time. We give a negative answer to this side question by proving the following theorem.

Theorem 4. *The following problem is* P*-hard under logspace reductions: Given a graph G, determine whether the degeneracy of G is at most* 2*.*

Thus, if determining degeneracy exactly was in para-AC^1, or even in para-AC^i for any i, then $\mathsf{NC} = \mathsf{P}$. Moreover, the same can be concluded about approximating degeneracy up to any factor $\alpha < \frac{3}{2}$. Since Theorem 4 is not directly relevant for our main result, but we find it interesting on its own, we include a proof in Appendix A.

3.2 Coloring graphs of bounded degeneracy

Recall from Proposition 5 that graphs of bounded degeneracy can be colored using a bounded number of colors. In this section, we show the following, parallelized variant of this result.

Lemma 7. *The following transformation parameterized by d is in* para-AC^1*: Given a graph of degeneracy at most d, compute its proper coloring with $(4d+1)^2$ colors.*

We outline the proof below. Recall that a similar statement in the context of distributed computing was obtained by Barenboim and Elkin [8]. The rest of Section 3 is devoted to outlining a proof of Lemma 7.

We shall first present how to greedily color bounded degree graphs in para-AC^1, and then leverage this understanding to color graphs of bounded degeneracy in para-AC^1. But before this, we present a key technical lemma that will be used multiple times. Essentially it says that provided we have already achieved some proper coloring with h colors, we may then use it to compute a better coloring using a circuit of depth linear in h. This trick was also used by Barenboim and Elkin [8].

Lemma 8. *There exists a* dlogtime*-uniform family of* AC*-circuits $(K_{n,d,h})_{n,d,h\in\mathbb{N}}$ such that each $K_{n,d,h}$, given a n-vertex graph G together with its block vertex ordering $\tau = (B_1, \ldots, B_\ell)$ with $\ell \leqslant h$ and of degeneracy d with each block B_i being an independent set in G, computes a proper coloring of G with $d+1$ colors. Each $K_{n,d,h}$ has size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(h)$.*

Proof. We build a coloring $\lambda\colon V(G) \to \{1, \ldots, d+1\}$ in h rounds, where in round i all vertices from block B_i receive their colors in λ. In round i every vertex $u \in B_i$ inspects all its neighbors and sets its own color to be the smallest color that is not present among its neighbors residing in lower blocks. Note that such a color always exists since the number of neighbors is at most d, by the assumption about the degeneracy of the input block vertex ordering, and there are $d+1$ colors available. To see that λ constructed in this way will be a proper coloring of G, observe that every edge uv of G connects two vertices from different blocks, say $u \in B_i$ and $v \in B_j$ for $i < j$. Hence v will pick its color in λ to be different than $\lambda(u)$.

We implement the above procedure by an AC-circuit of polynomial size and depth $\mathcal{O}(h)$ in a natural way: the circuit consists of h layers, where the ith layer corresponds to the ith iteration. Thus, it suffices to implement the assignment of color to every vertex $u \in B_i$ using an AC-circuit of polynomial size and constant depth. To this end, for every color $j \in \{1, \ldots, d+1\}$ we create a circuit of constant depth that computes whether j is present among neighbors of u from lower blocks ($B_{i'}$ for $i' < i$). ; this boils down to taking a disjunction over all vertices v of the conjunction of the fact that $v <_\tau u$, v is a neighbor of u, and v has already received color j.

Then, the smallest color that is not present among neighbors of u from lower blocks may be chosen as one that is not present, but all smaller ones are present; this requires one additional level in the circuit. □

Corollary 9. *There is a* dlogtime*-uniform family of* AC*-circuits* $(L_{n,d,h})_{n,d,h\in\mathbb{N}}$ *such that each* $L_{n,d,h}$*, given a* n*-vertex graph* G *of maximum degree at most* d *together with its proper coloring with* h *colors, computes a proper coloring of* G *with* $d+1$ *colors. Each* $L_{n,d,h}$ *has size* $n^{\mathcal{O}(1)}$ *and depth* $\mathcal{O}(h)$*.*

Proof. If $\lambda\colon V(G) \to \{1,\ldots,h\}$ is the input proper coloring, then $(\lambda^{-1}(1),\ldots,\lambda^{-1}(h))$ is a block vertex ordering of G of degeneracy at most d where each block is an independent set in G. Hence we may just compute this block vertex ordering using an AC-circuit of depth $\mathcal{O}(1)$ and apply the circuit $K_{n,d,h}$ provided by Lemma 8. □

Our first step towards the proof of Lemma 7 is the treatment of graphs of bounded degree. A graph of maximum degree at most Δ can be greedily colored with $\Delta+1$ colors. A naive implementation of this greedy procedure is sequential, but we will show now how to perform this task in para-AC^1. We remark that finding optimum or near-optimum proper colorings of graphs of bounded maximum degree is a very classic topic in distributed computing with a vast existing literature. We refer to the work of Barenboim [7] for the currently fastest algorithms and an excellent overview of the area.

We first show a weaker result, namely that a proper coloring with $\Delta+1$ colors of a graph of maximum degree at most Δ can be computed in para-$\mathsf{AC}^{1\uparrow}$, when parameterized by Δ. Recall that this means that we allow depth $f(\Delta)\cdot\log n$ instead of $f(\Delta)+\mathcal{O}(\log n)$. This can be done using a simple Divide&Conquer trick, which dates back to a classic $\mathcal{O}(\Delta\log n)$-time distributed algorithm for this problem of Goldberg et al. [24] (see also [3]).

Lemma 10. *The following transformation parameterized by* Δ *is in* para-$\mathsf{AC}^{1\uparrow}$*: Given a graph of maximum degree at most* Δ*, compute its proper coloring with* $\Delta+1$ *colors.*

Proof. We perform a divide-and-conquer algorithm. Given the input graph G with n vertices, arbitrarily partition its vertex set into two subset V_1 and V_2, each of size at most $\lceil n/2\rceil$. Let G_1 and G_2 be the subgraphs induced by V_1 and V_2 in G, respectively. Each of G_1, G_2 has maximum degree at most Δ, hence we can apply the algorithm recursively to both these graphs, yielding proper colorings λ_1,λ_2 of G_1 and G_2, respectively, each using $\Delta+1$ colors. By taking the union of these two colorings, where colors from different subgraphs are considered different, we obtain a proper coloring of G with $2\Delta+2$ colors. We may now apply the circuit given by Corollary 9 for $h=2\Delta+2$ to compute a coloring of G with $\Delta+1$ colors.

To turn the above algorithm into a circuit, observe that the depth of the recursion is $\mathcal{O}(\log n)$ and computation on each level requires a circuit of polynomial size and depth $\mathcal{O}(\Delta)$, by Corollary 9. Hence, overall the size of the circuit is polynomial and its depth is $\mathcal{O}(\Delta\log n)$. □

We now show containment in para-AC^1.

Lemma 11. *The following transformation parameterized by* Δ *is in* para-AC^1*: Given a graph of maximum degree at most* Δ*, compute its proper coloring with* $\Delta+1$ *colors.*

Proof. Let every vertex u of G arbitrarily (say, according to the order of inputs) put numbers $1, \ldots, \deg(u)$ on edges incident to it; thus every edge is labelled with two numbers, one originating from each endpoint. Such labeling can be computed by a circuit of size $f(\Delta) \cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(1)$ using the circuits provided by Theorem 3 to count, for every neighbor v of u, the number of neighbors of u with smaller indices than v.

For every pair of indices (i, j) with $1 \leqslant i \leqslant j \leqslant \Delta$, let $G_{i,j}$ be the subgraph of G with $V(G_{i,j}) = V(G)$ and $E(G_{i,j})$ consisting of those edges e of G, for which one endpoint of e labelled e with i, and the second labelled it with j. Observe that the maximum degree of $G_{i,j}$ is at most 2, since every vertex u of G can be adjacent to at most two edges of $G_{i,j}$: the one it labelled with i and the one it labelled with j. Using Lemma 10 we can compute, for each $1 \leqslant i \leqslant j \leqslant \Delta$, a proper 3-coloring $\lambda_{i,j}$ of $G_{i,j}$ using an AC-circuit of size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log n)$. Next we construct a product coloring λ of G with $3^{\binom{\Delta+1}{2}}$ colors: a color of a vertex u in λ is the $\binom{\Delta+1}{2}$-tuple of colors of u in the colorings $\lambda_{i,j}$ for all $1 \leqslant i \leqslant j \leqslant \Delta$. Since each edge of G participates in exactly one of subgraphs $G_{i,j}$, it is clear that λ is a proper coloring of G. We may finally apply Corollary 9 for $h = 3^{\binom{\Delta+1}{2}}$ to compute a proper coloring of G with $\Delta + 1$ colors using an AC-circuit of size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(3^{\binom{\Delta+1}{2}})$. Thus, in total we have constructed a circuit of size $f(\Delta) \cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(3^{\binom{\Delta+1}{2}} + \log n)$. □

We finally have all the tools to prove Lemma 7.

Proof (of Lemma 7). By Lemma 6 we may compute a block vertex ordering τ of G of degeneracy at most $4d$ and length at most $\log n$ in para-AC^1. Partition the edges of G into two graphs G_1, G_2 on the same vertex set as G: the edge set of G_1 consists of all edges whose endpoints lie in the same block of τ, while the edge set of G_2 consists of all edges whose endpoints lie in different blocks of τ. Observe that G_1 is a graph of maximum degree at most $4d$, hence we may apply Lemma 11 to compute its proper coloring λ_1 with $4d + 1$ colors in para-AC^1. On the other hand, τ is a block vertex ordering of G_2 with degeneracy at most $4d$, length at most $\log n$, and every block being an independent set in G_2. Hence, we may apply Lemma 8 to compute a proper coloring λ_2 of G_2 with $4d + 1$ colors using a circuit of polynomial size and depth $\mathcal{O}(\log n)$. Finally, let λ be the product coloring of λ_1 and λ_2: the color a vertex u receives in λ is the pair of colors it received in λ_1 and λ_2. Since each edge of G participates either in G_1 or in G_2, λ constructed in this manner is a proper coloring of G with $(4d + 1)^2$ colors. The fact that the constructed circuit satisfies the required size and depth bounds follows directly from the construction and from the bounds provided by Lemma 6, Lemma 8, and Lemma 11. □

4 Computing low treedepth colorings

As discussed in the previous section, a graph of bounded degeneracy admits a proper coloring using a bounded number of colors. There is a generalization of this result to graphs of bounded expansion, in terms of *low treedepth colorings*. Intuitively, for a fixed $p \in \mathbb{N}$, such a coloring is a coloring using a bounded number of colors, such that any p color classes induce a graph which has a depth-first search forest of bounded depth. Such colorings turn out to be very useful for many algorithmic purposes, among others, for model-checking. In this section, we generalize the result from the previous section, and show that such colorings can be computed in para-AC^1. As

previously, the such colorings are obtained by first finding an appropriate ordering of the graph, related to the notion of *weak reachability*, which we recall below.

4.1 Generalized coloring numbers

Generalized coloring numbers are key components of the algorithmic toolbox of the sparsity theory. The idea is that since in classes of bounded expansion bound edge density of shallow minors at every fixed depth r, and bounding edge density of subgraphs corresponds to bounding degeneracy, it is natural to generalize the notion of degeneracy to higher depth r as well.

We will use the following two generalized coloring numbers: admissibility and weak coloring number. We prove that the generalized coloring numbers can be approximated well in para-AC^1.

Admissibility. Suppose G is a graph and $S \subseteq V(G)$ is a subset of its vertices. The *back-connectivity* of a vertex $u \in S$ at depth r on S, denoted $\mathrm{bconn}_r(S,u)$, is the maximum cardinality of a family of paths $\mathcal{P}$ in G with the following properties:

- every path $P \in \mathcal{P}$ has length at most r, starts in u, ends in a vertex of S different from u, and all its internal vertices do not belong to S;
- every two paths from $\mathcal{P}$ share only the vertex u and otherwise are vertex-disjoint.

For a vertex ordering σ of G, the *r-admissibility* of a vertex v, denoted $\mathrm{adm}_r(G,\sigma,v)$, is equal to $\mathrm{bconn}_r(\{u\colon u \leqslant_\sigma v\}, v)$. The *$r$-admissibility* of σ is

$$\mathrm{adm}_r(G,\sigma) = \max_{v \in V(G)} \mathrm{adm}_r(G,\sigma,v)$$

and the *r-admissibility* of G, denoted $\mathrm{adm}_r(G)$, is the minimum possible r-admissibility of a vertex ordering of G.

It is known that admissibility can be used to characterize classes of bounded expansion in the following sense: a class $\mathscr{C}$ of graphs has bounded expansion if and only if there exists a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $\mathrm{adm}_r(G) \leqslant f(r)$ for each $G \in \mathscr{C}$ (see e.g. [18, 26, 36]).

We use the same trick as in Lemma 6 to compute a vertex ordering that achieves slightly worse r-admissibility, but uses a logarithmic number of rounds. It will be convenient to think again of block vertex orderings. For a block vertex ordering $\tau = (B_1, \ldots, B_\ell)$, the r-admissibility of a vertex $v \in B_i$ in this ordering is defined as $\mathrm{bconn}_r(\bigcup_{j=1}^{i} B_i, v)$, and the r-admissibility of τ is defined at the maximum r-admissibility of any vertex in τ. Again, it is straightforward to see that if τ is a block vertex ordering of admissibility at most k, then by ordering the blocks of τ arbitrarily and concatenating these orderings as in τ we obtain a vertex ordering of G of r-admissibility at most k.

The main idea behind this result is encapsulated in the following lemma.

Lemma 12 (explicit in Theorem 3.1 in [26]). *Suppose $r, d \in \mathbb{N}$, G is a graph, and S is a subset of vertices of G with the following property: for each $v \in S$, we have $\mathrm{bconn}_r(S,v) > 6rd^3$. Then G admits a depth-$(r-1)$ topological minor with edge density larger than d.*

In this work we do not use Lemma 12 directly, but we use its stronger variant — Lemma 14, to be stated later — which we prove explicitly in Appendix B.

From Lemma 12 it easily follows that for every graph G and integer r, we have

$$\mathrm{adm}_r(G) \leqslant 6r(\lceil \widetilde{\nabla}_{r-1}(G) \rceil)^3.$$

Indeed, consider the following greedy procedure. Start with $S = V(G)$ and an empty ordering, and repeatedly perform the following until S becomes empty: find a vertex $v \in S$ with minimum $\mathrm{bconn}_r(S, v)$, put v at the front of the constructed ordering, and remove v from S. Lemma 12 ensures us that at each step a vertex with r-admissibility at most $6r(\lceil \widetilde{\nabla}_{r-1}(G) \rceil)^3$ will be extracted, yielding the same bound on the r-admissibility of the final ordering.

This approach somehow mirrors the greedy algorithm for computing the degeneracy of the graph. Observe that it performs a linear number of iterations, hence a priori it not straightforward to parallelize it. We will now use the same trick as in Lemma 6 to compute a vertex ordering that achieves slightly worse r-admissibility, but uses a logarithmic number of rounds. It will be convenient to think again of block vertex orderings. For a block vertex ordering $\tau = (B_1, \ldots, B_\ell)$, the r-admissibility of a vertex $v \in B_i$ in this ordering is defined as $\mathrm{bconn}_r(\bigcup_{j=1}^{i} B_i, v)$, and the r-admissibility of τ is defined at the maximum r-admissibility of any vertex in τ. Again, it is straightforward to see that if τ is a block vertex ordering of admissibility at most k, then by ordering the blocks of τ arbitrarily and concatenating these orderings as in τ we obtain a vertex ordering of G of r-admissibility at most k.

Lemma 13. *The following transformation parameterized by integers r and d is in* para-AC^1*: Given a graph G with $\widetilde{\nabla}_{r-1}(G) \leqslant d$, compute a vertex block ordering of G with r-admissibility at most $6r^2d^3$ and length at most $\log n$.*

For the proof of Lemma 13 we need analogues of the two ingredients that we used in the proof of Lemma 6. First, we need to know that at every iteration a vast majority of the remaining vertices can be removed and set to be the next block. This is done by the following lemma.

Lemma 14. *Suppose $\varepsilon > 0$, $r, d \in \mathbb{N}$, G is a graph, and S is a subset of vertices of G with the following property: for at least $\varepsilon|S|$ vertices $v \in S$ we have $\mathrm{bconn}_r(S, v) > 6\varepsilon^{-1}rd^3$. Then G admits a depth-$(r-1)$ topological minor with edge density larger than d.*

The proof of Lemma 14 follows closely the lines of the proof of Lemma 12, however we need to argue that having large back-connectivity of at least ε-fraction of vertices of S, instead of all, is sufficient to construct a depth-$(r-1)$ topological minor model of a graph with edge density larger than d. This essentially requires modification of numerical parameters in the proof. For the sake of completeness we include the proof of Lemma 14 in Appendix B.

The second necessary ingredient is that we need to compute the set of vertices with low back-connectivity efficiently, similarly as in the proof of Lemma 6 it was necessary to compute the set of vertices with low degree in the remaining graph using Corollary 1. For this we use the following lemma, whose proof, similarly as that of Corollary 1, relies on color coding.

Lemma 15. *Consider the following problem parameterized by r and k: Given a graph G, its vertex subset S, and a vertex $u \in S$, determine whether $\mathrm{bconn}_r(S, u) < k$. Then this problem can be solved by a family $(A_{n,r,k})_{n,r,k \in \mathbb{N}}$ of AC-circuits where each $A_{n,r,k}$ has depth $\mathcal{O}(\log r)$ and size $f(r,k) \cdot n^{\mathcal{O}(1)}$, for some computable function f.*

Proof. We first recall the toolbox used by Bannach et al. [4].

Definition 2 (Universal coloring family). For integers n, k, c, an (n, k, c)-*universal coloring family* is a family Λ of functions from $[n]$ to $[c]$ with the following property: for every subset $S \subseteq [n]$ with $|S| \leqslant k$ and function $f\colon S \to [c]$, there exists $\lambda \in \Lambda$ such that the restriction of λ to S is equal to f.

Theorem 5 (Theorem 3.2 of [4]). *There is a* dlogtime*-uniform family of* AC*-circuits* $(C_{n,k,c})_{n,k,c\in\mathbb{N}}$ *without inputs such that each* $C_{n,k,c}$*:*

- *has depth* $\mathcal{O}(1)$ *and size* $f(k,c) \cdot n^{\mathcal{O}(1)}$ *for a computable function* f*; and*
- *outputs an* (n, k, c)*-universal coloring family, encoded as a list of function tables.*

Suppose Λ is a (n, rk, k)-universal coloring family on the vertex set of G (identified with $[n]$). Provided $\mathrm{bconn}_r(S, u) \geqslant k$ there exists a path family $\mathcal{P}$ witnessing this fact: $|\mathcal{P}| = k$, each $P \in \mathcal{P}$ has length at most r and leads from u to another vertex of S through vertices outside of S, and paths from $\mathcal{P}$ pairwise share only u. Letting $\mathcal{P} = \{P_1, \ldots, P_k\}$ we have $|\bigcup_{i=1}^{k}(V(P_i) - \{u\})| \leqslant rk$, hence there exists a coloring $\lambda \in \Lambda$ such that for each $i \in [k]$, all the vertices of $V(P_i) - \{u\}$ are colored with color i in λ. We shall then say that $\mathcal{P}$ is *well-colored* by λ.

This suggest the following construction of $A_{n,r,k}$. First, apply the circuit $C_{n,rk,k}$ given by Theorem 5 to construct an (n, rk, k)-universal coloring family Λ on the vertex set of G. Then, for each $\lambda \in \Lambda$ construct a circuit verifying whether there is a path family $\mathcal{P}$ as above that is well-colored by λ. For this, for each $i \in [k]$ construct the graph G_i which is the subgraph of G induced by u and all vertices of G that are colored with color i in λ. Then check whether any vertex of S is at distance at most r from u in G_i using the circuit provided by Lemma 2; this circuit has size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log r)$. The discussion from the previous paragraph proves that $\mathrm{bconn}_r(S, u) < k$ if and only if for every $\lambda \in \Lambda$ the circuit constructed for λ did not succeed in finding a path family well-colored by λ. □

Observe that Lemma 14 and Lemma 15 do not fit so nicely together for the proof of Lemma 13 as it was the case for Lemma 6. The reason is that the circuits provided by Lemma 15 have depth $\mathcal{O}(\log r)$ instead of $\mathcal{O}(1)$, so applying exactly the same strategy — of removing at each step half of vertices — would yield a circuit of depth $\mathcal{O}(\log r \cdot \log n)$, which is too much for para-AC^1. The idea is to put $\varepsilon = 1/r$ instead of $\varepsilon = 1/2$ in order to trade the approximation factor for the depth of the circuit. Thus the iteration has length $\mathcal{O}(\log n / \log r)$, so in the final circuit we need $\mathcal{O}(\log n / \log r)$ layers of depth $\mathcal{O}(\log r)$ each, and the unwanted term $\log r$ cancels out. We now provide formal details.

Proof (of Lemma 13). Consider the following iterative procedure. Starting with $S = V(G)$ and empty block vertex ordering, repeatedly find the set of those vertices $v \in S$ for which $\mathrm{bconn}_r(S, v) > 6r^2d^3$, remove it from S and put it as the next block at the front of the constructed block vertex ordering. By Lemma 14, each iteration results in decreasing the size of S by a multiplicative factor larger than r, thus the iteration terminates yielding a block vertex ordering of the whole graph within at most $\log_r n = \frac{\log n}{\log r} \leqslant \log n$ iterations. Thus, it is straightforward to see that the obtained block vertex order has length at most $\log n$ and r-admissibility at most $6r^2d^3$, as requested. For the implementation using a circuit family, by

Lemma 15 each iteration can be performed using a circuit of size $f(r,d)\cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log r)$, and we may apply $\lfloor \frac{\log n}{\log r} \rfloor$ iterations sequentially. Thus the obtained circuit has size $f(r,d)\cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log n)$, as claimed. □

Weak coloring number. Suppose G is a graph, $r \in \mathbb{N}$, and σ is a vertex ordering of G. For two vertices $u, v \in V(G)$ with $u \leqslant_\sigma v$, we say that u is *weakly r-reachable* from v if there is a path of length r in G that leads from v to u and whose all internal vertices are larger than u in σ. The set of vertices weakly r-reachable from v in σ is denoted by $\mathrm{WReach}_r[G,\sigma,v]$. The *weak r-coloring number* of σ is equal to

$$\mathrm{wcol}_r(G,\sigma) = \max_{v\in V(G)} |\mathrm{WReach}_r[G,\sigma,v]|$$

and the weak r-coloring number of G, denoted $\mathrm{wcol}(G)$, is the smallest weak r-coloring number of a vertex ordering of G.

While measuring the complexity of vertex orderings via weak coloring numbers is arguably more useful than via admissibility, it turns out that r-admissibility and weak r-coloring number are functionally equivalent. While it is straightforward that $\mathrm{adm}_r(G,\sigma) \leqslant \mathrm{wcol}_r(G,\sigma)$ for any $r \in \mathbb{N}$, graph G, and its vertex ordering σ, the weak coloring number is also bounded from above by a function of admissibility as follows:

Lemma 16 (Theorem 2.6 of [18]). *For any $c, r \in \mathbb{N}$ with $c \geqslant 2$, graph G, and vertex ordering σ of G with $\mathrm{adm}_r(G,\sigma) \leqslant c$, we have $\mathrm{wcol}_r(G,\sigma) \leqslant \frac{c^{r+1}-1}{c-1}$.*

Thus, by setting $g(r,d) \coloneqq \frac{(6r^2d^3)^{r+1}-1}{6r^2d^3-1}$ and combining Lemma 13 with Lemma 16, we obtain the following.

Theorem 6. *The following transformation parameterized by integers r and d is in* para-AC^1*: Given a graph G with $\nabla_{r-1}(G) \leqslant d$, compute a vertex ordering of G with $\mathrm{wcol}_r(G) \leqslant g(r,d)$.*

Our approach to proving Theorem 6 is as follows. The weak coloring numbers are closely related to another measure called admissibility. More precisely, every order witnessing that the r-admissibility is small also witnesses that the weak r-coloring number is small. For computing r-admissibility there exists a greedy approximation algorithm [15, 26]. We turn this greedy approximation algorithm into a low-depth circuit using a similar trick as we did for degeneracy. In each single step of the algorithm we make use of the color coding toolbox provided by Bannach et al. [4].

For applications we will need to efficiently compute the weak r-reachability relation, as expressed in the next lemma.

Lemma 17. *Consider the following problem parameterized by r: Given a graph G, its vertex ordering σ, and two vertices u and v, determine whether u is weakly r-reachable from v in σ. Then this problem can be solved by a* dlogtime*-uniform family $(W_{n,r})_{n,r\in\mathbb{N}}$ of* AC*-circuits where each $W_{n,r}$ has size $n^{\mathcal{O}(1)}$ depth $\mathcal{O}(\log r)$.*

Proof. It suffices to verify whether $u \leqslant_\sigma v$ and the distance between u and v in the subgraph induced by vertices not smaller in σ than u is at most r. The former can be read from an input gate, while the latter can be done using a circuit of size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log r)$ by Lemma 2.□

Actually, at this point we may already prove Theorem 2. Amiri et al. [1] observed that given a vertex ordering with a low weak $2r$-coloring number, an approximate distance-r dominating set can be found using a very simple selection rule.

Theorem 7 (explicit in the proof of Theorem 8 of [1]). *Suppose $r \in \mathbb{N}$, G is a graph, and σ is a vertex ordering of G. For $u \in V(G)$, let $d(u)$ be the vertex of $\mathrm{WReach}_r[G, \sigma, u]$ that is the smallest in the ordering σ, and define $D := \{d(u) \colon u \in V(G)\}$. Then D is a distance-r dominating set in G and $|D| \leqslant \mathrm{wcol}_{2r}(G, \sigma) \cdot \mathrm{dom}_r(G)$.*

Proof (of Theorem 2). Since $\mathscr{C}$ has effectively bounded expansion, for every $r \in \mathbb{N}$ there exists a constant $d \in \mathbb{N}$, computable from r, such that no graph from $\mathscr{C}$ admits a depth-$(2r-1)$ topological minor with edge density larger than d. Consequently, given an n-vertex graph $G \in \mathscr{C}$ we may apply the circuit provided by Theorem 6 to compute a vertex ordering σ of G with $\mathrm{wcol}_{2r}(G, \sigma) \leqslant g(2r, d)$, where the latter is a constant depending only on r in a computable manner. This circuit has size $f(r, d) \cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log n)$, for computable f. Next, by Lemma 17 we may compute, for every pair of vertices $u, v \in V(G)$, whether $u \in \mathrm{WReach}_r[G, \sigma, v]$ using a circuit of size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log r) \leqslant \mathcal{O}(\log n)$. Finally, for every vertex $u \in V(G)$ we may construct a circuit of depth $\mathcal{O}(1)$ that finds $d(u)$. We conclude by adding one output gate for each $v \in V(G)$ that computes whether v is to be included into D by checking whether $v = d(u)$ for any $u \in V(G)$. It follows directly from the construction that the circuit satisfies the required size and depth constraints, while the correctness of the output is asserted by Theorem 7. □

4.2 Low treedepth colorings

Being able to efficiently compute vertex orderings with low weak coloring numbers enables us to compute low treedepth colorings. Let us now introduce the relevant definitions.

A *separation forest*[2] of a graph G is a forest F on the same vertex set as G such that whenever uv is an edge in G, then either u is an ancestor of v, or v is an ancestor of u in F The *treedepth* of a graph G is the smallest possible depth of a separation forest of G. For an integer p, a coloring $\lambda \colon V(G) \to \{1, \ldots, M\}$ is a *treedepth-p coloring* of G if every i-tuple of color classes in λ, $i \leqslant p$, induces in G a graph of treedepth at most i.

It is shown in [30] that a class $\mathscr{C}$ of graphs has bounded expansion if and only if for every p there is a number M such that every graph $G \in \mathscr{C}$ admits a treedepth-p coloring using M colors. We remark that the above definition of a treedepth-p coloring can be relaxed, yielding a less restrictive definition that is sufficient for most algorithmic purposes, including our purposes in this paper. Namely, it would be sufficient to require that every p-tuple of classes induces in G a graph of treedepth at most $f(p)$, for some function $f \colon \mathbb{N} \to \mathbb{N}$. It follows from the proof in [30] that this weaker variant yields a notion that is still equivalent to having bounded expansion. Below, we use the original notion of treedepth-p colorings.

We now show how to compute low treedepth colorings orderings with low weak r-coloring numbers. Suppose G is a graph and σ is a vertex ordering of G. For $r \in \mathbb{N}$, let $G\langle r, \sigma \rangle$ be the *weak r-reachability graph* of σ, whose vertex set is $V(G)$ and where $u <_\sigma v$ are considered

[2] This notion is also called *elimination forest* in the literature; we find the name *separation forest* more explanatory.

adjacent if and only if $u \in \mathrm{WReach}_r[G, \sigma, v]$. The following lemma explains the relation between weak r-reachability graphs and low treedepth colorings. [3]

Lemma 18 (implicit in Theorem 2.6 of [38]). *For any graph G, its vertex ordering σ, and integer $p \in \mathbb{N}$, every proper coloring of $G\langle 2^{p-2}, \sigma\rangle$ is a treedepth-p coloring of G.*

Observe that if G is a graph with a vertex ordering σ such that $\mathrm{wcol}_{2^{p-2}}(G, \sigma) \leqslant c$ for some $c \in \mathbb{N}$, then σ is actually a vertex ordering of $G\langle 2^{p-2}, \sigma\rangle$ of degeneracy $c-1$. This implies that $G\langle 2^{p-2}, \sigma\rangle$ admits a proper coloring with c colors. We already know how to efficiently compute vertex orderings with low weak r-coloring numbers for graphs from any class of bounded expansion, see Theorem 6. Applying Lemma 7 to the graph $G\langle 2^{p-2}, \sigma\rangle$, we get a parallelized algorithm for computing a treedepth-p coloring of a given graph G. The main part of this section will be devoted to the proof of the following lemma. From now we assume that a coloring of an n-vertex graph with M colors is represented by Mn gates, M for each vertex u, where in an encoding of a coloring exactly one of these gates is set to true and this gate denotes the color of u. We get the following result as an immediate corollary.

Theorem 8. *Suppose $\mathscr{C}$ is a class of effectively bounded expansion. Then for every $p \in \mathbb{N}$ there exists a constant $M = M(p)$, computable from p, such that the following transformation parameterized by p is in* para-AC^1*: given a graph $G \in \mathscr{C}$, compute its treedepth-p coloring using M colors.*

Proof. Let $r = 2^{p-2}$. Since $\mathscr{C}$ has effectively bounded expansion, there exists a constant $d \in \mathbb{N}$, computable from p, such that no graph from $\mathscr{C}$ admits a depth-$(r-1)$ topological minor with edge density larger than d. Consequently, given $G \in \mathscr{C}$ we may apply the circuit provided by Theorem 6 to compute a vertex ordering σ of G with $\mathrm{wcol}_r(G, \sigma) \leqslant g(r, d)$, where the latter is again a constant computable from p. This circuit has size $f(p, d) \cdot n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log n)$, for computable f. Next, by applying the circuit provided by Lemma 17 to each pair of vertices in G we may compute the adjacency matrix of the graph $G\langle r, \sigma\rangle$. Since σ witnesses that $G\langle r, \sigma\rangle$ is $(g(r, d) - 1)$-degenerate, by applying the circuit provided by Lemma 7 to $G\langle r, \sigma\rangle$ we may compute a proper coloring of this graph using at most $M := (4g(r, d))^2$ colors, which is a constant depending on p in a computable manner. Lemma 18 asserts that this coloring is a treedepth-p coloring of G. The claimed size and depth bounds on the obtained circuit follow directly from the construction and from bounds provided by Theorem 6, Lemma 17, and Lemma 7. □

We remark that the problem of computing a low treedepth coloring can be also approached using *fraternal augmentations*, as was done e.g. in [16, 30, 35]. Both in our line of reasoning and in this approach the key step is computing a vertex ordering of low degeneracy, that is, Lemma 6.

[3] To be more precise, (the proof of) Theorem 2.6 of [38] asserts that the proper coloring of $G\langle 2^{p-2}, \sigma\rangle$ obtained by the greedy coloring procedure of Proposition 5 is a *p-centered coloring* of G. The fact that the coloring is obtained by the greedy procedure is irrelevant in the proof, the reasoning works for any proper coloring of $G\langle 2^{p-2}, \sigma\rangle$. Also, the notion of a p-centered coloring is actually stronger than that of a treedepth-p colorings: every p-centered coloring is in particular a treedepth-p coloring. We invite the reader to [36, Chapter 2] for a comprehensive presentation of these results.

4.3 Computing separation forests

A low treedepth coloring is still not enough for the model-checking algorithm to work, as we also need to compute separation forests witnessing that appropriate induced subgraphs have bounded treedepth. In general computing separation forests of optimal depth is a hard computational problem, but if one allows approximate depth there is a very simple and well-known way to do it (see Section 17.3 in [34]). The first observation is that graphs of bounded treedepth do not contain long paths.

Lemma 19 (see Section 6.2 in [34]). *A path on 2^k vertices has treedepth $k+1$.*

Since treedepth is a monotone parameter, i.e. can only decrease under taking a subgraph, we immediately obtain the following statement.

Corollary 20. *A graph of treedepth at most h does not contain a path on 2^h vertices as a subgraph. Consequently, in a graph of treedepth h every connected component has diameter smaller than 2^h.*

Given a graph G, a *DFS forest* of G is any separation forest F of G that has the following property: whenever u is a parent of v in F, then there is an edge uv in G. The name is derived from the easy fact that any forest constructed by subsequent recursive calls of a depth-first search in G is a separation forest of G with this property. If such a DFS forest F had depth at least 2^h, then this would witness that G contains a path on 2^h vertices as a subgraph, and consequently, by Lemma 19, the treedepth of G would be larger than h. Therefore, we have the following.

Lemma 21. *Any DFS forest of a graph G of treedepth at most h is a separation forest of G of depth smaller than 2^h.*

Lemma 21 gives a very simple linear-time algorithm to compute a bounded-depth separation forest of a graph of bounded treedepth — just run depth-first search on the graph and output the obtained DFS forest. However, depth-first search is an inherently sequential algorithm. As shown by Bannach et al. [5, Lemma 6], on graphs of bounded treedepth a DFS forest can be computed in para-$\mathsf{AC}^{0\uparrow}$ parameterized by treedepth.

We now reprove this result for completeness. Henceforth, every separation forest F on n vertices, in particular every DFS forest, will be represented in our circuits by its parent relation, encoded as a boolean $n \times n$ matrix.

Lemma 22 (see also Lemma 6 of [5]). *The following transformation parameterized by h is in* para-$\mathsf{AC}^{0\uparrow}$*: Given a graph of treedepth h, compute any its DFS forest.*

Proof. We first observe the following direct consequence of Corollary 20.

Claim 1. *The following problem parameterized by h is in* para-$\mathsf{AC}^{0\uparrow}$*: Given an n-vertex graph G of treedepth at most h and two vertices $u, v \in V(G)$, determine whether u and v are in the same connected component of G.*

Proof. By Corollary 20, to check whether u and v are in the same connected component of G it suffices to check whether they are at distance at most 2^h in G. By Lemma 2, this can be done by an AC-circuit of size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(h)$. ⌟

With this statement in mind, we proceed with the proof. Let G be the input graph; we assume that there is an implicit order $\preceq$ on the vertices of G, say imposed by the order of inputs.

First, let R be the set consisting of the $\preceq$-smallest vertex from each connected component of G. Observe that R can be computed as follows: a vertex u belongs to R if for every other vertex v, it is not true that $v \prec u$ and u, v are in the same connected component of G; the latter check can be done in para-$\mathsf{AC}^{0\uparrow}$ by Claim 1. The set R constitutes the roots of DFS trees in the connected components of G.

We then proceed in $2^h - 1$ rounds, where in the ith round we construct the $(i+1)$st level of the DFS forest. We maintain vertex subsets $X \subseteq Z$, initially both set to R, with the following meaning: Z is the set of vertices already placed in the constructed forest (i.e. in the ith round Z comprises vertices from levels $1, \ldots, i$ of the forest), and X is the set of vertices placed in the constructed forest in the previous round (i.e. in the ith round X comprises vertices from level i). In this round we will compute the set of vertices contained in the next, $(i+1)$st, level of the DFS forest, and for each of them we will compute its parent in the DFS forest. Transforming this encoding to the assumed encoding of separation forests via the ancestor-descendant relation will be done at the end.

To compute the next level, we perform the following steps. First, compute the set of all vertices that are not in Z, but have a neighbor in X; call it M. Also, for every vertex in M record its $\preceq$-smallest neighbor in X. Second, let Y be the set of $\preceq$-minimal vertices of M in their respective connected components of $G - Z$; that is, a vertex $v \in M$ belongs to Y if and only if there is no vertex $u \in M$ such that $u \prec v$ and u, v are in the same connected component of $G - Z$. Finally, we set Y to be the next level of the DFS forest: put $Z \coloneqq Z \cup Y$ and $X \coloneqq Y$, and for each $u \in Y$ set its recorded neighbor in X to be its parent in the DFS forest.

It is straightforward to see that the above procedure correctly computes a DFS forest of G. By Lemma 21, any DFS forest of G has depth smaller than 2^h, hence performing $2^h - 1$ rounds suffices to exhaust the whole vertex set. As for the implementation, it is easy to perform each round in para-$\mathsf{AC}^{0\uparrow}$ using Claim 1, and since the number of rounds is bounded by a computable function of h, it follows that the overall circuit family satisfies size and depth bounds for para-$\mathsf{AC}^{0\uparrow}$. □

Finally, we need to change the encoding of the separation forest, from the child-parent relation to ancestor-descendant relation. This requires computing the transitive closure of the child-parent relation. Since the computed rooted forest has depth less than 2^h, it suffices to compute the 2^h boolean power of the matrix of the child-parent relation (treated as a directed graph), which boils down to squaring this matrix h times. Hence, this can be done by a circuit of polynomial size and depth $\mathcal{O}(h)$.

5 Model checking

In this section we prove our main result, Theorem 1, and along the lines reprove the result of Dvořák et al. [16] that model-checking FO on classes of effectively bounded expansion is in linFPT.

The general idea of our proof is as follows. We prove the existence of a certain efficient quantifier-elimination procedure for classes of bounded expansion. This is first done in the case of forests of bounded depth. This lifts immediately to classes of bounded treedepth. Finally, this lifts to classes of bounded expansion, via low treedepth colorings.

5.1 Preliminaries on relational structures and logic

We assume familiarity with basic notation for relational structures. A *vocabulary* is a finite relational signature Σ consisting of relation names. Each relation name $R \in \Sigma$ has a prescribed arity $\mathrm{ar}(R)$, which is a positive integer — that is, we do not allow 0-ary relations. The *arity* of a vocabulary Σ is the maximum arity of a relation in Σ. Throughout this section all vocabularies will be of arity at most 2; that is, they consist only of unary and binary relations. Unary relations will be also called *labels* for brevity.

For a vocabulary Σ, a Σ-*structure* $\mathbb{A}$ consists of a nonempty universe $V(\mathbb{A})$ and, for each relation name $R \in \Sigma$, its *interpretation* $R^{\mathbb{A}} \subseteq V(\mathbb{A})^{\mathrm{ar}(R)}$. If the structure $\mathbb{A}$ is clear from the context we may drop the superscript, thus identifying a relation name with its interpretation in the structure. The *size* of a structure $\mathbb{A}$, denoted $|\mathbb{A}|$, is the cardinality of its universe. The *Gaifman graph* of $\mathbb{A}$, denoted $G(\mathbb{A})$, has $V(\mathbb{A})$ as the vertex set, and we make two distinct elements u, v adjacent in $G(\mathbb{A})$ iff u and v appear together in some relation in $\mathbb{A}$.

In our circuit constructions any Σ-structure $\mathbb{A}$ on n elements will be always encoded using a boolean table of length n for each unary relation in Σ and a boolean $n \times n$ matrix for each binary relation in Σ. Similarly as for graphs, in our circuit families we create one circuit $C_{n,k}$ responsible for tackling instances with n elements and parameter value k. The input to $C_{n,k}$ thus consists of $n|\Sigma_1| + n^2|\Sigma_2|$ gates, where Σ_1, Σ_2 respectively denote the sets of unary and binary relations of the vocabulary Σ.

For brevity we often write $\bar{x}$ as a shorthand for a tuple $(x_1, \ldots, x_k)$ of variables, denoting the length of the tuple by $|\bar{x}| := k$. Notation $\exists_{\bar{x}}$ is a shorthand for $\exists_{x_1} \exists_{x_2} \ldots \exists_{x_k}$. For a formula $\varphi(x_1, \ldots, x_k) \in \mathrm{FO}[\Sigma]$, by $\varphi(\mathbb{A})$ we denote the set of all tuples $(u_1, \ldots, u_k) \in V(\mathbb{A})^k$ for which $\mathbb{A} \models \varphi(u_1, \ldots, u_k)$. Formulas $\varphi(\bar{x})$ and $\varphi'(\bar{x})$ are *equivalent over* $\mathbb{A}$ if $\varphi(\mathbb{A}) = \varphi'(\mathbb{A})$, and simply *equivalent* if they are equivalent over every Σ-structure $\mathbb{A}$.

A formula $\varphi(\bar{x})$ is *quantifier-free* if it has no quantifiers, i.e., it is a boolean combination of relation checks on the free variables from $\bar{x}$. Further, $\varphi(\bar{x})$ is *existential* if it is a positive boolean combination (i.e. without negations) of formulas in *prenex existential form* $\exists_{\bar{y}}\, \psi(\bar{x}, \bar{y})$, where ψ is quantifier-free. Existential formulas are closed under positive boolean combinations. directly from the definition, but it can be easily seen that every existential formula can be reduced to prenex existential form.

Proposition 23. *For every existential formula $\varphi(\bar{x}) \in \mathrm{FO}[\Sigma]$ there exists an equivalent formula $\varphi'(\bar{x}) \in \mathrm{FO}[\Sigma]$ in the prenex existential form, computable from φ.*

5.2 Quantifier elimination on forests of bounded depth

In this section we work out a basic primitive for our quantifier elimination procedure, namely the case of unordered, labeled forests of bounded depth.

Rooted forests. Suppose $\mathbb{T}$ is a rooted, unordered forest, so far without any labels. We use standard notions like parent, child, ancestor, descendant, and we follow the convention that each node is regarded as its own ancestor and descendant. The size $|\mathbb{T}|$ of a forest $\mathbb{T}$ is the number of nodes in it. The *depth* of a node x is the number of its ancestors, and the *depth* of a forest $\mathbb{T}$ is the largest depth of a node in $\mathbb{T}$.

Throughout this section we work with forests of depth at most d, for a fixed constant $d \in \mathbb{N}$. A forest $\mathbb{T}$ of depth at most d will be modeled as a relational structure whose universe is the node set and there is one binary relation parent, where $\mathsf{parent}(x, y)$ holds if x is the parent of y. That is, such a $\{\mathsf{parent}\}$-structure models a forest of depth at most d if the following conditions hold: each node has at most one parent and after following the parent relation from any node we always reach a node with no parent within at most $d - 1$ steps.

The following formulas will play a key role in our reasonings.

Proposition 24. *There exist existential formulas*

$$\mathsf{lcd}_0(x, y), \mathsf{lcd}_1(x, y), \ldots, \mathsf{lcd}_d(x, y) \in \mathrm{FO}[\mathsf{parent}],$$

each of quantifier depth at most $2d$*, such that for every forest* $\mathbb{T}$ *of depth at most* d*,* $i \in [0, d]$*, and nodes* x, y *we have* $\mathbb{T} \models \mathsf{lcd}_i(x, y)$ *if and only if* x *and* y *have exactly* i *common ancestors in* $\mathbb{T}$*.*

Proof. Make a disjunction over possible depths i_x of x and i_y of y. For fixed (i_x, i_y), quantify existentially the i_x ancestors of x and i_y ancestors of y, and check that exactly the first i of them are equal to each other. □

Observe that the condition in Proposition 24 can be equivalently stated as follows: $\mathsf{lcd}_0(x, y)$ holds iff x and y have no common ancestor (i.e. they reside in different trees of the forest), and for $i \geqslant 1$ $\mathsf{lcd}_i(x, y)$ holds iff the least common ancestor of x and y is at depth i in $\mathbb{T}$. Note that for a node x, the formula $\mathsf{lcd}_i(x, x)$ holds if and only if x is at depth i. Moreover, the condition that x is an ancestor of y can be expressed as follows: $\mathsf{lcd}_i(x, y)$ holds iff $\mathsf{lcd}_i(x, x)$ holds, for all $i \in [0, d]$. Thus the formulas lcd_i can be used to check the depths of nodes and the ancestor relation (using boolean combinations).

For a forest $\mathbb{T}$ and a finite label set Λ, a Λ-*labeling* of $\mathbb{T}$ is any structure obtained from $\mathbb{T}$ by adding a unary relation c for each label $c \in \Lambda$. A Λ-labeling of a forest will be also called a Λ-*labeled forest*. For another label set $\widehat{\Lambda}$, a $\widehat{\Lambda}$-*relabeling* of a Λ-labeled forest $\mathbb{T}$ is any $\widehat{\Lambda}$-labeled forest $\mathbb{S}$ such that the underlying unlabeled forests of $\mathbb{S}$ and $\mathbb{T}$ are equal; that is, $\mathbb{S}$ is obtained from $\mathbb{T}$ by clearing all labels from Λ and adding new labels from $\widehat{\Lambda}$ in an arbitrary way. We may drop the label set used in a labeled forest if we do not wish to specify it.

Lcd types. Fix a label set Λ; we consider Λ-labeled forests of depth at most d. A formula $\psi(\bar{x}) \in \mathrm{FO}[\{\mathsf{parent}\} \cup \Lambda]$ is *lcd-reduced* if it uses neither quantifiers nor the parent relation, but it may use formulas lcd_i for $i \in [0, d]$ as atoms. That is, an lcd-reduced formula is a boolean combination of label tests and formulas lcd_i. Note that lcd-reduced formulas are closed under boolean combinations.

Suppose $\bar{x} = (x_1, \ldots, x_k)$ is a tuple of variables. For each choice of functions $\gamma \colon \bar{x} \to \mathcal{P}(\Lambda)$ and $\delta \colon \bar{x} \times \bar{x} \to [0, d]$ we define the *lcd-type formula*

$$\mathsf{type}_{\gamma,\delta}(\bar{x}) \in \mathrm{FO}[\{\mathsf{parent}\} \cup \Lambda]$$

as the lcd-reduced formula stating the following:

- for each $i \in [k]$, $c(x_i)$ holds for all $c \in \gamma(x_i)$ and $c(x_i)$ does not hold for all $c \in \Lambda - \gamma(x_i)$; and

- for all $i, j \in [k]$, the number of common ancestors of x_i and x_j is $\delta(x_i, x_j)$.

Observe that the second condition implies that the depth of x_i is equal to $\delta(x_i, x_i)$.

Note that lcd-type formulas with k free variables are mutually exclusive and cover all k-tuples: for each tuple of nodes $(u_1, \ldots, u_k)$ there is exactly one choice of γ and δ for which $\mathsf{type}_{\gamma,\delta}(u_1, \ldots, u_k)$ holds. Let $\mathbf{Types}(k, d, \Lambda)$ be the set of lcd-type formulas with k free variables for depth d and label set Λ; note that $\mathbf{Types}(k, d, \Lambda)$ is finite and computable from k, d, Λ.

Observe that for an lcd-reduced formula $\varphi(\bar{x})$ and a tuple $\bar{u}$ of nodes with $|\bar{u}| = |\bar{x}|$, knowing which lcd-type formula from $\mathbf{Types}(k, d, \Lambda)$ is satisfied on $\bar{u}$ is sufficient to determine whether $\varphi(\bar{u})$ holds. This immediately yields the following.

Proposition 25. *For every lcd-reduced formula $\varphi(\bar{x}) \in \mathrm{FO}[\{\mathsf{parent}\} \cup \Lambda]$ there exists a formula $\varphi'(\bar{x}) \in \mathrm{FO}[\{\mathsf{parent}\} \cup \Lambda]$ of the form*

$$\varphi'(\bar{x}) = \bigvee_{\beta \in I} \beta(\bar{x})$$

for some $I \subseteq \mathbf{Types}(k, d, \Lambda)$ such that φ and φ' are equivalent over every Λ-labeled forest of depth at most d. Furthermore, φ' can be computed from d, Λ, and φ.

An lcd-reduced formula $\varphi(\bar{x})$ is in the *basic normal form* if it is a disjunction of basic formulas applied to $x_1, \ldots, x_k$, that is, it is of the form

$$\varphi(\bar{x}) = \bigvee_{\beta \in I} \beta(\bar{x})$$

for some $I \subseteq \mathbf{Types}(k, d, \Lambda)$.

Quantifier elimination. We now show how to eliminate a single existential quantifier for bounded depth forests. This will be used later for quantifier elimination on low-treedepth decompositions.

Lemma 26. *Let $d \in \mathbb{N}$ and Λ be a label set. Then for every formula $\varphi(\bar{x}) \in \mathrm{FO}[\{\mathsf{parent}\} \cup \Lambda]$ with $|\bar{x}| \geqslant 1$ and of the form*

$$\varphi(\bar{x}) = \exists_y \ \psi(\bar{x}, y)$$

where ψ is lcd-reduced, and every Λ-labeled forest $\mathbb{T}$ of depth at most d, there exists a label set $\widehat{\Lambda}$, an lcd-reduced formula $\widehat{\varphi}(\bar{x}) \in \mathrm{FO}[\{\mathsf{parent}\} \cup \widehat{\Lambda}]$, and a $\widehat{\Lambda}$-relabeling $\mathbb{S}$ of $\mathbb{T}$ such that $\varphi(\mathbb{T}) = \widehat{\varphi}(\mathbb{S})$.

Moreover, the following effectiveness assertions hold. The label set $\widehat{\Lambda}$ is computable from d and Λ, the formula $\widehat{\varphi}$ is computable from φ, d, Λ, and the following transformation which computes $\mathbb{S}$ given $\mathbb{T}$, parameterized by φ, d, Λ is in linFPT *and in* para-$\mathsf{AC}^{0\uparrow}$.

Proof. We first present the combinatorial construction of $\widehat{\Lambda}$, $\mathbb{S}$, and $\widehat{\varphi}$. The effectiveness assertions will be discussed at the end.

We start by observing that without loss of generality we may assume that $\psi(\bar{x}, y)$ is an lcd-type formula. Indeed, by Proposition 25 we may assume that $\psi(\bar{x}, y)$ is of the form $\bigvee_{\beta \in I} \beta(\bar{x}, y)$ for some $I \subseteq \mathbf{Types}(k+1, d, \Lambda)$. Since existential quantification commutes with disjunction, φ is equivalent to the formula $\bigvee_{\beta \in I}(\exists_y \ \beta(\bar{x}, y))$. Suppose now that for each $\beta \in I$ we find a

suitable lcd-reduced formula $\widehat{\beta}(\bar{x})$ and a $\widehat{\Lambda}_\beta$-recoloring s_β of $\mathbb{T}$. Having set $\widehat{\Lambda} \coloneqq \bigcup_{\beta \in I} \widehat{\Lambda}_\beta$, we can take $\mathbb{S}$ to be the union relabeling, obtained by including all labels from all relabelings $(s_\beta)_{\beta \in I}$, and define $\widehat{\varphi}(\bar{x}) \coloneqq \bigvee_{\beta \in I} \widehat{\beta}(\bar{x})$. Note that thus $\widehat{\varphi}$ is lcd-reduced.

Therefore, from now on we assume that

$$\psi(\bar{x}, y) = \mathsf{type}_{\gamma,\delta}(\bar{x}, y)$$

for some $\gamma \colon \bar{z} \to \mathcal{P}(\Lambda)$ and $\delta \colon \bar{z} \times \bar{z} \to [0, d]$, where $\bar{z}$ is $\bar{x}$ with y appended.

Let us inspect values $\delta(x_i, y)$ for $i \in [k]$ and without loss of generality assume that $\delta(x_1, y)$ is the largest of them; here we use the assumption that $k \geqslant 1$. Denote $h_y = \delta(y, y)$, $h_1 = \delta(x_1, x_1)$, and $h = \delta(x_1, y)$. Note that we may assume that h_y and h_1 are larger or equal to $\max(h, 1)$, for otherwise ψ is not satisfiable and we may return an always false formula as $\widehat{\varphi}$.

From now on we assume for simplicity that $h > 0$, that is, x_1 and y have a common ancestor. The case $h = 0$ is essentially the same and differs in notation details; we discuss it at the end.

Call a node w in a Λ-labeled forest $\mathbb{T}$ a *candidate* if w has depth h_y and *label pattern* $\gamma(y)$: $c(w)$ holds for all $c \in \gamma(y)$ and $c(w)$ does not hold for all $c \in \Lambda - \gamma(y)$. For a node v of $\mathbb{T}$ at depth h, we define an integer $\kappa(v)$ as follows:

$$\kappa(v) \coloneqq \text{the number of subtrees rooted at children of } v \text{ that contain a candidate.}$$

Note that in case $h = h_y$ the above value is meaningless — it is always equal to 0. Hence, in this case we redefine $\kappa(v)$ to be equal to 1 if v is a candidate and to 0 if v is not a candidate.

We now define the relabeling $\mathbb{S}$ of $\mathbb{T}$. First, we include in $\mathbb{S}$ all the labels from $\mathbb{T}$; thus the final label set $\widehat{\Lambda}$ will be a superset of Λ. Next, for every node u record the following finite information using new labels at u:

(a) Provided u is at depth h_1, record $\kappa(v)$ where v is the unique ancestor of u at depth h, or ∞ if this number is larger than k.

(b) Provided u is at depth larger than h, record whether there exists a candidate w such that the lowest common ancestor of u and w is at depth larger than h.

The above information can be recorded using $k + 2$ new labels: $k + 1$ to record possible values in Item 19, and 1 to record the boolean value in Item 19. Thus, $\widehat{\Lambda}$ is obtained by adding these $k + 2$ new labels to Λ.

Having defined $\mathbb{S}$, we now write an lcd-reduced formula $\widehat{\varphi}(\bar{x}) \in \mathrm{FO}[\{\mathsf{parent}\} \cup \widehat{\Lambda}]$ that, when applied on some evaluation of $\bar{x}$ in $\mathbb{S}$, verifies whether $\exists_y \, \psi(\bar{x}, y)$ holds in $\mathbb{T}$. Obviously we may start with verifying that:

- the label pattern of x_i is equal to $\gamma(x_i)$, for each $i \in [k]$; and
- for all $1 \leqslant i \leqslant j \leqslant k$, the number of common ancestors of x_i and x_j is $\delta(x_i, x_j)$.

Both these checks can be easily done by lcd-reduced formulas. We are left with writing an lcd-reduced formula which verifies that, in addition to the above, a suitable node y exists. Actually, this formula will only use the new labels at nodes $x_1, \ldots, x_k$.

Observe that if nodes $u_1, \ldots, u_k, w$ in a forest $\mathbb{T}$ are such that $\mathbb{T} \models \psi(u_1, \ldots, u_k, w)$, then the subforest of $\mathbb{T}$ (with labels forgotten) formed the ancestors of $u_1, \ldots, u_k, w$ is uniquely determined

by the function δ, up to isomorphism. Let this unique forest be f; note that f can be either uniquely constructed from δ alone, or values contained in δ witness that no such f exists and ψ is not satisfiable, in which case we may return an always false formula as $\widehat{\varphi}$. We may naturally label some nodes of f with variables $x_1, \ldots, x_k, y$; note that thus each leaf of f is labelled. Whenever nodes $u_1, \ldots, u_k, w$ in a tree $\mathbb{T}$ are such that $\mathbb{T} \models \psi(u_1, \ldots, u_k, w)$, then we may define a natural isomorphism η from f to the ancestor closure of $\{u_1, \ldots, u_k, w\}$ in $\mathbb{T}$ by first mapping y to w and each x_i to u_i, and then extending the mapping to their ancestors.

We first resolve the corner case when $h = h_y$; equivalently y is an ancestor of x_1 in f. We claim that then it suffices to check whether the information encoded at x_1 in point (a) of the construction of f asserts that in the unique ancestor of x_1 at depth $h = h_y$ the value of $\kappa(\cdot)$ is positive. Indeed, this unique ancestor needs to be the evaluation of y, and since $h = h_y$, we have that $\kappa(\cdot)$ at this node is 0 if it does not have label pattern $\gamma(y)$, and 1 if it has label pattern $\gamma(y)$ (recall that in case $h = h_y$ we redefined $\kappa(v)$ to be equal to 1 if v is a candidate and to 0 if v is not a candidate). Hence, from now on we assume that $h_y > h$.

Let z be the unique ancestor at depth h of y in f; equivalently z is the least common ancestor of y and x_1 in f. Note that z exists by the assumption $h > 0$ and $z \neq y$ by the assumption $h_y > h$. Let q be the child of z that is also an ancestor of y. Then the subtree of f rooted at q is a path with y as the only leaf, and this subtree does not contain any node from $\{x_1, \ldots, x_k\}$. Let $p_1, \ldots, p_\ell$ be the other children z in f, i.e. those that are not ancestors of y. Note that $\ell \leqslant k$.

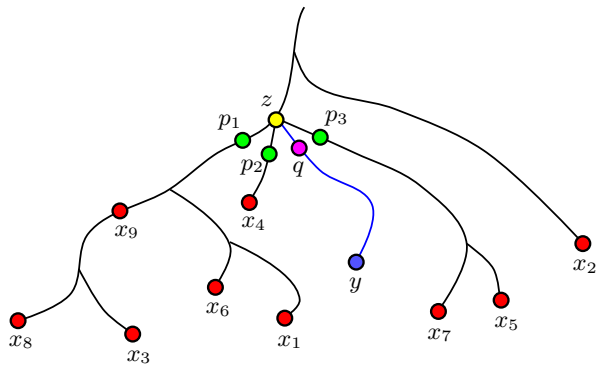

Figure 1: Example forest f and nodes mentioned in the proof. In this example, to compute the value of ℓ' it suffices to count for how many of (the evaluations of) variables x_1, x_4, x_5 we have recorded information "true" in point Item 19 of the construction of $\mathbb{S}$.

Suppose we have a Λ-labeled tree $\mathbb{T}$ and nodes $u_1, \ldots, u_k, w$ of $\mathbb{T}$ such that $\mathbb{T} \models \psi(u_1, \ldots, u_k, w)$. Let η be the natural isomorphism from f to the ancestor-closure of $\{u_1, \ldots, u_k, w\}$ in $\mathbb{T}$. Observe that knowing the labels at $u_1, \ldots, u_k$ in $\mathbb{S}$, we may uniquely deduce which of the subtrees of $\mathbb{T}$ rooted at the children $\eta(p_1), \ldots, \eta(p_\ell)$ of $\eta(z)$ contain a candidate. Indeed, each subtree of $\mathbb{S}$ rooted at some p_j, $j \in [\ell]$, contains some node x_i, $i \in [k]$, and then the corresponding subtree of $\mathbb{T}$ rooted at $\eta(p_j)$ contains a candidate if and only if we recorded information "true" at the node u_i in point (b) of the construction of $\mathbb{S}$. Let $\ell' \leqslant \ell$ be the number of those subtrees.

Now comes the crux: having evaluated $x_1, \ldots, x_k$ to $u_1, \ldots, u_k$, in order to verify whether a suitable evaluation of y exists it suffices to check whether $\kappa(v) > \ell'$, where v is the unique ancestor of u_1 at depth h; note that $v = \eta(z)$ in the notation of the previous paragraph. Indeed,

if such evaluation w of y exists, then the subtree rooted at a child of v containing w is not among the ones contributing to the value of ℓ', and hence it witnesses that $\kappa(v) > \ell'$. Conversely, if $\kappa(v) > \ell'$ then there exists a child a of v different from $\eta(p_1), \ldots, \eta(p_\ell)$, such that a has a candidate w as a descendant. Then evaluating y to w makes $\psi(u_1, \ldots, u_k, w)$ satisfied.

Note that since $\ell' \leqslant \ell \leqslant k$, the inequality $\kappa(v) > \ell'$ will always hold if $\kappa(v) > k$. Therefore, the information recorded at u_1 in point (a) of the construction of $\mathbb{S}$ is sufficient to determine whether $\kappa(v) > \ell'$. The computation presented above — determination of ℓ' and verification whether $\kappa(v) > \ell'$ — can be easily encoded using a formula that accesses only labels at nodes $u_1, \ldots, u_k$ in $\mathbb{S}$. This concludes the construction of $\widehat{\varphi}$ in the case $h > 0$.

When $h = 0$ the difference is that the node z — the lowest common ancestor of y and x_1 in f — does not exist. However, then y is in a different tree of the forest f than all other nodes $x_1, \ldots, x_k$. We may define q to be the root of the tree of f containing y and $p_1, \ldots, p_\ell$ to be the roots of all other trees of f. Also, in point (a) of the construction of $\mathbb{S}$ we replace $\kappa(v)$ (as now v does not exist) with the number of trees in f that contain a candidate. The rest of the reasoning is exactly the same.

We are left with discussing the effectiveness assertions. We discuss the case when ψ is an lcd-type formula, as the lift to arbitrary lcd-reduced formulas is immediate, both for linFPT and para-AC^0. It is clear that $\widehat{\Lambda}$ is computable from d, Λ and $\widehat{\varphi}$ is computable from φ, d, Λ.

First, we argue that $\mathbb{S}$ can be constructed in linear FPT time. This boils down to computing information added in points (a) and (b) for each node u of $\mathbb{T}$. For point (a), it suffices to run a suitable depth-first search in $\mathbb{T}$, where the return value from a subtree is the information whether it contains a candidate. For point (b), we may first use depth-first search to mark all nodes of $\mathbb{T}$ that have a descendant that is a candidate. Then we may apply a second depth-first search that computes the sought information; here it suffices to remember whether on the path from a root to the current node there was a marked node at depth larger than h.

Finally, to implement the transformation from $\mathbb{T}$ to $\mathbb{S}$ in para-$\mathsf{AC}^{0\uparrow}$ we do the following. First, compute the ancestor relation in $\mathbb{T}$ by computing the dth boolean power of the matrix of the parent relation in $\mathbb{T}$; this can be done by a polynomial-size circuit of depth $\mathcal{O}(\log d)$ by iterative squaring. Next, for every pair of nodes x and y compute the number of common ancestors of x and y; this can be done in para-AC^0 using Theorem 3. Using the above, the information recorded in point 19 for each node u can be computed in para-AC^0 using Theorem 3, while the information recorded in point Item 19 can be computed in para-AC^0 directly from the definition. □

5.3 Quantifier elimination for bounded expansion classes

Skeletons. We first introduce *skeletons*, which are relational structures formed by putting a bounded number of forests of bounded depth on top of each other. Essentially, they will be our abstraction for low treedepth decompositions.

Let Γ be a vocabulary (of arity 2) and let $d \in \mathbb{N}$. A Γ-structure $\mathbb{A}$ is a Γ-*skeleton of depth* d if for every binary relation $R \in \Gamma$, the structure obtained from $\mathbb{A}$ by dropping all relations apart from R and preserving the universe is a rooted forest of depth at most d, with R serving the role of the parent relation. Note that thus *all* binary relations in Γ serve the roles of bounded-depth forests. The reader should think of a skeleton of depth d as the union of several depth-d forests over the same universe, which is moreover labeled with some label set (i.e. with unary relations from Γ).

For every binary relation $R \in \Gamma$ and $i \in [0, d]$ we may construct a formula $\mathsf{lcd}_i^R(x, y)$ as in Proposition 24, but using R instead of parent. As before, a formula $\varphi(\bar{x}) \in \mathrm{FO}[\Gamma]$ is *lcd-reduced* if it does not use any quantifiers or binary relations, but may use formulas lcd_i^R as atoms; thus, it is a boolean combination of label checks and atoms lcd_i^R. We note that lcd-reduced formulas can be easily turned into existential formulas.

Lemma 27. *For every $d \in \mathbb{N}$ and lcd-reduced formula $\varphi(\bar{x}) \in \mathrm{FO}[\Gamma]$ there exists an existential formula $\xi(\bar{x}) \in \mathrm{FO}[\Gamma]$ such that φ and ξ are equivalent over every Γ-skeleton of depth at most d. Furthermore, ξ can be computed from d, Γ, and φ.*

Proof. Since $\varphi(\bar{x}) \in \mathrm{FO}[\Gamma]$ is a boolean combination of label checks and atoms lcd_i^R, it may be rewritten as an equivalent formula $\zeta(\bar{x})$ in conjunctive normal form: $\zeta(\bar{x})$ is a conjunction over clauses, where each clause is a disjunction of literals: label checks, atoms lcd_i^R, and their negations. Observe that each negation of an atom $\mathsf{lcd}_i^R(x, y)$ may be replaced by the formula

$$\bigvee_{j \in [0,d],\, j \neq i} \mathsf{lcd}_j^R(x, y).$$

By applying such replacement exhaustively in $\zeta(\bar{x})$ we obtain a formula $\zeta'(\bar{x})$ that is equivalent to $\zeta(\bar{x})$ over every Γ-skeleton of depth d. Moreover, $\zeta(\bar{x})$ is again in conjunctive normal form, however now atoms lcd_i^R appear only positively. By replacing these atoms with existential formulas provided by Proposition 24 we turn $\zeta'(\bar{x})$ into a positive boolean combination of existential formulas, hence into an existential formula. □

For future reference we observe that lcd-reduced formulas can be efficiently evaluated. Let us remark that as far as sequential algorithms are concerned, we always assume that all rooted forests, including forests in skeletons, are encoded by specifying for each element its parent; thus we assume that this parent can be found in constant time in the RAM model.

Lemma 28. *The following problem parameterized by $d \in \mathbb{N}$ and an lcd-reduced formula $\alpha(\bar{x}) \in \mathrm{FO}[\Gamma]$ is in* para-$\mathsf{AC}^{0\uparrow}$ *and can be computed in time $\mathcal{O}(d|\alpha|)$: given a Γ-skeleton $\mathbb{A}$ of depth d and a tuple $\bar{u} \in V(\mathbb{A})^{|\bar{x}|}$, verify whether $\mathbb{A} \models \alpha(\bar{u})$.*

Proof. Observe that in $\alpha(\bar{u})$ every atom of the form $\mathsf{lcd}_h^R(u_i, u_j)$ can be evaluated in time $\mathcal{O}(d)$ by pursuing the parent relation R from u_i and u_j at most d times. Moreover, for each binary relation $R \in \Sigma$, interpreted as a parent relation in a forest of depth at most d, we can compute the corresponding ancestor relation using a circuit of size $n^{\mathcal{O}(1)}$ and depth $\mathcal{O}(\log d)$: just compute the dth boolean power of the matrix of R using iterative squaring. With this information available, all atoms of the form $\mathsf{lcd}_h^R(u_i, u_j)$ can be evaluated in para-AC^0 using Theorem 3. Having evaluated all the atoms it is straightforward to evaluate the whole formula within the stated complexity bounds. □

Guarded structures. For the remainder of this section we fix a graph class $\mathscr{C}$ with effectively bounded expansion. Without loss of generality we may assume that $\mathscr{C}$ is closed under taking subgraphs. Let us fix the function $M(\cdot)$ given by Theorem 8 for the class $\mathscr{C}$; this means that given a graph $G \in \mathscr{C}$ and parameter p we may compute a treedepth-p coloring of G using at

most $M(p)$ colors in para-AC^1. We note that the same computational task can be also done in linFPT by the results of Nešetřil and Ossona de Mendez [30, 33].

A structure $\mathbb{A}$ is *guarded* by $\mathscr{C}$ if the Gaifman graph of $\mathbb{A}$ belongs to $\mathscr{C}$. Further, a structure $\mathbb{B}$ with the same universe as $\mathbb{A}$ (but possibly different vocabulary) is *guarded* by $\mathbb{A}$ if the Gaifman graph of $\mathbb{B}$ is a subgraph of the Gaifman graph of $\mathbb{A}$. Note that if $\mathbb{A}$ guards $\mathbb{B}$ and $\mathbb{B}$ guards $\mathbb{C}$ then $\mathbb{A}$ guards $\mathbb{C}$, and if further $\mathbb{A}$ is guarded by $\mathscr{C}$, then so are $\mathbb{B}$ and $\mathbb{C}$.

Quantifier elimination. We finally proceed to our main goal, the quantifier elimination procedure for FO on structures with Gaifman graphs from $\mathscr{C}$. The following definition explains our goal in this procedure.

Definition 3. Let Σ be a vocabulary (of arity 2) and let $\varphi(\bar{x}) \in \mathrm{FO}[\Sigma]$ for a tuple of variables $\bar{x} = (x_1, \ldots, x_k)$. We say that $\varphi(\bar{x})$ is *reducible* if there exists $d \in \mathbb{N}$, a vocabulary Γ, an lcd-reduced formula $\alpha(\bar{x}) \in \mathrm{FO}[\Gamma]$, and, for every Σ-structure $\mathbb{A}$ guarded by $\mathscr{C}$, a Γ-skeleton $\mathbb{B}$ of depth at most d guarded by $\mathbb{A}$ such that $\varphi(\mathbb{A}) = \alpha(\mathbb{B})$.

This reducibility is *effective* if d, Γ, and α are computable from Σ and φ, and the transformation computing $\mathbb{B}$ given $\mathbb{A}$, parameterized by Σ and φ, is in linFPT and para-AC^1.

Note that in Definition 3, the fact that $\mathbb{A}$ is guarded by $\mathscr{C}$ and guards $\mathbb{B}$, entails that $\mathbb{B}$ is guarded by $\mathscr{C}$. Hence we may further apply further reducibility on the structure $\mathbb{B}$ and so on. This chaining property of the notion of reducibility will be crucial in our reasoning.

In the following, whenever a vocabulary of a formula is not specified, it is an arbitrary vocabulary. Given Definition 3, quantifier elimination can be stated in a very simple way.

Theorem 9. *Every formula $\varphi(\bar{x})$ with $|\bar{x}| \geqslant 1$ is effectively reducible.*

The proof of Theorem 9 is by induction on the structure of the formula. We find it most convenient to directly solve the case of existential formulas first, from which both the induction base and the induction step will follow.

Lemma 29. *Every existential formula with at least one free variable is effectively reducible.*

Proof. Let $\varphi(\bar{x}) \in \mathrm{FO}[\Sigma]$ be the formula in question, where Σ is a vocabulary and $\bar{x} = (x_1, \ldots, x_k)$ are the free variables of φ. We may assume that φ is in prenex existential form, say, $\varphi(\bar{x}) = \exists_{\bar{y}}\, \psi(\bar{x}, \bar{y})$, where $\bar{y} = (y_1, \ldots, y_\ell)$ and $\psi(\bar{x}, \bar{y})$ is quantifier-free. Suppose $\mathbb{A}$ is the given Σ-structure guarded by $\mathscr{C}$. We describe how a suitable skeleton $\mathbb{B}$ guarded by $\mathbb{A}$ should be constructed, while its depth d, its vocabulary Γ, and the final lcd-reduced formula $\alpha(\bar{x})$ will be constructed along the way.

Let $p = k + \ell$ and let $G = G(\mathbb{A})$. Since $G \in \mathscr{C}$, there is a treedepth-p coloring $\lambda\colon V(\mathbb{A}) \to [M]$ of G, where $M = M(p)$, which can be computed in para-AC^1 and in linFPT using Theorem 8 and the results of [33, 30], respectively. Let $\mathcal{U}$ be the family of all subsets of $[M]$ of size p. For $C \in \mathcal{U}$, let $V^C = \lambda^{-1}(C)$ be the set of elements with colors from C, and let $G^C = G[V^C]$ be the subgraph induced by them. Then G^C has treedepth at most p. As observed before, we may compute, for each $C \in \mathcal{U}$, a DFS forest F^C of the induced subgraph G^C of depth at most $d \coloneqq 2^p - 1$ in para-$\mathsf{AC}^{0\uparrow}$ (and in linFPT by just running a depth-first search).

Using Lemma 22, we may compute, for each $C \in \mathcal{U}$, a DFS forest F^C of the induced subgraph G^C of depth at most $d := 2^p - 1$. This can be done in para-$\mathsf{AC}^{0\uparrow}$ by Lemma 22 and in linFPT by just running a depth-first search.

Fix any $C \in \mathcal{U}$ and let $\mathbb{T}^C$ be the unlabeled forest of depth at most d with node set V^C and binary relation parent^C interpreted as the parent relation of F^C. We now prove that the substructure induced in $\mathbb{A}$ by V^C can be entirely encoded in a labeling of $\mathbb{T}^C$.

Claim 2. *There exists a label set Λ^C, a Λ^C-labeling $\mathbb{S}^C$ of $\mathbb{T}^C$, and, for every relation $R \in \Sigma$, an lcd-reduced formula η_R^C with as many free variables as the arity of R such that $R(\mathbb{A}[V^C]) = \eta_R^C(\mathbb{S}^C)$, where $\mathbb{A}[V^C]$ denotes the substructure of $\mathbb{A}$ induced by V^C.*

Proof. For each $u \in V^C$, record the following information using labels at u:

(a) For each unary relation $R \in \Sigma$, record whether $R(u)$ holds.

(b) For each binary relation $R \in \Sigma$ and each $i \in [d]$ not larger than the depth of u, let v be the unique ancestor of u at depth i. Record whether $R(u, v)$ holds and whether $R(v, u)$ holds.

Note that to record the information above we may use a set Λ^C consisting of at most $2d|\Sigma|$ labels: one label for each unary relation in Σ, and two labels for each binary relation in Σ and each $i \in [d]$. Let $\mathbb{S}^C$ be the obtained Λ^C-labeling of $\mathbb{T}^C$.

We now write the lcd-reduced formula η_R^C for a relation $R \in \Sigma$. If $R(x)$ is unary, we may simply check an appropriate new label of x, added in point 23. If $R(x, y)$ is binary, we make a disjunction over two cases: either x is an ancestor of y in $\mathbb{S}^C$ or vice versa. If x is an ancestor of y, then denoting the depth of x as i, it can be checked whether $R(x, y)$ holds by reading one of the two labels added in point 23 at y for R and depth i. The case when y is an ancestor of x is symmetric. If neither of x, y is an ancestor of the other, then $R(x, y)$ is surely false, because F^C is a DFS forest of the G^C. This concludes the construction of η_R^C; note that the above verification can be indeed encoded using an lcd-reduced formula. ⌟

Now consider formula $\psi^C(\bar{x}, \bar{y}) \in \mathrm{FO}[\{\mathsf{parent}^C\} \cup \Lambda^C]$ obtained from $\psi(\bar{x}, \bar{y})$ by replacing each relation symbol R with the corresponding formula η_R^C. Since ψ was quantifier-free, ψ^C is lcd-reduced. Let

$$\varphi^C(\bar{x}) := \exists_{\bar{y}}\ \psi^C(\bar{x}, \bar{y}).$$

Since $k \geqslant 1$, we may iteratively apply Lemma 26 to consecutive quantifiers in φ^C, starting with the deepest. This yields a new label set $\widehat{\Lambda}^C$, a $\widehat{\Lambda}^C$-relabeling $\mathbb{U}^C$ of $\mathbb{S}^C$, and an lcd-reduced formula $\alpha^C(\bar{x})$ such that $\alpha^C(\mathbb{U}^C) = \varphi^C(\mathbb{S}^C)$.

We now build $\mathbb{B}$ and its vocabulary Γ. Start by setting the universe of $\mathbb{B}$ to be equal to the universe of $\mathbb{A}$, and Γ is so far empty. For each $C \in \mathcal{U}$ add a unary relation class^C to Γ, and interpret it in $\mathbb{B}$ so that it selects the vertices of V^C. Next, import all the relations from all structures $\mathbb{U}^C$ to $\mathbb{B}$. That is, for each $C \in \mathcal{U}$ we add $\{\mathsf{parent}^C\} \cup \widehat{\Lambda}^C$ to the vocabulary Γ, while the interpretations of these relations are taken from $\mathbb{S}^C$. Note that thus elements outside of V^C do not participate in relations parent^C.

This concludes the construction of Γ and $\mathbb{B}$. Note that the only binary relations in Γ are the relations parent^C for $C \in \mathcal{U}$, and in $\mathbb{B}$ each of them induces a forest of depth at most d. Moreover,

since each F^C is a DFS forest of G^C, it follows that $\mathbb{B}$ is guarded by $\mathbb{A}$. Hence, $\mathbb{B}$ is a Γ-skeleton of depth d guarded by $\mathbb{A}$, as requested. Consider the formula

$$\alpha(\bar{x}) \coloneqq \bigvee_{C \in \mathcal{U}} \left(\alpha^C(\bar{x}) \wedge \bigwedge_{i=1}^{k} \mathsf{class}^C(x_i) \right).$$

Observe that $\alpha(\bar{x})$, as a boolean combination of lcd-reduced formulas, is lcd-reduced. We claim that $\alpha(\mathbb{B}) = \varphi(\mathbb{A})$. On one hand, by the construction it is clear that α selects only tuples that satisfy φ, thus $\alpha(\mathbb{B}) \subseteq \varphi(\mathbb{A})$. To see the reverse inclusion, observe that whenever we have some valuation $\bar{u}$ of $\bar{x}$ such that $\varphi(\bar{u})$ holds, this is witnessed by the existence of some valuation $\bar{v}$ of $\bar{y}$ such that $\psi(\bar{u}, \bar{v})$ holds. Since $|\bar{u}| + |\bar{v}| = k + \ell = p$ and λ was a treedepth-p coloring, there exists $C \in \mathcal{U}$ such that all elements of $\bar{u}$ and $\bar{v}$ belong to V^C. For this C the formula $\alpha^C(\bar{u}) \wedge \bigwedge_{i=1}^{k} \mathsf{class}^C(u_i)$ will be satisfied and, consequently, $\bar{u}$ will be included in $\alpha(\mathbb{B})$.

This proves reducibility. Effectiveness follows immediately from complexity bounds provided by the invoked results and a straightforward implementation of the construction of Claim 2. □

We now use Lemma 29 to give all the ingredients needed for the inductive proof of Theorem 9.

Lemma 30. *The following assertions hold:*

(a) Every quantifier-free formula with at least one free variable is effectively reducible.

(b) The negation of an effectively reducible formula is effectively reducible.

(c) Every formula of the form $\varphi(\bar{x}) = \exists_y \psi(\bar{x}, y)$ for an effectively reducible ψ and with $|\bar{x}| \geqslant 1$ is also effectively reducible.

Proof. Assertion 30 follows immediately from Lemma 29, because every quantifier-free formula is in particular existential. For assertion 30, after performing the constructions given by the effective reducibility of the formula $\varphi(\bar{x})$ in question, we may just negate the output lcd-reduced formula. Here we use that lcd-reduced formulas are closed under negation.

For assertion 30, apply first the effective reducibility of $\psi(\bar{x}, y)$, yielding an lcd-reduced formula $\beta(\bar{x}, y)$ working over some Γ-skeleton $\mathbb{C}$ guarded by $\mathbb{A}$. Then apply Lemma 27 and Proposition 23 to rewrite $\beta(\bar{x}, y)$ as an equivalent formula $\gamma(\bar{x}, y)$ in the prenex existential form. Finally, observe that formula $\exists_y \gamma(\bar{x}, y)$ is also in the prenex existential form, so we may apply Lemma 29 to it and $\mathbb{C}$, yielding the final lcd-reduced formula $\alpha(\bar{x})$ and skeleton $\mathbb{B}$. □

We may now conclude the proof of Theorem 9.

Proof (of Theorem 9). It is well-known that every formula can be rewritten into prenex normal form, i.e., to the form where there is a sequence of quantifiers (existential or universal) followed by a quantifier-free formula. Further, such a formula in the prenex normal form can be constructed from the quantifier-free formula using a sequence of negations and existential quantifications. Reducibility now follows from Lemma 30: assertion 30 states that the quantifier-free formula is reducible, while assertions 30 and 30 imply that during this construction procedure we encounter only reducible formulas.

For effectiveness, a sequential composition of the constructions yields both computability of the final depth, vocabulary, and formula from the input vocabulary and formula, as well as containment in linFPT of the transformation. For containment in para-AC^1, there is a slight caveat: a sequential composition of constructions would yield only containment in para-$\mathsf{AC}^{1\uparrow}$, and not in para-AC^1. We may solve this issue as follows. Observe that the only place in the proof where we need to use the computational power of para-AC^1 is when we invoke Theorem 8 to compute a treedepth-p coloring of the Gaifman graph; this happens in the beginning of the proof of Lemma 29. Since the final depth $d_{\max}$ is computable from the input vocabulary and formula, we may compute it beforehand. It is easy to see that during the construction procedure we never use treedepth-p colorings for any $p > d_{\max}$. Therefore, we may in the very beginning apply Lemma 29 on the Gaifman graph of the input structure for every value of $p \leqslant d_{\max}$ in *parallel*; this can be done in para-AC^1. Then all further applications of Lemma 29 can be replaced with the usage of a pre-computed treedepth-p coloring for an appropriate p. As the remainder of the construction actually works in para-$\mathsf{AC}^{0\uparrow}$, the result follows. □

5.4 Piecing together the proof of Theorem 1

With quantifier elimination in place, we may conclude the proof of our main result, Theorem 1.

Proof (of Theorem 1). Let $\mathbb{A}$ be an input Σ-structure on n elements, for a vocabulary Σ of arity at most 2, and let $\varphi \in \mathrm{FO}[\Sigma]$ be the input sentence. We would like to apply Theorem 9 to φ. However there is a slight mismatch: Theorem 9 assumes that the input formula has at least one free variable. To circumvent this, let z be a fresh variable that is not used in φ and let us consider φ as a formula $\varphi(z)$ with one free variable z that is never used. Note that $\varphi(z)$ is true either for every element of $\mathbb{A}$ or for no element of $\mathbb{A}$, depending on whether φ is true or false in $\mathbb{A}$. Now apply Theorem 9 to $\varphi(z)$, yielding a Γ-skeleton $\mathbb{B}$ of some depth d guarded by $\mathbb{A}$ and an lcd-reduced formula $\alpha(z) \in \mathrm{FO}[\Gamma]$ such that $\alpha(\mathbb{B}) = \varphi(\mathbb{A})$. It remains to evaluate α on any element of the structure using Lemma 28. □

The same reasoning allows to reprove the result of Dvořák et al. [16] that for every class of effectively bounded expansion $\mathscr{C}$, it can be verified in linear-FPT time whether an input sentence φ holds in a given structure whose Gaifman graph belongs to $\mathscr{C}$.

A cautious reader might be a bit worried by the strange-looking introduction of the dummy variable z in the proof above. Let us explain its combinatorial meaning. Unraveling the proof of Theorem 9, the need for this workaround is the assumption $k \geqslant 1$ in Lemma 26. This assumption is essential for the proof of Lemma 26 to work: the information about the existence of a suitable evaluation of y is encoded in the new label of x_1, and we need to have this variable x_1 in order to store the information somewhere. Moreover, the assumption is actually necessary for Lemma 26 to hold as stated, as an lcd-reduced formula with no free variables can only say "true" or "false", while whether a suitable evaluation of y exists depends on the forest $\mathbb{T}$, and not just on the input formula φ. A way to overcome the issue would be to generalize the notion of a labeled forest by allowing additional arity-0 relations (aka *flags*); then we could store the information in a flag in $\mathbb{S}$ and the output lcd-reduced sentence $\widehat{\varphi}$ would just output this flag. The implemented resolution by adding a dummy free variable z is a variant of this: the variable z serves as a "placeholder" for storing the relevant information, which boils down to encoding an arity-0 relation as an arity-1 relation that is satisfied either in all or in no element of the structure.

6 Conclusions

In this paper we showed that the model-checking problem for first-order logic on classes of effectively bounded expansion is in para-AC^1, which means that it can be solved by a family of AC-circuits of size $f(\varphi) \cdot n^{\mathcal{O}(1)}$ and depth $f(\varphi) + \mathcal{O}(\log n)$, where f is a computable function. This can be regarded as a parallelized variant of the result of Dvořák et al. [16] stating that the problem is fixed-parameter tractable.

By the result of Grohe et al. [27], model-checking FO is fixed-parameter tractable even on every nowhere dense class of structures. When trying to generalize our result to the nowhere dense setting, the main issue is that the proof of Grohe et al. [27] does not yield a robust quantifier elimination procedure, but a weak variant of Gaifman local form that is sufficient for fixed-parameter tractability of model-checking, but not variations of the problem.

Our techniques uncover tight connections between the paradigms of distributed computing and circuit complexity in the context of sparse graphs classes. Methods of the theory of sparsity seem very well-suited for the design of distributed algorithms, yet so far little is known. Nešetřil and Ossona de Mendez gave a logarithmic-time distributed algorithm to compute low treedepth colorings on classes of bounded expansion [35]. In the light of this paper, it is very natural to repeat the question asked by Nešetřil and Ossona de Mendez [35] of whether on every class of bounded expansion, model-checking local first-order formulas can be performed by a distributed algorithm with running time $f(\varphi) \cdot \log n$ in the *local broadcast* model. As computation of low treedepth colorings is already settled [35], it remains to examine the quantifier elimination procedure; we hope that our presentation of this argument may help with this. Stronger models of communication (such as the so-called *congested clique* model) may allow efficient distributed algorithms for more general problems, like model-checking of first-order formulas that are not necessarily local.

Acknowledgements. The authors thank Thomas Zeume for discussions on dynamic FO in the context of sparse graphs, which inspired this work.

References

[1] S. A. Amiri, P. Ossona de Mendez, R. Rabinovich, and S. Siebertz. Distributed domination on graph classes of bounded expansion. *CoRR*, abs/1702.02848, 2017.

[2] S. R. Arikati, A. Maheshwari, and C. D. Zaroliagis. Efficient computation of implicit representations of sparse graphs. *Discrete Applied Mathematics*, 78(1-3):1–16, 1997.

[3] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *FOCS 1989*, pages 364–369. IEEE Computer Society, 1989.

[4] M. Bannach, C. Stockhusen, and T. Tantau. Fast parallel fixed-parameter algorithms via color coding. In *IPEC 2015*, volume 43 of *LIPIcs*, pages 224–235. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2015.

[5] M. Bannach and T. Tantau. Parallel multivariate meta-theorems. In *IPEC 2016*, volume 63 of *LIPIcs*, pages 4:1–4:17. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2016.

[6] M. Bannach and T. Tantau. Computing Hitting Set kernels by AC^0-circuits, 2017. Manuscript, accepted for publication at STACS 2018.

[7] L. Barenboim. Deterministic $(\Delta + 1)$-coloring in sublinear (in Δ) time in static, dynamic, and faulty networks. *J. ACM*, 63(5):47:1–47:22, 2016.

[8] L. Barenboim and M. Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010.

[9] Y. Chen, J. Flum, and X. Huang. Slicewise definability in first-order logic with bounded quantifier rank. In *CSL 2017*, volume 82 of *LIPIcs*, pages 19:1–19:16. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2017.

[10] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[11] A. Dawar, M. Grohe, and S. Kreutzer. Locally excluding a minor. In *LICS 2007*, pages 270–279. IEEE Computer Society, 2007.

[12] A. Dawar and S. Kreutzer. Domination problems in nowhere-dense classes. In *FSTTCS 2009*, volume 4 of *LIPIcs*, pages 157–168. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2009.

[13] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[14] P. G. Drange, M. S. Dregi, F. V. Fomin, S. Kreutzer, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, F. Reidl, F. Sánchez Villaamil, S. Saurabh, S. Siebertz, and S. Sikdar. Kernelization and sparseness: the case of Dominating Set. In *STACS 2016*, volume 47 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2016.

[15] Z. Dvořák. Constant-factor approximation of the domination number in sparse graphs. *European Journal of Combinatorics*, 34(5):833–840, 2013.

[16] Z. Dvořák, D. Král', and R. Thomas. Testing first-order properties for subclasses of sparse graphs. *Journal of the ACM (JACM)*, 60(5):36, 2013.

[17] Z. Dvořák. *Asymptotical structure of combinatorial objects*. PhD thesis, Charles University, Faculty of Mathematics and Physics, 2007.

[18] Z. Dvořák. Constant-factor approximation of the domination number in sparse graphs. *Eur. J. Comb.*, 34(5):833–840, 2013.

[19] K. Eickmeyer, A. C. Giannopoulou, S. Kreutzer, O. Kwon, M. Pilipczuk, R. Rabinovich, and S. Siebertz. Neighborhood complexity and kernelization for nowhere dense classes of graphs. In *ICALP 2017*, volume 80 of *LIPIcs*, pages 63:1–63:14. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, 2017.

[20] M. Elberfeld, C. Stockhusen, and T. Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015.

[21] J. Flum and M. Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM J. Comput.*, 31(1):113–145, 2001.

[22] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[23] M. Frick and M. Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48(6):1184–1206, 2001.

[24] A. V. Goldberg, S. A. Plotkin, and G. E. Shannon. Parallel symmetry-breaking in sparse graphs. In *STOC 1987*, pages 315–324. ACM, 1987.

[25] M. Grohe and S. Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*, pages 181–206. American Mathematical Society, 2011.

[26] M. Grohe, S. Kreutzer, R. Rabinovich, S. Siebertz, and K. Stavropoulos. Colouring and covering nowhere dense graphs. In *WG 2015*, pages 325–338. Springer, 2015.

[27] M. Grohe, S. Kreutzer, and S. Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM (JACM)*, 64(3):17:1–17:32, 2017.

[28] N. Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[29] S. Kreutzer, R. Rabinovich, and S. Siebertz. Polynomial kernels and wideness properties of nowhere dense graph classes. In *SODA 2017*, pages 1533–1545. SIAM, 2017.

[30] J. Nešetřil and P. Ossona de Mendez. Grad and classes with bounded expansion I. Decompositions. *European Journal of Combinatorics*, 29(3):760–776, 2008.

[31] J. Nešetřil and P. Ossona de Mendez. First order properties on nowhere dense structures. *The Journal of Symbolic Logic*, 75(03):868–887, 2010.

[32] J. Nešetřil and P. Ossona de Mendez. On nowhere dense graphs. *European Journal of Combinatorics*, 32(4):600–617, 2011.

[33] J. Nešetřil and P. Ossona de Mendez. Grad and classes with bounded expansion II. Algorithmic aspects. *European Journal of Combinatorics*, 29(3):777–791, 2008.

[34] J. Nešetřil and P. Ossona de Mendez. *Sparsity — Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012.

[35] J. Nešetřil and P. Ossona de Mendez. A distributed low tree-depth decomposition algorithm for bounded expansion classes. *Distributed Computing*, 29(1):39–49, 2016.

[36] M. Pilipczuk and S. Siebertz. Lecture notes for the course "Sparsity" given at Faculty of Mathematics, Informatics, and Mechanics of the University of Warsaw, Winter Semester 2017/18. Available at https://www.mimuw.edu.pl/~mp248287/sparsity.

[37] D. Seese. Linear time computable problems and logical descriptions. *Electr. Notes Theor. Comput. Sci.*, 2:246–259, 1995.

[38] X. Zhu. Colouring graphs with bounded generalized colouring number. *Discrete Mathematics*, 309(18):5562–5568, 2009.

A Hardness of computing degeneracy exactly

In this section we prove Theorem 4, that is, we show that the problem of determining whether an input graph has degeneracy at most 2 is P-hard under logspace reductions. We reduce from the following Circuit Evaluation problem. We are given an AC-circuit C, with one output gate and no restrictions on depth, and an evaluation of the input gates of C. The task is to determine whether the only output gate of C evaluates to 1. It is well-known that this problem is P-complete under logspace reductions, since the computation of a polynomial-time deterministic Turing machine can be encoded as a polynomial-size AC-circuit.

Intuition. We first present the intuition behind the reduction. Suppose we are given a graph G and we are interested in finding out whether its degeneracy is at most 2. Consider the following *elimination procedure*: starting with the original graph G, iteratively remove a vertex of degree at most 2 from the current graph as long as there is one. If we manage to exhaust the whole vertex set of G by the elimination procedure, then by ordering the vertices according to the time of their removal we obtain an ordering of degeneracy at most 2. Otherwise, if the elimination procedure gets stuck at a subgraph with minimum degree at least 3, then by Proposition 3 this witnesses that the degeneracy of the graph is at least 3.

The idea for the reduction is as follows: given the input circuit C we construct a graph G by replacing each gate by an appropriate gadget so that the elimination procedure on G corresponds to the natural bottom-up evaluation procedure for the gates of C. More precisely, a gadget gets removed in the elimination procedure if and only if the corresponding gate evaluates to 1. If the output gate of C evaluates to 1 — which means that the corresponding gadgets gets removed — then this triggers a special "switch" that makes all gadgets removable, and hence G has degeneracy at most 2. Otherwise, all gadgets corresponding to gates that evaluate to 0 induce a subgraph of minimum degree at least 3, thus certifying that G has degeneracy at least 3.

Preprocessing. We first make some preprocessing of the input circuit C in order to streamline the construction; it will be straightforward to see that this pre-processing can be done in logarithmic space. Suppose the input gates of C are $x_1, \ldots, x_n$, the output gate of C is y, and we are also given an evaluation $\eta\colon \{x_1, \ldots, x_n\} \to \{0, 1\}$ of the input gates. We now introduce TRUE and FALSE gates; such gates have always fan-in 0 and evaluate to 1 and 0, respectivley. For each x_i such that $\eta(x_i) = 1$, replace the input gate x_i with a TRUE gate. Similarly, if $\eta(x_i) = 0$ then replace x_i with a FALSE gate. Moreover, every OR gate with no input is replaced with a FALSE gate and every AND gate with no input is replaced by a TRUE gate. Further, by the standard technique of eliminating negation using de Morgan's laws we may assume that C also has no NOT gates. Thus, from now on we may assume that C has only AND, OR, TRUE, and FALSE gates, and every AND and OR gate has fan-in at least 1.

Next, since we are not concerned with the depth of the circuit, we may assume that every gate has fan-in at most 2 by replacing each gate of larger fan-in with a binary tree of gates of the same type of fan-in 2. Similarly, we may further assume that every gate has fan-out at most 2, i.e. it is wired as the input to at most 2 other gates. For this, we modify every gate u with fan-out $d > 2$ by adding a path consisting of $d - 2$ AND-gates $u_1, u_2, \ldots, u_{d-2}$ with fan-in 1 and fan-out 2 arranged as follows: denoting $u_0 = u$, every gate u_i for $i \geqslant 1$ takes as the only input

the gate u_{i-1}. Thus, the additional AND-gates copy the evaluation of u and in total all the gates $u, u_1, \ldots, u_{d-2}$ have fan-out d; this fan-out d can be used to re-wire the d wires originally going out of u.

Finally, every OR-gate with fan-in 1 is replaced by an AND-gate with the same input and output — they have exactly the same functionality. All in all, we have achieved the following properties:

- circuit C has only AND, OR, TRUE, and FALSE gates, out of which there is one output gate y;
- each AND gate of C has fan-out at most 2 and fan-in 1 or 2;
- each OR gate of C has fan-out at most 2 and fan-in 2.

The problem is to determine whether the output gate y evaluates to 1.

Gadgets. We now present the gadgets for the OR/AND gates; they are depicted in Figure 2.

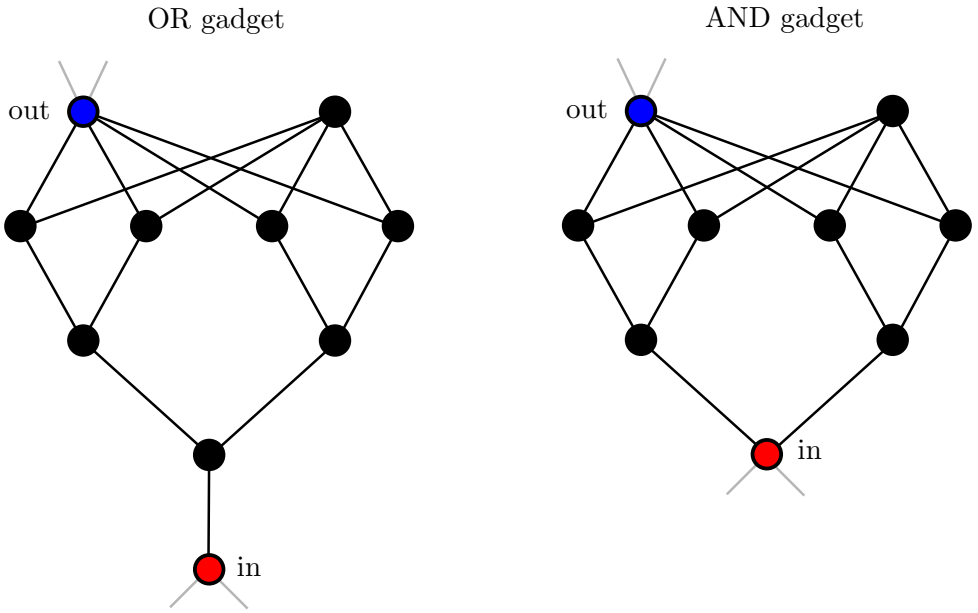

Figure 2: Gadgets for the OR/AND gates. Input vertices are depicted in red, output vertices are depicted in blue. The grey edges outgoing from the input and output vertices depict places where connections to other gadgets are attached.

The OR gadget has 10 vertices out of which there are two named: one *input vertex* named "in", and one *output vertex* named "out". Similarly for the AND gadget. We make the following two observations. First, in each gadget there is only one vertex of degree smaller than 3, which is the input vertex; in the OR gadget it has degree 1, and in the AND gadget it has degree 2. Further, each gadget has a vertex ordering with degeneracy 2 — just order the vertices in a top-down manner.

In the final construction we will use many copies of these gadget. For a copy A of the OR/AND gadget, by $\text{in}[A]$ and $\text{out}[A]$ we denote the input and output vertiex in A, respectively, and by $\sigma[A]$ we denote the abovementioned top-down vertex ordering of A with degeneracy 2.

Construction. We now present the construction of the graph G out of the circuit C. We first introduce gadgets to reflect the structure of C.

1. For each AND gate x in C, introduce a copy A_x of the AND gadget.
2. For each OR gate x in C, introduce a copy A_x of the OR gadget.
3. For each TRUE or FALSE gate x in C, introduce a copy A_x of the AND gadget.
4. Whenever there is a wire from gate z to gate x (i.e. z is an input to x), add an edge between $\text{out}[A_z]$ and $\text{in}[A_x]$.

Next, we perform the following construction similar to the reduction of fan-out. Let k be the number of FALSE gates. Introduce k AND gadgets $B_1, B_2, \ldots, B_k$, and denoting $A_y = B_0$ (recall that y is the output gate of C), add an edge between $\text{out}[B_{i-1}]$ and $\text{in}[B_i]$ for each $i \in [k]$. Finally, to each FALSE gate x assign a different integer $i \in [k]$ and add an edge between $\text{out}[B_i]$ and $\text{in}[A_x]$. This concludes the construction; it is clear that it can be implemented in logspace.

Correctness. To verify the correctness of the reduction we need to prove the following lemma.

Lemma 31. *Gate y evaluates to 1 in C if and only if the degeneracy of G is at most 2.*

The proof of Lemma 31 is divided into two lemmas, each showing one implication.

Lemma 32. *If gate y evaluates to 1 in C, then G has degeneracy at most 2.*

Proof. Since C is a circuit, it is acyclic and therefore there exists a topological ordering τ on the gates of C; that is, for every gate x of C, all the inputs of x are before x in τ. Construct a vertex ordering σ of G as follows:

1. First, concatenate the vertex orderings $\sigma[A_x]$ for all gates x of C that evaluate to 0, where the order of concatenation is the reverse of τ.
2. Next, append the vertex orderings $\sigma[B_k], \sigma[B_{k-1}], \ldots, \sigma[B_1]$, in this order.
3. Finally, append the concatentaion of the vertex orderings $\sigma[A_x]$ for all gates x of C that evaluate to 1, where again the order of concatenation is the reverse of τ.

We now verify that σ has degeneracy 2. Consider any vertex u of G; we check that at most two neighbors of u are placed before u in σ. If u is an internal (i.e. not input or output) vertex of any gadget, then all its neighbors are within the same gadget, and in the vertex ordering of this gadget at most two neighbors of u were placed before u. If u is an output vertex of any gadget, then it has four neighbors within this gadget and at most two outside of this gadget. However, all 4 neighbors within the gadget are placed after u in the vertex ordering of this gadget, hence u can have at most two neighbors placed before it in σ.

We are left with the case when u is the input vertex of some gadget. Consider first the case when $u = \text{in}[B_i]$ for some $i \in [k]$. Since B_i is an AND gadget, u has two neighbors within B_i, both placed before it in $\sigma[B_i]$, and one other neighbor $\text{out}[B_{i-1}]$. By the construction of σ and since y evaluates to 1 in C, the vertex $\text{out}[B_{i-1}]$ is placed after u in σ, hence u has only two neighbors placed before it in σ.

Next consider the case when $u = \text{in}[A_x]$ for some gate x that evaluates to 0 in G. If x is a FALSE gate, then A_x is a copy of the AND gadget and u has two neighbors within A_x, both placed before it in σ, and one neighbor in some B_i, which is placed after it in σ. If x is an OR/AND gate, then u has either one or two neighbors within A_x, placed before it in σ, and one or two neighbors in some other gadgets, say A_z and possibly $A_{z'}$. However, then $x \geqslant_\tau z$ and $x \geqslant_\tau z'$ and hence both A_z and $A_{z'}$ are placed entirely after A_x in σ — regardless whether z and z' evaluate to 0 or 1 in C.

Finally, consider the case when $u = \text{in}[A_x]$ for some gate x that evaluates to 1 in G. If x is a TRUE gate, then u has only two neighbors, both lying within A_x and placed before u in σ. If x is an OR gate, then u has one neighbor within A_x, placed before u in σ, and two neighbors in other gadgets, say A_z and $A_{z'}$, with $x \geqslant_\tau z$ and $x \geqslant_\tau z'$. Since x evaluates to 1, either z or z' evaluates to 1 as well, and consequently the corresponding gadget A_z or $A_{z'}$ is placed entirely after A_x in σ. Hence again u has at most two neighbors placed before u in σ. If x is an AND gate, then u has two neighbors within A_x, both placed before u in σ, and one or two neighbors in other gadgets, say A_z and possibly $A_{z'}$, with $x \geqslant_\tau z$ and $x \geqslant_\tau z'$. Since x evaluates to 1, both z and z' also need to evaluate to 1, and hence the corresponding gadgets A_z and $A_{z'}$ are both placed entirely after A_x in σ. So again u has at most two neighbors placed before it in σ.

Having considered all the cases, we conclude that indeed the degeneracy of σ is at most 2.□

Lemma 33. *If gate y evaluates to 0 in C, then the subgraph of G induced by all gadgets B_i, $i \in [k]$, and all gadgets A_x for gates x that evaluate to 0 in C has minimum degree at least 3.*

Proof. Let H be this induced subgraph of G. Observe that every non-input vertex of every gadget has at least three neighbors already within this gadget. Since gadgets are included in H in entirety, it follows that all non-input vertices contained in H have degree at least 3 in H.

It remains to show that every input vertex u in H also has degree at least 3 in H. Suppose first that $u = \text{in}[B_i]$ for some $i \in [k]$. Since B_i is a copy of the AND gadget, u has two neighbors within B_i and one neighbor being $\text{out}[B_{i-1}]$. All gadgets $A_y = B_0, B_1, \ldots, B_k$ are included in H, because y evaluates to 0 in C, so in particular $\text{out}[B_{i-1}]$ is also included in H.

Now suppose that $u = \text{in}[A_x]$ for some gate x that evaluates to 0 in C. If x is a FALSE gate, then u has two neighbors within A_x and one neighbor in some B_i, which is also included in H. If x is an OR gate, then u has one neighbor within A_x and two neighbors in other gadgets A_z and $A_{z'}$, where z and z' are the inputs to x. Since x evaluates to 0 in C, both z and z' have to evaluate to 0 in C as well, hence both A_z and $A_{z'}$ are included in H. Consequently, u has three neighbors in H. Finally, if x is an AND gate, then u has two neighbors within A_x and one or two neighbors in other gadgets, say A_z and possibly $A_{z'}$, where z and z' are the inputs to x. Again, since x evaluates to 0 in C, either z or z' have to evaluate to 0 in C as well, hence at least one of A_z and $A_{z'}$ is included in H. So again u has three neighbors in H.

Having considered all the cases, we conclude that H indeed has minimum degree at least 3.□

Lemma 31 now directly follows from Lemma 32 and Lemma 33: Lemma 32 provides the left-to-right implication, while Lemma 33 in combination with Proposition 3 provides the right-to-left implication. This concludes the proof of Theorem 4.

At the end we would like to remark that in Theorem 4 the constant 2 can be replaced by any integer $c \geqslant 2$. To see this, consider adding $c-2$ universal vertices (i.e. adjacent to all other vertices) to the graph G obtained in the reduction. It is not hard to see that this increases the degeneracy of the graph by exactly $c-2$.

B Proof of Lemma 14

Our presentation is based on the proof of Lemma 12 in [36, Lemma 8 of Chapter 2], while the proof itself is a generalization of the proof of Grohe et al. [26].

Proof (of Lemma 14). For the sake of contradiction suppose G does not admit any depth-$(r-1)$ topological minor with edge density larger than d. Let $\ell \coloneqq 6\varepsilon^{-1}rd^3$. Let $U \subseteq S$ be the set of those vertices $v \in S$ for which $\mathrm{bconn}_r(S,v) > \ell$. We know that $|U| \geqslant \varepsilon|S|$. For each $v \in U$, let us fix any path family $\mathcal{P}_v$ witnessing $\mathrm{bconn}_r(S,v) > \ell$. That is, $\mathcal{P}_v$ consists of more than ℓ paths in G such that each $P \in \mathcal{P}_v$ has length at most r, leads from v to another vertex of S, and all its internal vertices are outside of S, and moreover all paths in $\mathcal{P}_v$ pairwise share only the vertex v.

Let $\mathcal{Q}$ be an inclusion-wise maximal family of paths in G satisfying the following conditions:

- Each path $Q \in \mathcal{Q}$ has length at most $2r-1$, connects two different vertices in S, and all its internal vertices do not belong to S.
- Paths from $\mathcal{Q}$ are pairwise internally vertex-disjoint (i.e. they can share only the endpoints).
- For every pair of distinct vertices $u, v \in S$, there is at most one path from $\mathcal{Q}$ that connects u and v.

Consider a graph H on the vertex set S where $u, v \in S$ are adjacent if and only if there is a path in $\mathcal{Q}$ connecting u and v. Observe that paths in $\mathcal{Q}$ witness that H is a depth-r topological minor of G. Therefore, the edge density of H is at most d, implying that $|\mathcal{Q}| \leqslant d|S|$.

Observe further that every subgraph of H is also a depth-$(r-1)$ topological minor of G, and hence has edge density at most d. By the hand-shaking lemma, a graph of edge density at most d contains a vertex of degree at most $2d$. Consequently, every subgraph of H has minimum degree at most $2d$, which means, by Proposition 3, that H is $2d$-degenerate. By Proposition 5, H admits a proper coloring λ with $(2d+1)$-colors.

Coloring λ partitions U into $2d+1$ color classes. Let $I \subseteq U$ be the largest of them; then $|I| \geqslant \frac{|U|}{2d+1} \geqslant \frac{\varepsilon|S|}{2d+1} \geqslant \frac{\varepsilon|S|}{3d}$. Note that I is an independent set in H.

Let $K \coloneqq S \cup \bigcup_{Q \in \mathcal{Q}} V(Q)$. Since each path $Q \in \mathcal{Q}$ contains at most $2r-2$ internal vertices lying outside of S, and $|\mathcal{Q}| \leqslant d|S|$, we have

$$|K| \leqslant (1 + (2r-2)d)|S| \leqslant 2rd|S|. \tag{1}$$

For every $v \in I$, we construct a path family $\mathcal{P}'_v$ from $\mathcal{P}_v$ by trimming paths as follows. For a path $P \in \mathcal{P}$, let u be the first (closest to v) vertex of P that belongs to K; such a vertex always exists since the other endpoint of P belongs to S. Then we define P' as the prefix of P from v

to u, and we let $\mathcal{P}'_v$ to consist of all paths P' for $P \in \mathcal{P}_v$. The following assertions follow directly from the construction:

- $|\mathcal{P}'_v| = |\mathcal{P}_v|$;
- each path $P' \in \mathcal{P}'_v$ has length at most r, connects v with another vertex of K, and all its internal vertices do not belong to K; and
- paths from $\mathcal{P}'_v$ pairwise share only v, and in particular their endpoints other than v are pairwise different.

Let $\mathcal{R} \coloneqq \bigcup_{v \in I} \mathcal{P}'_v$. The following claim is the crucial step in the proof.

Claim 3. *Any two distinct paths $R, R' \in \mathcal{R}$ are internally vertex-disjoint and do not have the same endpoints.*

Proof. Suppose $R \in \mathcal{P}'_v$ and $R' \in \mathcal{P}'_{v'}$ for some $v, v' \in I$. If $v = v'$ then the claim follows from the properties of $\mathcal{P}'_v$ stated above, hence suppose otherwise. Suppose first that R and R' intersect at some vertex $w \notin K$; in particular w is an internal vertex of both R and R'. Consider the union of the prefix of R from v to w and the prefix of R' from v' to w. This union contains a path of length at most $2r-2$ connecting v and v', whose all internal vertices do not belong to K; call this path T. Since $v, v' \in I$, v and v' are non-adjacent in H, which means that in $\mathcal{Q}$ there is no path connecting v and v'. It follows that T could be added to $\mathcal{Q}$ without spoiling any of the conditions imposed on $\mathcal{Q}$, a contradiction with the maximality of $\mathcal{Q}$.

We are left with verifying that it is not the case that R and R' have exactly the same endoints v and v'. But in this case R would be a path of length at most r connecting v and v' that would be disjoint from K. So again R could be added to $\mathcal{Q}$ without spoiling any of the conditions imposed on $\mathcal{Q}$, which would contradict the maximality of $\mathcal{Q}$. ⌟

Consider now a graph J on the vertex set K where two vertices v, w are adjacent if and only if there is a path in $\mathcal{R}$ that connects them. By Claim 3 the paths in $\mathcal{R}$ witness that J is a depth-$(r-1)$ topological minor of G, so in particular the edge density of J is at most d, implying $|E(J)| \leqslant d|K|$. On the other hand, by Claim 3 every path in $\mathcal{R}$ gives rise to a different edge in $E(J)$, implying that $|E(J)| \geqslant |\mathcal{R}|$. By combining this with (1) we infer that

$$2rd^2|S| \geqslant d|K| \geqslant |\mathcal{R}| > |I| \cdot \ell \geqslant \frac{\varepsilon|S|}{3d} \cdot 6\varepsilon^{-1}rd^3 = 2rd^2|S|.$$

This is a contradiction. □